

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

**BİYOİNFORMATİK SEKANSLAR İÇİN DİZİ
EŞLEŞTİRME ALGORİTMALARI ÜZERİNE BİR
ÇALIŞMA**

Abdullah Ammar KARCIOĞLU

Danışman: Doç. Dr. Hasan BULUT

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı

İzmir
2022

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum “**Biyoinformatik Sekanslar İçin Dizi Eşleştirme Algoritmaları Üzerine Bir Çalışma**” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

22 / 02 / 2022

İmzası

Abdullah Ammar KARCIOĞLU

ÖZET**BIYOİNFORMATİK SEKANSLAR İÇİN DİZİ EŞLEŞTİRME
ALGORİTMALARI ÜZERİNE BİR ÇALIŞMA**

KARCIOĞLU, Abdullah Ammar

Doktora Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Hasan BULUT

Şubat 2022, 106 sayfa

Dizi arama, izinsiz giriş tespit sistemleri, dolandırıcılık tespiti, biyoinformatik sekanslarda örüntü arama gibi dizi eşleştirme algoritmaları bilgisayar biliminin temel alanları arasındadır. Kesin dizi eşleştirme algoritmaları tekli ve çoklu olmak üzere iki kısma ayrılır. Tekli kesin dizi eşleştirme algoritmaları, bir T metninde bir P dizisinin tüm oluşumlarını bulmayı içerir. Bu algoritmalar, öncelikle metin arama ve hesaplamalı biyoloji gibi çeşitli alanlardaki uygulamaları nedeniyle bilgisayar bilimlerinde kapsamlı bir şekilde incelenmiştir. Çoklu kesin dizi eşleştirme algoritmaları, belirli bir T metninde $|P|$ dizi kümesinin elemanlarını bulmayı içerir. Kesin dizi eşleştirme algoritmalarının temel amacı, mümkün olan en kısa zaman çerçevesinde tüm dizi eşleşmelerini doğru bir şekilde bulmaktır. Bu nedenle literatürde var olan kesin dizi eşleştirme algoritmalarındaki eksiklikler tespit edilip daha verimli yeni yaklaşımlar geliştirilmelidir.

Bu tez çalışmasında, DNA sekanslarına yönelik tekli ve çoklu kesin dizi eşleştirme algoritmaları geliştirilmiştir. Dizi eşleştirme işlemleri, DNA dizileri için zaman açısından verimli bir şekilde yapılmalıdır. T metninin hacmi arttıkça ve aranan dizilerin sayısı arttıkça toplam çalışma süresi artar. Bu arama işlemlerini en kısa sürede gerçekleştirmek için verimli algoritmalar seçilmelidir. Yapılan deneysel çalışmalar sonucu geliştirilen yaklaşımların literatürdeki diğer yaklaşımlara kıyasen performans metrikleri açısından daha başarılı sonuçlar elde ettiği gösterilmiştir.

Anahtar sözcükler: Dizi Eşleştirme Algoritmaları, Desen Eşleştirme, DNA Sekansları, Sekans Analizi, Hash Fonksiyonu, Biyoinformatik Algoritmalar, Hesaplamalı Biyoloji.



ABSTRACT**A STUDY ON STRING MATCHING ALGORITHMS FOR
BIOINFORMATIC SEQUENCES**

KARCIOĞLU, Abdullah Ammar

Ph.D. in Computer Engineering

Supervisor: Assoc. Dr. Hasan BULUT

February 2022, 106 pages

The string matching algorithms are among the essential fields in computer science, such as text search, intrusion detection systems, fraud detection, and pattern search in bioinformatics sequences. The exact string matching algorithms are divided into two parts as single and multiple. Single exact string matching algorithms involve finding all occurrences of a pattern P in a text T . These algorithms have been extensively studied in computer science, primarily because of their applications in various fields such as text search and computational biology. Multiple string matching algorithms involve finding elements of the pattern set $|P|$ in a given input text T . The main goal of exact string matching algorithms is to find all pattern matches correctly within the shortest possible time frame. For this reason, the limitations of exact string matching algorithms in the literature should be determined and more efficient new approaches should be developed.

In this thesis, single and multiple exact string matching algorithms for DNA sequences have been developed. String matching processes should be done in a time-efficient manner for DNA sequences. As the volume of the text T increases and the number of search patterns increases, the total runtime increases. Efficient algorithms should be selected to perform these search operations as soon as possible. As a result of the experimental studies, it has been shown that the developed approaches have achieved more successful results in terms of performance metrics compared to other approaches in the literature.

Keywords: String Matching Algorithms, Pattern Matching, DNA Sequences, Sequence Analysis, Hash Function, Bioinformatic Algorithms, Computational Biology.



ÖNSÖZ

Bu tez çalışmasında, DNA sekansları için kesin dizi eşleştirme algoritmalarındaki eksiklikler tespit edilerek daha verimli yeni algoritmaların geliştirilmesi hedeflenmiştir. Bu konuda çalışmaya danışman hocam Doç. Dr. Hasan BULUT'un tavsiyeleri üzerine karar verdim. DNA sekansı veri setleri açık kaynak olarak paylaşım sağlayan NCBI sitesinden metin formatında indirilmiştir. Bu veri setlerine uygulanan kesin dizi eşleştirme algoritmaları incelendikten sonra hash tabanlı kesin dizi eşleştirme algoritmalarında eksiklikler olduğu dikkatimizi çekti. DNA sekanslarına yönelik hash çakışmalarını ortadan kaldıran bir hash fonksiyonu geliştirilmesinin literatüre ne gibi katkı sağlayacağını merak ettik. Bu merak üzerine giderek tez çalışmamı şekillendirdim. DNA sekanlarında hash çakışmasını ortadan kaldıran yeni bir hash fonksiyonu geliştirdik. Bu fonksiyonu hash tabanlı tekli ve çoklu kesin dizi eşleştirme algoritmalarına uygulayarak yeni algoritmalar geliştirdik. Geliştirmiş olduğumuz dizi eşleştirme algoritmalarının literatürdeki diğer algoritmalarından tüm performans metriklerinde daha başarılı sonuçlar ürettiği gösterilmiştir. Geliştirdiğimiz algoritmaların bu alanda çalışacak olan araştırmacılara öncülük edeceğini düşünmekteyim.

İZMİR

22 Şubat 2022

Abdullah Ammar KARCIOĞLU

İÇİNDEKİLER

Sayfa

ÖZET	v
ABSTRACT	vii
ÖNSÖZ	ix
İÇİNDEKİLER	xi
ŞEKİLLER DİZİNİ	xvi
ÇİZELGELER DİZİNİ	xix
SİMGELER VE KISALTMALAR DİZİNİ	xxii
1. GİRİŞ	1
2. ÖNCEKİ ÇALIŞMALAR	5
2.1 Kesin Dizi Eşleştirme Algoritmaları	6
2.1.1 Tekli Kesin Dizi Eşleştirme Algoritmaları	6
2.1.1.1 Hash Tabanlı Tekli Kesin Dizi Eşleştirme Algoritmaları.....	8
2.1.2 Çoklu Kesin Dizi Eşleştirme Algoritmaları.....	12
2.2 Yaklaşık Dizi Eşleştirme Algoritmaları.....	14
3. VERİ SETLERİ	16
4. GELİŞTİRİLEN MODELLER.....	18
4.1. DNA Sekansları İçin Mükemmel Hash Fonksiyonuna Dayalı Hash-q Kesin Dizi Eşleştirme Algoritmaları.....	18

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
4.1.1. Motivasyon.....	18
4.1.2. Önerilen Mükemmel Hash Fonksiyonu	21
4.1.3. UniqueFNG’li Hash-q Algoritması (HqUF)	23
4.1.4. UniqueFNG’li Hash-q Boyer-Moore Algoritması (HqUF-BM).....	24
4.1.5. Önerilen Yaklaşımların Genel Çerçevesi	26
4.1.6. Önerilen Hash Fonksiyonunun Karakter Karşılaştırma Sayısı Üzerindeki Etkisine Örnekler.....	27
4.1.7. Önerilen Yaklaşımlar İçin Çalışan Örnekler	29
4.2. DNA Sekansları İçin q-Çerçeve Hash Karşılaştırmasına Dayalı Kesin Dizi Eşleştirme Algoritmaları	31
4.2.1. Eşsiz q-Çerçeve Hash Karşılaştırmalı Hash-q Algoritması (Hq-QF)	32
4.2.2. Eşsiz q-Çerçeve Hash Karşılaştırmalı Hash-q Boyer-Moore Algoritması (HqBM-QF).....	34
4.2.3. Karakter Sayısı ve Hash Karşılaştırmalarına İlişkin Önerilen Yaklaşımlara Örnekler	36
4.2.4. Önerilen Yaklaşımlar İçin Çalışan Örnekler	37
4.3. DNA Sekansları İçin WM-q Çoklu Kesin Dizi Eşleştirme Algoritması.....	39
4.3.1. WM Algoritması	39
4.3.2. Önerilen Yaklaşım: WM-q Algoritması.....	41
4.3.3. Önerilen Yaklaşımın Başarılı ve Başarısız Eşleşme Durum Örnekleri...	43

İÇİNDEKİLER (devam)

Sayfa

4.3.4. Önerilen Yaklaşımın Genel Çerçevesi.....	46
4.3.5. Önerilen Yaklaşım İçin Çalışan Örnekler.....	47
4.4. DNA Sekansları İçin q-Gram Hash Karşılaştırmasına Dayalı Çoklu Kesin Dizi Eşleştirme Algoritması	51
4.4.1. Önerilen Yaklaşım: q-Gram WM Algoritması.....	52
4.4.2. WM ve q-Gram WM Algoritmalarının Karşılaştırma Sayısı Örnekleri.	54
4.4.3. Önerilen Yaklaşımın Genel Çerçevesi.....	57
4.4.4. Önerilen Yaklaşımın Çalışan Örneği.....	58
5. DENEYSEL ÇALIŞMALAR.....	61
5.1. Hq-UF ve HqUF-BM Algoritmalarının Deneysel Çalışma Sonuçları	61
5.1.1. E. Coli Veri Seti Sonuçları	62
5.1.2. Sentetik Veri Seti Sonuçları.....	63
5.1.3. Human Chromosome 1 Veri Seti Sonuçları	65
5.1.4. İyi Bilinen Algoritmalarla Deneysel Karşılaştırma	67
5.1.4.1. E. Coli Veri Seti Sonuçları	67
5.1.4.2. Sentetik Veri Seti Sonuçları.....	67
5.1.4.3. Human Chromosome 1 Veri Seti Sonuçları	68
5.1.5. Sonuçlar ve Gelecek Çalışmalar	69

İÇİNDEKİLER (devam)

Sayfa

5.2. Hq-QF ve HqBM-QF Algoritmalarının Deneysel Çalışma Sonuçları	70
5.2.1. E. Coli Veri Seti Sonuçları	71
5.2.2. Human Chromosome 1 Veri Seti Sonuçları	73
5.2.3. E. Human Chromosome 2+3 Veri Seti Sonuçları	76
5.2.4. İyi Bilinen Algoritmalarla Deneysel Karşılaştırmalar	79
5.2.4.1. E. Coli Veri Seti Sonuçları	79
5.2.4.2. Human Chromosome 1 Veri Seti Sonuçları	79
5.2.4.3. Human Chromosome 2+3 Veri Seti Sonuçları	80
5.2.5. Sonuçlar	81
5.3. WM-q Algoritmasının Deneysel Çalışma Sonuçları	82
5.3.1. Ortalama Çalışma Süresi Sonuçları	82
5.3.2. Ortalama Karakter Karşılaştırma Sayısı Sonuçları	83
5.3.3. Ortalama Hash Karşılaştırma Sayısı Sonuçları	84
5.3.4. Önerilen Yaklaşımın İyi Bilinen Algoritmalarla Karşılaştırılması	85
5.3.5. Sonuçlar ve Gelecek Çalışmalar	87
5.4. q-Gram WM Algoritmasının Deneysel Çalışma Sonuçları	88
5.4.1. WM ve q-Gram WM Algoritmasının Karşılaştırılması	89
5.4.1.1. Ortalama Çalışma Süresi Sonuçları	89

İÇİNDEKİLER (devam)Sayfa

5.4.1.2. Ortalama Karakter Karşılaştırma Sayısı Sonuçları.....	90
5.4.1.3. Ortalama Hash Karşılaştırma Sayısı Sonuçları	90
5.4.2. İyi Bilinen Algoritmalarla Karşılaştırma	91
5.4.3. Sonuçlar ve Gelecek Çalışmalar	93
6. SONUÇ.....	94
7. TARTIŞMA.....	95
KAYNAKLAR DİZİNİ.....	96
TEŞEKKÜR	104
ÖZGEÇMİŞ.....	105

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
Şekil 2.1. Dizi eşleştirme tanımı (Hakak et al., 2019).	5
Şekil 2.2. Dizi eşleştirme algoritmalarının sınıflandırılması (Hakak et al., 2019).....	6
Şekil 2.3. Karp-Rabin algoritması örneği (Hakak et al., 2019).....	9
Şekil 2.4. Hash tabanlı kesin dizi eşleştirme algoritmalarının sınıflandırılması (Hakak et al., 2019).....	11
Şekil 2.5. Yaklaşık dizi eşleştirme örneği.	15
Şekil 3.1. E. Coli ve Sentetik Veri Setlerinin DNA Baz Karakterlerinin Frekansı.	17
Şekil 3.2. Human Chromosome 1,2,3,2+3 Veri Setlerinin DNA Bazlarının Frekansı.	17
Şekil 4.1. Önerilen hash fonksiyonunun $q=3$ için hash değeri hesaplaması.	22
Şekil 4.2. HqUF ve HqUF-BM yaklaşımların genel çerçevesi.	26
Şekil 4.3. Eşleşmeme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karakter karşılaştırma örneği ($q=3$ için).	28
Şekil 4.4. Eşleşme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karakter karşılaştırma örneği ($q=3$ için).	28
Şekil 4.5. Hash-q algoritmasının $q=3$ için çalışan örneği.	29
Şekil 4.6. HqUF algoritmasının $q=3$ için çalışan örneği.	30
Şekil 4.7. HqUF-BM algoritmasının $q=3$ için çalışan örneği.	31
Şekil 4.8. Önerilen Hq-QF ve HqBM-QF yaklaşımların genel çerçevesi.....	32

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
Şekil 4.9. Eşleşmeme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karşılaştırma örneği (q=3 için).....	37
Şekil 4.10. Eşleşme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karşılaştırma örneği (q=3 için).....	37
Şekil 4.11. Hash-q algoritmasının çalışan örneği (q=3 için).	38
Şekil 4.12. Hq-QF algoritmasının çalışan örneği (q=3 için).	38
Şekil 4.13. HqBM-QF algoritmasının çalışan örneği (q=3 için).	39
Şekil 4.14. WM algoritmasının eşleşmeme durumu.....	43
Şekil 4.15. WM-q algoritmasının eşleşmeme durumu (sonek eşleşmemesi). ..	44
Şekil 4.16. WM-q algoritmasının eşleşmeme durumu (önek eşleşmemesi)....	45
Şekil 4.17. WM algoritmasının eşleşme durumu.	45
Şekil 4.18. WM-q algoritmasının eşleşme durumu.	46
Şekil 4.19. Önerilen WM-q algoritmasının genel çerçevesi.	47
Şekil 4.20. WM algoritmasındaki atlama ve önek tabloları.	48
Şekil 4.21. WM-q algoritmasındaki atlama ve önek tabloları.	48
Şekil 4.22. WM algoritmasının çalışan örnekleri.	49
Şekil 4.23. WM-q algoritmasının çalışan örnekleri.....	50
Şekil 4.24. WM algoritmasının başarısız dizi eşleştirme durumu.	55
Şekil 4.25. q-Gram WM algoritmasının başarısız dizi eşleştirme durumu.	55

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
Şekil 4.26. WM algoritmasının başarılı dizi eşleştirme durumu.....	56
Şekil 4.27. q-Gram WM algoritmasının başarılı dizi eşleştirme durumu.	56
Şekil 4.28. Önerilen q-Gram WM yaklaşımının genel çerçevesi.....	57
Şekil 4.29. WM algoritmasındaki sonek ve önek tabloları.	58
Şekil 4.30. q-Gram WM algoritmasındaki sonek ve önek tabloları.	58
Şekil 4.31. WM algoritmasının çalışan örneği.....	59
Şekil 4.32. q-Gram WM algoritmasının çalışan örneği.	59
Şekil 5.1. WM-q algoritmasının ortalama çalışma süresi sonuçları (d=1000,3000,5000).....	83
Şekil 5.2. İyi bilinen algoritmalarla WM-q algoritmasının ortalama çalışma süresi sonuçları (d=1000,3000,5000).	86

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
Çizelge 3.1. Tez çalışmasında kullanılan tüm veri setleri.	16
Çizelge 4.1. Skip-q algoritmasının farklı hash değerlerinin sayısı.	19
Çizelge 4.2. Hash-q algoritmasının farklı hash değerlerinin sayısı.	20
Çizelge 4.3. Önerilen hash fonksiyonunun farklı hash değerlerinin sayısı.	23
Çizelge 4.4. Kesin dizi eşleştirme algoritmalarının zaman karmaşıklığı.	27
Çizelge 4.5. İyi bilinen kesin dizi eşleştirme algoritmalarının zaman karmaşıklık analizi.	36
Çizelge 4.6. Çoklu dizi eşleştirme algoritmalarının zaman karmaşıklığı.	51
Çizelge 5.1. HqUF ve HqUF-BM algoritmalarında kullanılan veri setleri.	61
Çizelge 5.2. Ortalama çalışma süresi (e. coli).	62
Çizelge 5.3. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (e. coli).	63
Çizelge 5.4. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (e. coli).	63
Çizelge 5.5. Ortalama çalışma süresi (sentetik veri).	64
Çizelge 5.6. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (sentetik veri).	64
Çizelge 5.7. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (sentetik veri).	65
Çizelge 5.8. Ortalama çalışma süresi (human chromosome 1).	65

ÇİZELGELER DİZİNİ (devam)

<u>Çizelge</u>	<u>Sayfa</u>
Çizelge 5.9. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 1).	66
Çizelge 5.10. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 1).	66
Çizelge 5.11. İyi bilinen algoritmalarda ortalama çalışma süreleri (e. coli).	67
Çizelge 5.12. İyi bilinen algoritmalarda ortalama çalışma süreleri (sentetik). .	68
Çizelge 5.13. İyi bilinen algoritmalarda ortalama çalışma süreleri (human chromosome 1).	68
Çizelge 5.14. Hq-QF ve HqBM-QF algoritmalarında kullanılan veri setleri.	70
Çizelge 5.15. Ortalama çalışma süreleri (e. coli).	71
Çizelge 5.16. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (e. coli).	72
Çizelge 5.17. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (e. coli).	73
Çizelge 5.18. Ortalama çalışma süreleri (human chromosome 1).	74
Çizelge 5.19. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 1).	75
Çizelge 5.20. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 1).	76
Çizelge 5.21. Ortalama çalışma süreleri (human chromosome 2+3).	77
Çizelge 5.22. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 2+3).	78

ÇİZELGELER DİZİNİ (devam)

<u>Çizelge</u>	<u>Sayfa</u>
Çizelge 5.23. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 2+3).	78
Çizelge 5.24. Ortalama çalışma süresi (e. coli).	79
Çizelge 5.25. Ortalama çalışma süresi (human chromosome 1).	80
Çizelge 5.26. Ortalama çalışma süresi (human chromosome 2+3).	80
Çizelge 5.27. WM-q algorimasında kullanılan veri setleri.	82
Çizelge 5.28. Ortalama karakter karşılaştırma sayısı sonuçları (d=1000,3000,5000).	84
Çizelge 5.29. Ortalama hash karşılaştırma sayısı sonuçları (d=1000,3000,5000).	85
Çizelge 5.30. q-Gram WM algorimasında kullanılan veri setleri.	88
Çizelge 5.31. Ortalama çalışma süresi.	89
Çizelge 5.32. Ortalama karakter karşılaştırma sayısı.	90
Çizelge 5.33. Ortalama hash karşılaştırma sayısı.	91
Çizelge 5.34. İyi bilinen algoritmarda ortalama çalışma süresi.	92

SİMGELER VE KISALTMALAR DİZİNİ

<u>Kısaltmalar</u>	<u>Açıklama</u>
KMP	Knuth, Morris ve Pratt
BM	Boyer-Moore
HA	Hızlı Arama
BR	Berry-Ravindran
FNG	Parmak İzi
WM	Wu-Manber
DNA	Deoksiribo Nükleik Asit
A	Adenine, Adenin
C	Cytosine, Sitozin
G	Guanine, Guanin
T	Thymine, Timin
HTTP	Hyper-Text Transfer Protocol
URL	Uniform Resource Locator
BMH-q	Boyer-Moore-Horspool-q
KK	Kaba Kuvvet
AC	Aho Corasick
CM	Commentz Walter
QS	Quick Search, Hızlı Arama
QWM	Quick WM
AFWM	Adres Filtreleme Tabanlı WM
HqUF	UniqueFNG'li Hash-q Algoritması
HqUF-BM	UniqueFNG'li Hash-q Boyer-Moore Algoritması
Hq-QF	Eşsiz q -Çerçeve Hash Karşılaştırmalı Hash-q
HqBM-QF Moore	Eşsiz q -Çerçeve Hash Karşılaştırmalı Hash-q Boyer-Moore

1. GİRİŞ

Dizi eşleştirme, tıp ve biyoinformatik gibi birçok alandaki uygulamaları nedeniyle en çok çalışılan problemlerden biri olmuştur. Yeni etkili yaklaşımlar, dizi eşleştirme süreçlerini hızlandırarak bu alanlardaki çalışmalara yardımcı olmaktadır. Örneğin, dizi eşleştirme işlemlerinin performansı artırılarak hastalıklara neden olan Deoksiribo Nükleik Asit (DNA) dizileri daha verimli bir şekilde tespit edilebilir (Mane and Pangu, 2016; Upama et al., 2015). Bir veya daha fazla karakterin DNA dizileri için belirli bir kod sözcüğü ile temsil edildiği bit kodlama (Wandelt et al., 2014; Deorowicz and Grabowski, 2013), DNA dizilerini sözlük tabanlı temsil ederek dizi eşleştirme (Bakr and Sharawi, 2013), büyük DNA veri tabanlarında indekslemeye dayalı benzerlik araması (Jeong et al., 2010), DNA dizi aramasını hızlandırmak için budama tekniği (Khitmoh et al., 2020), DNA dizilerini sıkıştırdıktan sonra dizi eşleştirme (Chen and Ram, 2004), DNA dizilerini sınıflandırmak için çiçek filtrelerini kullanma (Najam et al., 2019), DNA dizilerindeki kümeleme tabanlı dizi eşleştirme algoritmaları (Kanuga, 2015) ve DNA dizilerindeki hash tabanlı kesin dizi eşleştirmesi (Karp, 1987; Wu and Manber, 1994; Lecroq, 2007; Tarhio and Pentola, 1997; Faro, 2016; Kim and Kim, 1999) gibi DNA dizileri için dizi eşleştirme konusunda çeşitli çalışmalar gerçekleştirilmiştir. Kesin dizi eşleştirme, DNA dizilerinde en çok çalışılan araştırma konularından biri olmuştur.

Kesin dizi eşleştirme algoritmaları, n uzunluğunda bir T metinde, m uzunluğundaki bir P dizinin tüm oluşumlarını bulur. Dizi, metnin başlangıcına hizalanır ve dizi baştan sona doğru kontrol edilir. Dizi eşleşirse, var olduğu bildirir. Bu işlem metnin sonuna kadar devam eder. Böylece, eğer varsa, dizinin bir veya daha fazla tüm oluşumları bulunur. Kaba kuvvet (KK) dizi eşleştirme algoritması, karakter tabanlı dizi eşleştirme algoritmalarından biridir. Ayrıca dizi karşılaştırması için kaba bir yaklaşım olarak kabul edilir. Bu algoritmada dizi, metnin başlangıcına göre hizalanır. Daha sonra dizi karakterleri soldan sağa tek tek metin karakterleriyle karşılaştırılır. Dizi tam olarak eşleşirse, metinde bulunan dizinin başlangıç konumunu bildirir. Eşleşmezse, dizi bir birim sağa kaydırılarak aynı işlemler metnin sonuna kadar devam eder. KK'nin zaman karmaşıklığı $O(nm)$ 'dir.

Çoklu kesin dizi eşleştirme algoritmaları, bir T girdi metni içinde bir veya daha fazla dizi içeren bir $|P|$ dizi kümesindeki dizilerin oluşumlarını bulmayı amaçlar. Metin arama ve sahtecilik, bilgi alma, web filtreleme, izinsiz

giriş, algılama sistemleri, virüs tarayıcıları, spam filtreleri ve biyoinformatik sekans arama gibi çok çeşitli çalışma alanlarına sahiptir (Faro, 2016).

Literatürde, k -mers araçları gibi çalışmalar, uzun diziler için bir veri tabanı (Greenfield and Roehm, 2013) ve DNA'larda dizi eşleştirme için sözlük tabanlı yaklaşımlar (Zhang et al., 2021) kullanan k -mer dizilerinin hızlı ve basit tam eşleşmelerine dayanmaktadır. Ayrıca, istemci-sunucu çerçevelerinde daha hızlı işlem yapılmasını sağlayan kodların yeniden yazılmasını sağlamak için dizi eşleştirme ve analiz yapılmıştır (Gupta et al., 2019).

Büyük ölçekli veri akışları için çoklu dizi eşleştirmede dinamik dizi eşleştirme (Sukhanov et al., 2020), çoklu dizi eşleştirme için hızlı algoritmalar geliştirme (Song et al., 2021), büyük ölçekli veri kümeleri için çoklu dizi eşleştirmeyi hızlandırmak için sıkıştırılmış veri kullanma (Aldwairi et al., 2019), bağlamdan bağımsız gramerler kullanılarak çoklu dizi eşleştirme (Kumar et al., 2019), ASCII tabanlı çoklu dizi eşleştirme algoritmalarının geliştirilmesi (Singh and Sharma, 2017), küçük uzunluklu diziler için verimli çoklu dizi eşleştirme (Faro and Külekçi, 2013) gibi literatürde çoklu dizi eşleştirme algoritmaları kullanılarak birçok çalışma gerçekleştirilmiştir. Literatürde, guguk kuşu filtrelemeye dayalı derin paket incelemesi için çoklu dizi eşleştirmenin paralelleştirilmesi (Ho et al., 2018), çoklu dizi eşleştirme algoritmalarıyla başa çıkmak için Rabin Karp algoritmasının paralelleştirilmesi (Nunes et al., 2020), iki adımlı paralel programlama kullanılarak başka çalışmalar da yapılmıştır. Paralel çoklu dizi eşleştirme algoritması (Lai et al., 2019), çoklu dizi eşleştirme algoritmalarında çalışma zamanını azaltmak için GPU tabanlı algoritma (Rasool et al., 2019), mükemmel hash fonksiyonunu kullanarak Boyer Moore Horspool algoritmasına (Lin et al., 2017) dayalı paralel çoklu dizi eşleştirme algoritması geliştirilmiştir.

Son zamanlarda, biyoinformatik diziler için tam eşleştirme ve doğru konumlandırma modellerine artan bir ilgi olmuştur. DNA dizilerindeki bu çalışmalar, DNA dizilerinde dizi eşleştirme (Karcioglu and Bulut, 2021), oligonükleotitleri eşleştirmek için yüksek hesaplama maliyetini en aza indiren dizi eşleştirme yöntemlerini geliştirme (Hyyrö et al., 2005), sözlük eşleştirme işleminde hastalık adını belirlemek için etkili dizi eşleştirme algoritmalarını geliştirme (Xu et al., 2019), biyoinformatikte çoklu dizi palindromlarının haritalanması (Kim and Han, 2016), doktorların kanser ve diğer hastalıkların tedavisinde kişiselleştirilmiş teşhisler gerçekleştirmesini sağlayan ilişkili bir model oluşturma (Yang et al., 2018), oldukça benzer DNA dizileri için etkili

çoklu dizi eşleştirme (Al-Qiari and Al-Issa, 2018) gibi çok sayıda çoklu dizi eşleştirme algoritmaları geliştirilmiştir.

Bu tez çalışmasında, DNA sekanslarına yönelik tekli ve çoklu kesin dizi eşleştirme algoritmaları geliştirilmiştir. Geliştirilen algoritmalar hash tabanlı algoritmalarlardır. DNA sekanslarında hash çakışmasını ortadan kaldıran mükemmel hash fonksiyonu geliştirilmiştir ve bu hash fonksiyonu önerilen tekli ve çoklu kesin dizi eşleştirme algoritmalarına uygulanmıştır. Mükemmel hash fonksiyonu sayesinde literatüre DNA sekanslarında dizi eşleştirme işlemlerinde katkılar sağlanmıştır. Geliştirilen algoritmalar ile literatürdeki bazı hash tabanlı kesin dizi eşleştirme algoritmalarının performansları artırılmıştır.

Tez çalışmasının literatüre temel katkıları şu şekilde özetlenebilir:

- DNA dizileri için hash çakışmasını ortadan kaldıran mükemmel hash fonksiyonu önerilmiştir.
- Önerilen hash fonksiyonuna dayalı, geleneksel Hash-q algoritmasının ilk iyileştirmesi olarak UniqueFNG algoritmali Hash-q algoritması önerilmiştir.
- Önerilen hash fonksiyonuna dayalı, geleneksel Hash-q algoritmasına ikinci bir iyileştirme olarak UniqueFNG algoritmali Hash-q Boyer-Moore algoritması önerilmiştir.
- Karakter karşılaştırması yapmak yerine q -çerçeve hash karşılaştırmasına dayalı iki farklı tekli kesin dizi eşleştirme algoritması önerilmiştir.
- Önerilen iki algoritmada DNA dizileri için hash çakışmasını içermeyen mükemmel bir hash fonksiyonu kullanılmıştır. İlk yaklaşımda, son q karakter için bir hash eşleşmesinden sonra, Hash-q algoritmasındaki karakter karşılaştırma işlemi, q -çerçeve hash karşılaştırması ile değiştirilmiştir. İkinci yaklaşımda, iyi sonek kaydırma tablosunda $(m - 1)$ 'inci değerde belirtilen kaydırma miktarını kullanarak birinci yaklaşım geliştirilmiştir.
- DNA dizilerindeki hash çakışmalarını ortadan kaldıran mükemmel hash fonksiyonu, çoklu kesin dizi eşleştirme algoritmalarından olan Wu-Manber algoritmasına uygulanmıştır.

- DNA veri setlerinde arama yaparken q -uzunluklu sonek ve önek dizilerinin karakter karşılaştırmalarını yapmaya gerek kalmamıştır.

- Çakışmasız hash fonksiyonu sayesinde, Wu-Manber algoritmasında sabit blok uzunluğu $B = 2$ yerine önerilen WM- q algoritmasında $3 \leq q \leq 8$ aralığında daha büyük blok uzunlukları kullanılmıştır.

- Önek uyuşmazsa, metin üzerinde $B = 2$ yerine $3 \leq q \leq 8$ 'e kadar diziler kaydırılmıştır ve metin üzerinde daha hızlı eşleştirme sağlanmıştır.

- Geleneksel Wu-Manber algoritmasının aksine, DNA sekanslarında hash çakışmasını ortadan kaldıran hash fonksiyonunu kullanarak dizilerdeki arama işlemi q -gram hash karşılaştırması ile gerçekleştirilmiştir.

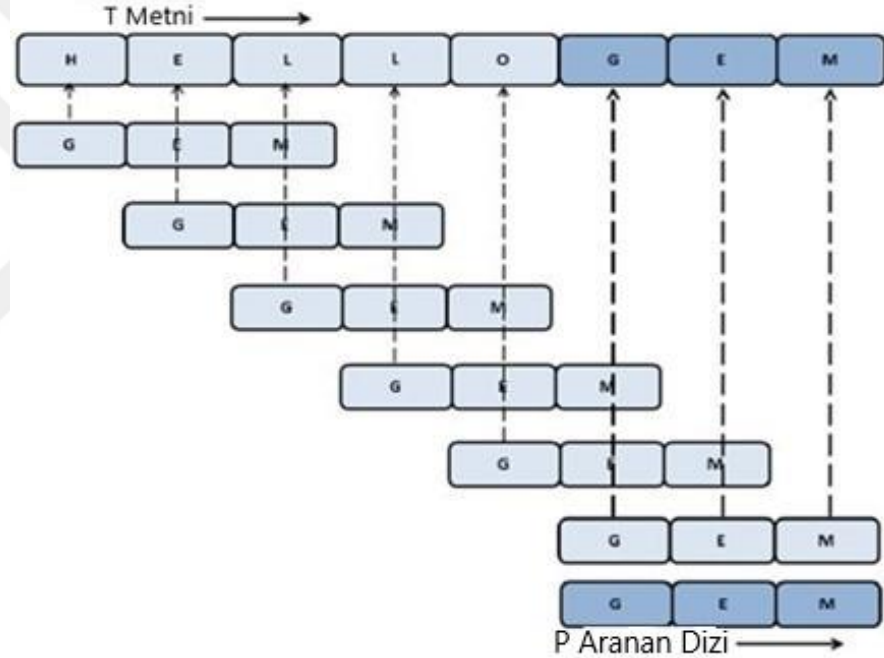
- Geliştirilen tüm algoritmalarda ortalama çalışma zamanı, ortalama karakter karşılaştırma sayısı ve ortalama hash karşılaştırma sayısı gibi performans metriklerinde başarılı sonuçlar elde edilmiştir.

Doktora eğitim ve tez sürecinde farklı problemler için çeşitli çalışmalar gerçekleştirilmiştir. Bu çalışmalar geliştirilen yöntemlerle literatürde iyi bilinen algoritmalar belli veri setleri üzerinde çalıştırılarak performans metrikleri açısından karşılaştırılmıştır. Elde edilen başarılı sonuçlar bilimsel yayınlar ile desteklenmiştir.

Tezin ikinci bölümünde literatürde yer alan geçmiş çalışmalardan bahsedilmiştir. Tezin üçüncü bölümünde, deneysel çalışmalarda kullanılan tüm veri setleri tanıtılmıştır. Tezin dördüncü bölümünde, tez kapsamında geliştirilen modellerden ve deneysel bulgulardan bahsedilmiştir. Tez çalışması kapsamında dört farklı çalışma gerçekleştirilmiştir. Geliştirilen modeller kapsamında her bir çalışmadan ve bu çalışmalarda elde edilen deneysel bulgulardan kendi içlerinde bahsedilmiştir. Son olarak beşinci bölümde, çalışmanın sonuçlarına yer verilmiştir, altıncı bölümde sonuçlardan genel olarak bahsedilmiştir ve yedinci bölümde ise elde edilen bulgular tartışılmıştır.

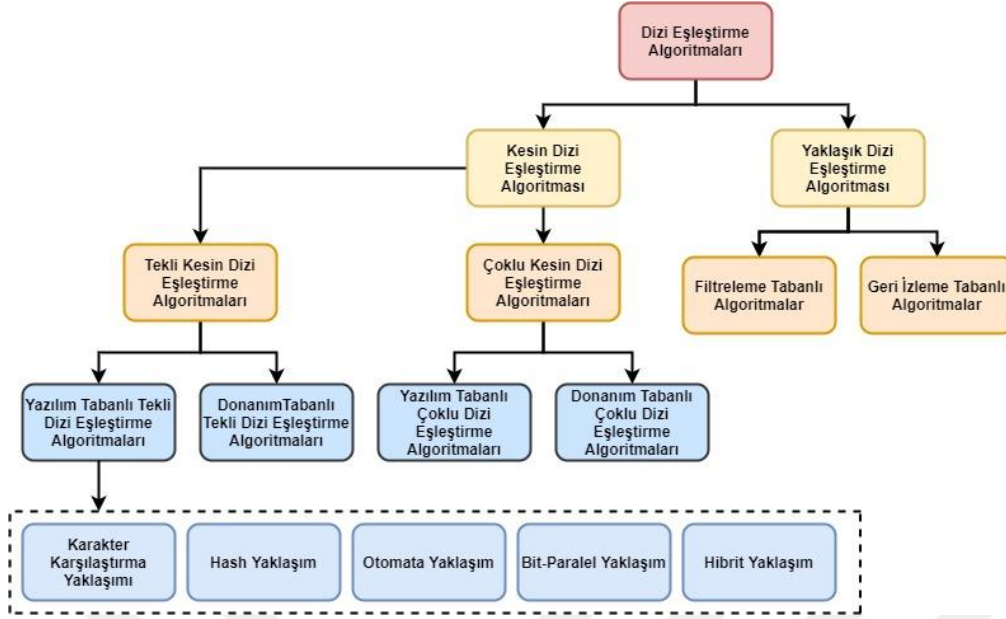
2. ÖNCEKİ ÇALIŞMALAR

Dizi eşleştirmenin asıl görevi, alfabe sırasına bakılmaksızın veri seti dizisinde aranan diziye bulmak olduğundan, bir dizi sıralı veya sıralanmamış olabilir. Dizi eşleştirme kavramı, Şekil 2.1'de gösterildiği gibi belirli bir T metninde bir P dizisinin bütün oluşumlarının bulunmasıdır. Şekil 2.1, belirli bir uzunluktaki bir dizi içindeki belirtilen uzunluktaki bir alt dizinin aranmasını göstermektedir. Bu işlem büyük miktarda hesaplama içerir ve bu nedenle zaman alıcıdır. En iyi dizi eşleştirme algoritmaları, literatürdeki problemlerin uygulama ve karmaşıklığına göre çok sayıda algortmadan seçilmektedir. Bu nedenle, onların güçlü ve zayıf yönlerini araştırmak için dizi eşleştirme algoritmalarını geliştirmeyi hedeflemekteyiz.



Şekil 2.1.Dizi eşleştirme tanımı (Hakak et al., 2019).

Dizi eşleştirme algoritmaları Şekil 2.2'de gösterildiği gibi kesin ve yaklaşık dizi eşleme algoritmaları olarak iki ana kategoriye ayrılır. Adından da anlaşılacağı gibi, kesin dizi eşleştirmede dizi metinde tam olarak aranır, oysa yaklaşık dize eşleştirmede, dizi metinde k uyumsuzluğuna kadar aranır, burada k dizi uzunluğuna kıyasla küçük bir sayı olarak kabul edilir. Kesin dizi eşleştirme algoritmaları da tekli ve çoklu dizi eşleştirme yöntemleri olarak sınıflandırılır. Bu iki kategori, yazılım ve donanım tabanlı yöntemler olarak sınıflandırılır. Yazılım tabanlı kesin dizi eşleştirme algoritmaları beş kategoriye ayrılır: karakter karşılaştırma, hash, otomata, bit-paralel ve hibrit yaklaşımlardır (Hakak et al., 2019).



Şekil 2.2. Dizi eşleştirme algoritmalarının sınıflandırılması (Hakak et al., 2019).

2.1 Kesin Dizi Eşleştirme Algoritmaları

Kesin dizi eşleştirme algoritmalarında, belirli bir metindeki verilen bir dizinin tüm oluşumları bulunur. Bu dizi eşleşmesinde, bir dizi penceresinde ve bir metin penceresinde bulunan karakterler karşılaştırılır. Her iki pencerenin uzunluğu karşılaştırma aşamasında eşit uzunlukta olmalıdır. Uyumsuzluk durumunda karakter kaydırılması etkin algoritmalar geliştirmek için gereklidir (AbdulRazzaq et al., 2013).

2.1.1 Tekli Kesin Dizi Eşleştirme Algoritmaları

Tekli kesin dizi eşleştirme algoritmalarında, algoritma girdi olarak sadece bir dizi alır ve bu belirli diziyi hedef veri tabanında (veya başka deyişle bir metin üzerinde) arar. Bu grup ayrıca donanım ve yazılım tabanlı eşleştirmenin iki alt grubuna ayrılır. Bazı uygulamalar, DNA'daki mutasyonların analizinde olduğu gibi birden fazla aranacak dizi gerektirmektedir. Bu tür uygulamalar için de çoklu dizi eşleştirme algoritmaları önerilmiştir (Alfred, 2014).

Tekli kesin dizi eşleştirme problemini doğrusal zamanda çözen algoritmalarından biri Knuth, Morris ve Pratt'dir (KMP) (Faro and Lecroq, 2013). KMP algoritması (Knuth et al., 1977) sonlu durumlu makine otomasyonuna dayanmaktadır. P dizisi, geçişi kabul eden bir M sonlu durum makinesi oluşturmak için önceden işlenir. Sonlu durum makinesi genellikle bir geçiş tablosu olarak temsil edilir. Ortalama ve en kötü durum performansı için

algoritmanın karmaşıklığı $O(n + m)$ 'dir. Boyer-Moore algoritması (BM) (Boyer and Moore, 1977), kaydırma miktarını maksimize etmeyi amaçlayan çeşitli algoritmaların öncüsüdür. BM, P dizisinin karakterlerini sağdan sola T metninin karakterleriyle karşılaştırır. Dizi için maksimum kaydırma değerini belirlemek için "İyi Sonek Tablosu" ve "Kötü Karakter Tablosu"nun kaydırma değerlerini kullanır. Bu algoritmanın en kötü durum karmaşıklığı $O(n + m)$ ve ortalama durum karmaşıklığı $O(n/m)$ 'dir. Horspool algoritması (Horspool, 1980) BM algoritmasının basitleştirilmiş bir versiyonudur. En kötü zaman karmaşıklığında ikinci dereceden olmasına rağmen, pratik durumlarda daha iyi performans gösterir. Horspool algoritmasında "İyi Sonek Tablosu" kullanılmamaktadır. Algoritmanın zaman karmaşıklığı $O(nm)$ 'dir.

Sunday tarafından geliştirilen Hızlı Arama (HA) algoritması (Sunday, 1990) Horspool algoritmasının basit bir versiyonudur. Her denemeden sonra kaydırma değeri, $[s \dots s + m - 1]$ karakter penceresine dayalı olarak hesaplanır; burada s , metnin hizalanmış indisidir. HA algoritmasının geliştirilmiş versiyonu olan Berry-Ravindran (BR) algoritmasında (Bery and Ravindran, 1999), dizi, metin karakterlerinin alt dizisinin en sağına hizalanacak şekilde ilerletilir. Bu iki algoritmanın zaman karmaşıklığı $O(nm)$ 'dir. Benzersiz oligonükleotitler içeren bir DNA veri tabanında dizileri aramak, hesaplama açısından maliyetlidir. Bununla başa çıkmak için on adet dizi eşleştirme yöntemi uygulanmış ve bunların en verimlisi işlemeyi hızlandırmak için seçilmiştir (Hyrrö et al., 2005). Başka bir çalışmada, biyomedikal metinleri bulmak için minimum dizi kimliği uygulanarak optimum metin eşleştirmesi sağlanmıştır (Wang et al., 2005). Ayrıca, sözlük eşleştirme için hastalık adlarını etkin ve doğru bir şekilde eşleştiren verimli kesin dizi eşleştirme yöntemi uygulanmıştır (Xu et al., 2019). İki nümerik olarak temsil edilen dizi arasındaki benzerliği ve eşleşmeyi kartezyen ağaçları kullanılarak benzerlik ve eşleşmenin gerçekleştirilmesi (Park et al., 2020), kartezyen ağaçları kullanılarak alt dizilerin eşleştirilmesi (Oizumi et al., 2022), örnek otomatasının iki adımlı simülasyonuna dayalı etkili dizi eşleştirme algoritmalarının geliştirilmesi (Faro and Scafiti, 2021) gibi son zamanlarda dizi eşleştirme algoritmaları geliştirilmiştir.

2.1.1.1 Hash Tabanlı Tekli Kesin Dizi Eşleştirme Algoritmaları

Dizi eşleştirme işlemlerinde çalışma zamanı açısından daha verimli algoritmalar geliştirmek için hash tabanlı dizi eşleştirme algoritmaları geliştirilmiştir. Karp-Rabin algoritması (Karp and Rabin, 1987), hash tabanlı algoritmalara öncülük etmiştir. Bu algoritmanın temel amacı, tüm diziyi baştan sona kontrol etmek yerine, dizi ve metin arasındaki benzerliğe dayanarak daha hızlı eşleştirme bulmaktır. Ön işleme aşamasında dizi h hash fonksiyonundan geçirilir ve bir hash değeri elde edilir. Bu ön işleme aşaması, $O(m)$ zaman karmaşıklığına sahiptir. Arama aşamasında dizi metnin soluna hizalandıktan sonra, metin karakterleri belirtilen h hash fonksiyonundan geçirilir. Üretilen hash fonksiyonunun değerlerine parmak izi (*FNG*) denir. Hash değerleri eşitse, dizinin tüm karakterleri karşılaştırılır. Hash değerleri eşit değilse, bir eşleşme olduğu anlaşılar ve dizi, metinde m karakterden az olana kadar bir birim sağa kaydırılır. Arama aşamasının zaman karmaşıklığı $O(nm)$ 'dir. Hash tabanlı algoritmalarından bazıları Karp-Rabin (Karp and Rabin, 1987), Wu-Manber (Wu and Manber, 1994), Hash-q (Lecroq, 2007), BMH-q (Tarhio and Pentola, 1997), Skip-q (Faro, 2016), S. Kim (Kim and Kim, 1999) algoritmalarıdır.

Hash tabanlı yaklaşımlar, Şekil 2.3'te gösterildiği gibi, karakter çiftlerini karşılaştırmadan önce, ilk önce dizilerin tamamından veya bir kısmından oluşturulan hash değerleri karşılaştırır. Böylece eşleştirme aşamasında hızlanma sağlanır. Dizi (veya alt dizi) için oluşturulan hash değerlerinin ve metindeki karşılık gelen karakterlerin eşleşmediğini varsayalım. Bu durumda, bir eşleşme olduğunda dizi algoritmalar tarafından belirlenen miktar kadar sağa kaydırılır. Ancak, hash değerleri oluşturmak da zaman alıcı olabilir. Bu nedenle, karakter karşılaştırma ve hash değerleri oluşturma arasında bir zaman maliyeti (trade-off) vardır (Kanuga, 2015). Karp ve Rabin dizi eşleştirme problemi için hash tabanlı yaklaşımlara öncülük etmiştir. Karp-Rabin algoritması, m uzunluklu dizi için bir hash değeri üretir. Bir aramada, metnin hizalanmış bölgesi için hash değeri oluşturulur ve karşılaştırılır. Hash değerleri eşleşirse, dizideki karakterler ile metinde karşılık gelen karakterler karşılaştırılır. Kullanılan hash fonksiyonu, Karp-Rabin algoritmasında hash çakışmasına neden olabilir. Dizinin ve metnin tüm karakter çiftleri birbiriyle eşleşirse, dizi metin üzerinde bir birim sağa kaydırılır. Hash değerleri eşleşmezse veya karakter karşılaştırması sırasında bir eşleşme oluşursa, dizi de bir birim sağa kaydırılır. Algoritma, hash değerlerinde bir eşleşme olması durumunda karakter karşılaştırmalarını kaydeder. En kötü durumda zaman

karmaşıklığı $O(nm)$ 'dir. Ancak, uzun diziler (Greenfield and Roehm, 2013) için eşleştirme işlemi yavaştır ve hash fonksiyonundaki büyük asal sayılar taşmalara neden olabilir (Zhang et al., 2021).



Şekil 2.3. Karp-Rabin algoritması örneği (Hakak et al., 2019).

Ön işleme aşamasında genellikle bir dizi ondalık sayıya dönüştürülür. Yani, c karakter dizisi d ondalık sayı dizisine (Radix tabanlı) dönüştürülür. Tüm dizi karakterleri için hash, $m - 1$ 'e kadar hesaplanır ve m karakterlerini içeren bir dizi penceresidir. Hash değerleri Horner's kuralı kullanılarak hesaplanır. P dizisi, önceden tanımlanmış asal sayı olan q 'ya bölünür. Mod alma işlemi daha sonra p dizisinin geri kalanını q ile hesaplamak için kullanılır. Atlama(shift) $s = 0$ ile $n - m$ arasında değişen her kaydırma değeri, dizi ve metnin geri kalanını eşleştirme için karşılaştırılır. Bir eşleşme bulunduğunda, eşleşmenin sonucunu doğrulamak için Şekil 2.3'te gösterildiği gibi kaba kuvvet yaklaşımı uygulanır. Bu örnekte, verilen bir t metni ve bir p dizisidir. Aranacak dizinin sayı değeri 8628'dir, bu da belirli bir kalanın elde edilmesi için önceden belirlenen asal sayıya bölünür, bu örnek için kalan 4'tür. Bu nedenle, arama aşamasında, her yeni dizi penceresi, bu pencerenin kalanının ve p 'nin aynı olup olmadığını kontrol etmek için aynı asal sayıya bölünür. Koşul pozitifse, durumu doğrulamak için kaba kuvvet yaklaşımı kullanılır.

• Verilen Metin $t = 12348628$, Dizi $p = 8628, q = 11$

• $Kalan(t) = 6, Kalan(p) = 4$. Ancak mod alma işlemi nedeniyle aşağıdaki sorunlar ortaya çıkabilir.

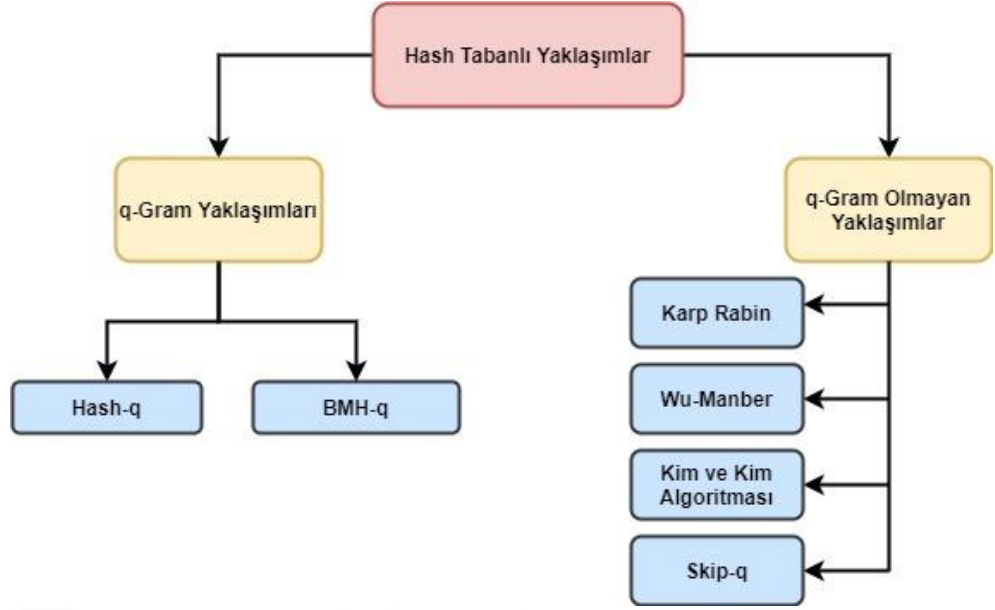
1) Kalan $r1 = r2$ ise eşleşme başarılı olur.

2) Eğer kaba kuvvetten sonra kalan $r1 = r2$ 'dir. Fakat $r1 \neq r2$ ise, o zaman çakışma olur.

3) Kalan $r1 \neq r2$ ise, eşleşme başarısız olur.

Literatürde diziler ve DNA sekansları için birçok hash tabanlı yaklaşım geliştirilmiştir. Daha hızlı bir eşleştirme işlemi için HTTP protokolünde URL eşleştirme için çok modellenli ikili-hash tabanlı hibrit bir algoritma geliştirilmiştir (Zeng et al., 2017). Ben-Bassat ve Chor (2014), dizi grafiğini oluşturmak için Karp-Rabin algoritmasını değiştirmede yeni bir hash tabanlı yöntem kullanmıştır. Policriti ve Prezza (2015), kısa özet veri yapılarıyla hızlı rastgele yaklaşık dizi eşleştirmesi için Hamming mesafesini kullanmıştır. Takenaka ve arkadaşları (2011), hash tablosunun boyutunu arttırmadan eşleşme bulmak için hash referanslarının sayısını azaltan, DNA sekansı haritalama için hash tabanlı bir yöntem sunmuştur. Yu ve Liu'nun (2020) çalışmasında, hash tabanlı öğrenme yöntemine dayalı olarak transkriptomları verimli bir şekilde okuyan bit eşleme RNA dizisi önerilmiştir. Yıldız (2019, 2020) yüksek performanslı sorgular için bitmap dizin kodlamasını kullanmıştır. Çıbıkdiken ve arkadaşları (2017), hash fonksiyonlarının kullanıldığı biyoinformatik sekanslar için dağıtılmış depolama sistemi aracılığıyla yeni bir platform geliştirmiştir.

Hash tabanlı dizi eşleştirme algoritmaları, Şekil 2.4'te gösterildiği gibi, hash değerinin hesaplanmasına göre q -gramlar ve q -gramlar olmayan yaklaşımlar olarak gruplandırılır (Hakak et al., 2019). q -gram yaklaşımlarında, bir dizi n tane alt diziye bölünür. Bu yöntemde literatürde çeşitli yaklaşımlar geliştirilmiştir. Lecroq tarafından geliştirilen Hash- q algoritması (Lecroq, 2007), $(m - q + 1)$ adet q -uzunluklu alt dizileri üreten bir dizi eşleştirme algoritmasıdır. Arama aşamasında, yaklaşım her q -uzunluklu alt dizisi için hash değeri hesaplar. Alt dizilerin hash değerleri eşit ise karakterler KK algoritmasındaki gibi karşılaştırılır. Ek olarak, hash fonksiyonu, $3 \leq q \leq 8$ için 0-255 arasında değerleri arasında birçok hash çakışması üretir.



Şekil 2.4. Hash tabanlı kesin dizi eşleştirme algoritmalarının sınıflandırılması (Hakak et al., 2019).

Bir q -gram yaklaşımı olan Boyer-Moore-Horspool- q (BMH- q) (Tarhio and Pentola, 1997) algoritması, Horspool algoritmasının bir versiyonudur ve DNA karakterlerinde etkilidir. Bu yaklaşımda A, C, G, T harfleri sırasıyla 0, 1, 2, 3 ile eşleştirilir. Daha sonra bu yeni değerler kullanılarak q -çerçevelerinin hash değerleri hesaplanır. Ancak bu algoritma, q 'nın tüm değerleri için değil, yalnızca $q = 4$ için eşsiz hash değerleri üretir. Ayrıca algoritma, 32-bit okumaya izin veren x86 mimarileri için geliştirilmiştir.

Q -gram olmayan yaklaşımlarda, tüm dizi için hash değeri hesaplanır. WM algoritması (Wu and Manber, 1994), bir T metni içindeki tüm dizilerin oluşumlarını inceler. q değeri, alt dizilerin uzunluğudur. Kaydırma işlemi, olası tüm q -uzunluklu diziler için ön işleme aşaması sırasında hesaplanır. q uzunluğundaki tüm B alt dizilerinin hash değerleri, belirtilen h hash fonksiyonu kullanılarak hesaplanır. Arama aşamasında, B uzunluğundaki alt dizilerin hash değerleri 0'dan büyükse, dizi bu hash değerine karşılık gelen kaydırma miktarı kadar kaydırılır. Hash değeri 0'a eşitse, dizi bir birim sağa kaydırılır. WM algoritmasının en kötü durum karmaşıklığı $O(n(T(h) + m))$ 'dir, burada $T(h)$, hash değerinin hesaplama süresidir.

S. Kim ve Y. Kim'in algoritması (Kim and Kim, 1999), WM algoritmasını geliştirilmiş bir versiyonudur ve bir hash şemasına ve birleştirilmiş kodlamaya dayanır. Algoritma çoklu diziler için uygundur ve büyük diziler için etkilidir. Simone, yoğunlaştırılmış alfabe için atlamalı arama

algoritmasına (Charras et al., 1998) dayanan Skip-q algoritmasını (Faro, 2016) önermişlerdir. Ön işleme aşamasında her alt dizi için bir hash değeri hesaplanır ve bu değerler ardından kaydırma tablosunda tutulur. Ek olarak, her dizi için tüm olası q -uzunluklu alt diziler oluşturulur. Arama aşamasında, atlamalı arama algoritmasına benzer şekilde, metnin her bir konum indisi için bir hash değeri hesaplanır ve dizinin hash değeri ile karşılaştırılır. Hash değerleri eşleşirse, dizinin karakterleri metne karşılık gelen karakterlerle karşılaştırılır. Skip-q algoritmasında hash fonksiyonu tüm alfabe için değerler ürettiğinden, DNA alfabeleri için hash çakışmaları meydana gelir. DNA veri setlerindeki alfabe sayısı $\Sigma = 4$, $\{A, C, G, T\}$ şeklindedir. Bu durumda, oluşabilecek farklı hash değerlerinin toplam sayısı Σ^q olur. Ancak Skip-q algoritmasında kullanılan hash fonksiyonu mükemmel hash değerleri üretilmediği için üretilen farklı hash değerlerinin sayısı Σ^q değerinden çok daha azdır.

2.1.2 Çoklu Kesin Dizi Eşleştirme Algoritmaları

Çoklu kesin dizi eşleştirme algoritmaları $T = t_0 t_1 t_2 \dots t_{n-1}$ olan n uzunluklu girdi dizisi içinde d adet $P = p^0 p^1 p^2 \dots p^{d-1}$ ve m uzunluklu $p^r = p_0^r p_1^r \dots p_{m-1}^r$ her bir dizinin tüm oluşumlarının bulunmasına dayanmaktadır. DNA sekanslarında Adenin (A), Sitozin (C), Timin (T) ve Guanin (G) olmak üzere 4 farklı baz kullanılmaktadır. Dizilerin toplam uzunluğu ise $|P|$ 'dir.

Verilen n uzunluklu veri setinde d adet dizinin KK algoritması kullanarak aranmasının zaman karmaşıklığı $O(n|P|)$ 'dir. Girdi dizisinin ve aranan dizilerin uzunluğu arttıkça işlem süresi artacağından bu yöntem verimli değildir. Bu yüzden literatürde çoklu dizilerin etkili bir şekilde aranılması için bazı algoritmalar geliştirilmiştir. Bu algoritmalar genellikle, birden çok diziyi eşzamanlı olarak arama yaparak çalışma zamanını etkili kullanma mantığına dayanmaktadır. Çoklu kesin dizi eşleştirme algoritmaları sonek, önek, faktör ve hash tabanlı algoritmalar olmak üzere dört farklı kategoriye ayrılır (Kouzinopoulos and Margaritis, 2014).

Önek tabanlı algoritmalar: Aranılan diziler düğümlerde temsil edilir. Burada kök düğüm boş bir diziyi temsil ederken, diğer düğümler ise her bir dizinin önekini ifade eder. Aho Corasick (AC) (Aho and Corasick, 1975) algoritması KMP (Knuth et al., 1977) algoritmasına dayalı olup literatürde önek tabanlı çoklu kesin dizi eşleştirme algoritması olarak sıkça kullanılmaktadır. AC algoritmasının zaman karmaşıklığı lineer zamandır, yani $O(n)$.

Sonek tabanlı algoritmalar: Önek tabanlı algoritmaları ters yönlü otomata olarak temsil eder. Literatürde sıkça kullanılan sonek tabanlı çoklu dizi eşleştirme algoritması Commentz Walter (CM) algoritmasıdır (Commentz-Walter, 1979). AC algoritmasının Boyer-Moore (Boyer and Moore, 1977) algoritmasındaki kötü karakter tablosu eklenerek geliştirilmiş versiyonu CM algoritmasıdır. CM algoritmasının AC algoritmasından farklı olarak ön işleme aşamasında tersine durum makinesi oluşturur. CM algoritmasının zaman karmaşıklığı alt-lineer'dir.

Faktör tabanlı algoritmalar: Dizilerin herhangi bir alt dizisini tanıyabilen ek geçişlere sahip bir metin ağacı (trie) oluştururlar. Faktör tabanlı çoklu dizi eşleştirme algoritmaları olarak Set Backward Oracle & Dawg Matching algoritmaları geliştirilmiştir (Navarro and Raffinot, 2002).

Hash tabanlı algoritmalar: Literatürde geliştirilen algoritmalar kullanılan bellek açısından verimli ve hızlı değildirler. Bu yüzden çoklu dizi eşleştirme işlemlerinde belleği ve zamanı daha verimli kullanmak için diziler verilen bir matematiksel hash fonksiyonundan geçirilerek elde edilen sonuca göre temsil edilirler. WM algoritması hash tabanlı çoklu dizi eşleştirme algoritmalarına öncülük etmektedir ve dizileri belirlenen blok uzunluklarında okuyarak eşleştirme yapar.

Literatürde WM algoritması kullanılarak, karakterlerin eşleşme olasılığının uyumsuzluk olasılığından daha yüksek olduğunu düşünen QPS algoritması (Al-Qiari and Al-Issa, 2018) ve kaydırma boyutunu artırmak için WM algoritmasına Hızlı Arama (QS) algoritmasını (Sunday, 1990) ekleyen Quick WM (QWM) algoritması (Ke and Young, 2006) gibi birçok algoritma geliştirilmiştir. Ayrıca, tek bir SHIFT tablosu yerine, Zhen ve Di (2008), daha fazla kaydırma boyutu sağlamak için iki SHIFT tablosu kullanmıştır. Diğer çalışmalar arasında SHIFT ve HASH tablolarındaki çakışmaları ortadan kaldıran geliştirilmiş Quick WM algoritması (QWM) (Dai, 2009), WM algoritmasının dizilerin çok kısa ve çok uzun olmasından kaynaklanan performans sorununa çözüm sunan HCWM algoritması (Zhang et al., 2009) önerilmiştir. Ayrıca, uzun diziler ve PREFIX tablosunu sık kullanmayan adres filtreleme tabanlı WM algoritması (AFWM) (Zhang et al., 2009) önerilmiştir.

2.2 Yaklaşık Dizi Eşleştirme Algoritmaları

Yaklaşık dizi eşleştirme algoritması, belirli bir dizi dizisine yakın bir alt dizi bulur. Bu algoritma, tam bir eşleşme bekleyen kesin dizi eşleştirme algoritmasına aykırıdır. Bu durumda, yakınlık derecesine karar vermek, sorunların uygulanmasına ve karmaşıklığına bağlı olarak zorluk gösterebilir.

Wu ve Manber'a göre (1994), k adet uyumsuzluk olduğunu kabul edersek, eşleştirme sırasında bu k adet sayıdan küçük eşit olan tüm eşleşmemelerin olduğu yerlerde alt dizilerin bulunduğu kabul edilmektedir. Kesin eşleştirme algoritmalarından farklı olarak belli sayılarda eşleşmemeler göz ardı edilmektedir. Başka bir deyişle, algoritmalar sayı eşleşmelerini sayar ve alt dizileri dizilerle eşleştirirken bazı eşik değerlerini k olarak sabitler. Bu yaklaşım genellikle, dizi ile verilen metin arasında k uyumsuzluğu bulunduğu kullanılır.

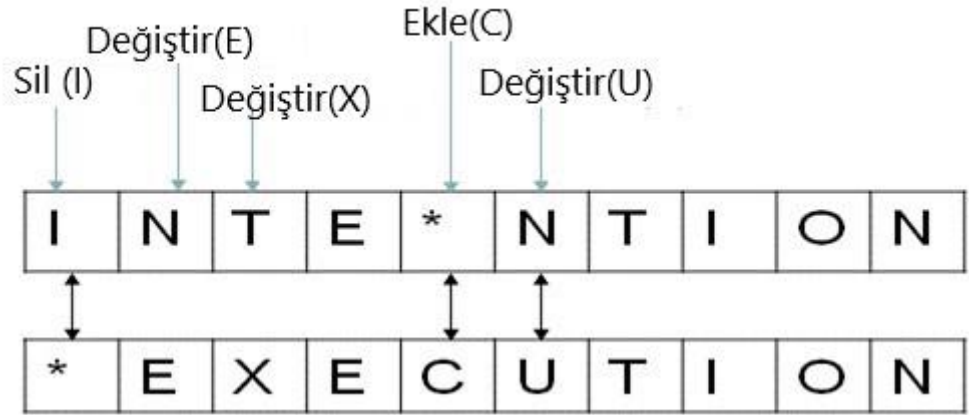
Bu yaklaşımda Hamming mesafe ölçümü ve Levenshtein mesafe ölçümü gibi iki popüler mesafe ölçümü kullanılmıştır (Sankoff, 1983; Levenstein, 1965). Bu algoritmalar, kalıplarda veya metinlerde bulunan imla hatalarını, düşük metin kalitesini ve yabancı isimleri arama zorluğunu ele almak için tanıtılmıştır (Saikrishna et al., 2012). Bu alana öncülük eden diğer algoritmalar ise Ukkonen (1995), Myers (1999) algoritmalarıdır. Ancak kesin dizi eşleştirme algoritmalarında belirli sayıda eşleşmeme durumu uygulanırsa yaklaşık dizi eşleştirme algoritmalarına dönüştürülebilirler.

Yaklaşık dizi eşleştirme yönteminde Şekil 2.5'te gösterildiği gibi ekleme, silme, değiştirme işlemleri sayesinde mesafe ölçüm değerleri belirlenmektedir. Yaklaşık dizi eşleştirme algoritmaları şu şekilde sınıflandırılabilir.

Filtreleme Tabanlı Algoritmalar: Bu algoritmalar iki aşamadan oluşmaktadırlar. İlk aşamada, metinde olası dizi oluşum yerleri tanımlanmıştır. İkinci aşamada, tüm bu konumlar tam olarak doğrulanmıştır. Bununla birlikte, bu algoritmalar en kötü durum senaryoları için yetersizdir (Kucherov et al., 2016).

Geri İzleme Tabanlı Algoritmalar: Bu algoritmalar genellikle kesin dizi eşleştirme algoritmalarının bir uzantısıdır. Bu yaklaşımda, mevcut kesin dizi eşleştirme algoritmaları, mesafe düzenleme işlemlerini kullanarak yaklaşık

aramayı sağlamak için geliştirilmişlerdir. Bu tür yaklaşımlarda sıkıştırılmış veri ve sonek tabanlı veri yapılarının kullanılmaktadır.



Şekil 2.5. Yaklaşık dizi eşleştirme örneği.

3. VERİ SETLERİ

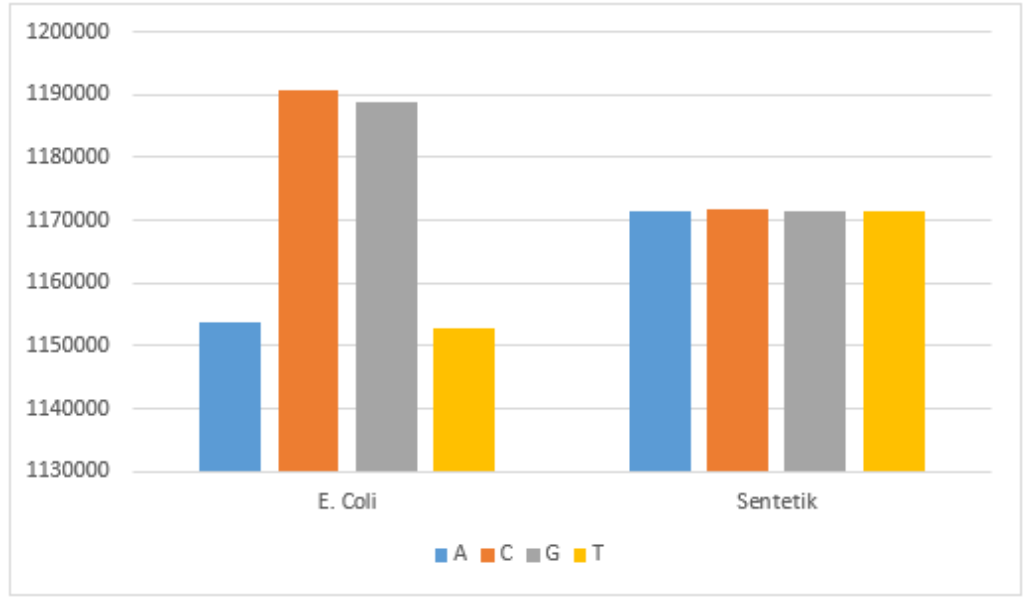
Deneysel çalışmalarda Çizelge 3.1'de gösterildiği gibi tez çalışmasında altı farklı veri seti kullanılmıştır. E. Coli (Durfee et al., 2008), Human Chromosome 1 (Gregory et al., 2004), Human Chromosome 2 (Hillier et al., 2005) ve Human Chromosome 3 (Muzny et al., 2006) veri setleri açık kaynak olarak biyoinformatik sekansları paylaşan NCBI sitesinden metin formatında indirilmiştir. Bu veri setleri, algoritmalar tarafından işlenmek üzere ön işleme tabi tutulmuştur. Bu ön işleme aşamasında alfabemizde $\{A, C, G, T\}$ olmayan 'N' gibi karakterler kaldırılmıştır. Ön işleme adımı, orjinal diziler korunduğu için biyoinformatik perspektik açısından veri setlerinin içeriği değiştirilmemiştir.

Human Chromosome 2 ve Human Chromosome 3 veri setlerini birleştirildi ve daha büyük bir veri seti elde etmek için Human Chromosome 2+3'ü oluşturduk. Çizelge 1'de gösterildiği gibi, E. Coli veri setinde 4 milyonun üzerinde baz bulunurken, Human Chromosome 1'de 243 milyondan fazla baz, Human Chromosome 2'de 248 milyondan fazla baz ve Human Chromosome 2+3 veri setlerinde 430 milyondan fazla baz bulunmaktadır. 4 milyondan fazla baza sahip olan sentetik veri seti bilgisayar tarafından rastgele üretilmiştir ve A, C, G, T bazlarının eşit dağılımına sahiptir. .

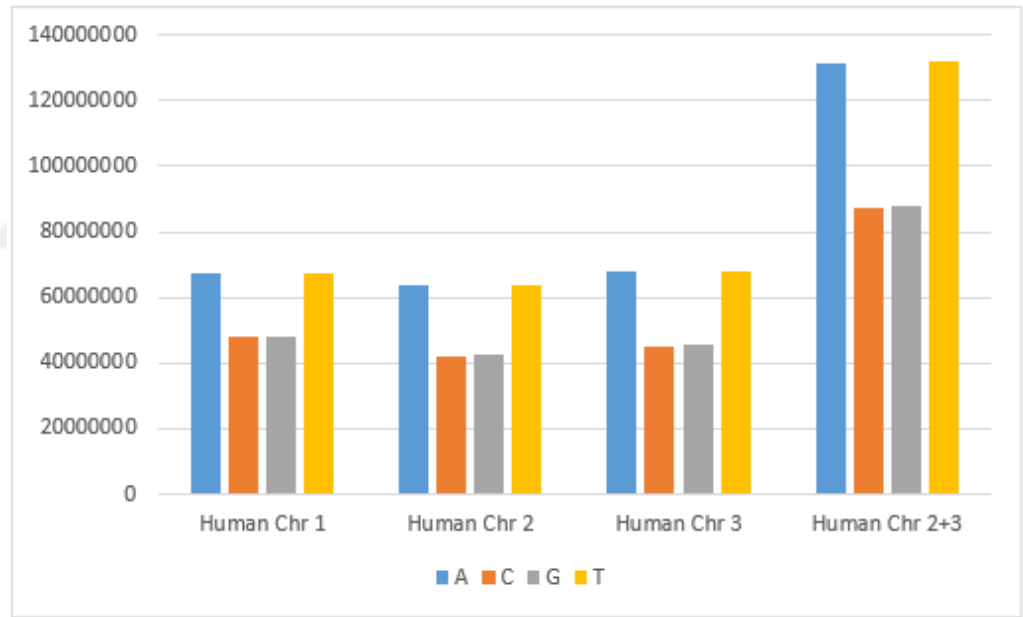
Çizelge 3.1. Tez çalışmasında kullanılan tüm veri setleri.

Veri kümesi	NCBI referans no	Uzunluk
E. Coli	NC_010473.1	4,753,183
Human Chromosome 1	NC_000001.11	248,956,422
Human Chromosome 2	NC_000002.12	243,199,373
Human Chromosome 3	NC_000003.12	198,295,559
Human Chromosome 2+3	-	441,494,932
Sentetik	-	4,753,208

Bu tez çalışması kapsamında kullanılan veri setlerinin DNA baz karakterlerinin frekans gösterimleri Şekil 3.1 ve 3.2'de gösterildiği gibidir.



Şekil 3.1. E. Coli ve Sentetik Veri Setlerinin DNA Baz Karakterlerinin Frekansı.



Şekil 3.2. Human Chromosome 1,2,3,2+3 Veri Setlerinin DNA Bazlarının Frekansı.

4. GELİŞTİRİLEN MODELLER

Tez çalışmasında DNA sekanslarında hash çakışmalarını ortadan kaldıran mükemmel hash fonksiyonu geliştirilmiş ve bu hash fonksiyonu kullanılarak hash tabanlı tekli ve çoklu kesin dizi eşleştirme algoritmalarına uygulanarak dört farklı çalışma gerçekleştirilmiştir. Bu çalışmalar aşağıda detaylı olarak anlatılmıştır.

4.1. DNA Sekansları İçin Mükemmel Hash Fonksiyonuna Dayalı Hash-q Kesin Dizi Eşleştirme Algoritmaları

Literatürdeki hash tabanlı algoritmalarındaki eksiklikleri göz önünde bulundurarak, DNA dizileri için hash çakışmalarını ortadan kaldırmak için tüm q değerlerinde eşsiz hash değerleri üreten yeni ve verimli bir hash fonksiyonu geliştirilmiştir. Önerilen hash fonksiyonunu kullanarak, daha verimli hash tabanlı algoritmalar önerilmiştir. Hash çakışmalarının ortadan kalkmasıyla çalışma zamanı, karakter karşılaştırma sayısı ve hash karşılaştırma sayısı gibi performans metriklerinde başarı elde edilmiştir. Önerilen hash fonksiyonunu kullanarak, Hash-q algoritmasına dayalı iki yaklaşım önerilmiştir. Motivasyon kısmında DNA sekanslarında hash çakışmalarını kaldıran mükemmel hash fonksiyonunun geliştirilmesine neden olan literatürdeki algoritmaların eksiklikleri ve bu eksikliklere yönelik geliştirilen hash fonksiyonundan bahsedilmiştir.

4.1.1. Motivasyon

Hash tabanlı kesin dizi eşleştirme algoritmalarından biri olan Boyer-Moore-Horspool- q (BMH- q) (Tarhio and Pentola, 1997) algoritmasıdır. Horspool algoritmasının etkili bir modifikasyonudur ve Adenin (A), Sitozin (C), Timin (T), Guanin (G) bazlarını içeren DNA dizileri için uygundur. Ancak bu algoritmanın dezavantajı, algoritmanın x86 mimarilerinin 32 bit okumaya izin veren mimari özelliklerini kullanarak yalnızca $q = 4$ için benzersiz hash değerleri üretmesidir (Tarhio and Pentola, 1997). Ayrıca, tüm q değerleri için benzersiz hash değerleri üretmez. Skip- q algoritmasında (Faro, 2016), hash değerlerinin aralığını belirleyen α parametresinin değeri 16 olarak ayarlanmıştır, yani tek bir 16 bitlik işlemcilerle çalışabilir. Ancak, Skip- q algoritmasında birçok hash çakışması vardır. Skip- q algoritmasının hash fonksiyonu Eşitlik 4.1'de gösterilmiştir.

$$v = \sum_{j=0}^{q-1} (x[i+j] \ll kj) \quad (4.1.)$$

Burada, x uzunluğu m olan bir dizidir. i , $0 \leq i \leq m - q$ olacak şekilde bir indis ve $kq \leq \alpha$ olacak şekilde k ve q iki tamsayıdır. DNA veri setlerinde oluşabilecek farklı hash indislerinin sayısı toplamda $\sum^q$ olur.

Çizelge 4.1. Skip-q algoritmasının farklı hash değerlerinin sayısı.

Skip-q algoritması	Skip-q'daki farklı hash indis sayısı	Maksimum eşsiz hash indis sayısı
Skip-2	16	16
Skip-3	63	64
Skip-4	243	256
Skip-5	415	1024
Skip-6	974	4096
Skip-7	2141	16384
Skip-8	4524	65536

Skip-q algoritmasının hash çakışmalarına örnek olarak Çizelge 4.1'de görüldüğü gibi Skip-3 algoritmasındaki farklı hash indislerinin sayısı 64 olmalıdır. Ancak farklı hash indislerinin sayısı 63'tür. Bunun nedeni Sitozin-Timin -Sitozin (CTC) ve Guanin-Sitosin-Guanine (GCG) dizileri, birbirlerinden farklı olmalarına rağmen $q = 3$ için aynı hash değeri üretir. Bu duruma hash çakışması denir. Skip-3'ün hash fonksiyonu $(x[i] \ll 4) + (x[i+1] \ll 2) + x[i+2]$ 'dir.

$$CTC = 16 * 67 + 4 * 84 + 67 = 1475$$

$$GCG = 16 * 71 + 4 * 67 + 71 = 1475$$

Skip-q algoritmasının hash fonksiyonu CTC ve GCG dizileri için $q = 3$ olduğunda aynı hash değerlerini üretir. Çizelge 4.1'de gösterildiği gibi q değeri arttıkça hash çakışma sayısı da artmaktadır. Kısacası, mevcut hash tabanlı algoritmalarındaki çakışma problemi olduğu gösterilmektedir.

Lecroq (2007), WM'nin (1994) çoklu dizi eşleştirme algoritmasını tekli dizi eşleştirme problemlerine uyarlayarak Hash-q algoritmasını önermiştir. Bu yaklaşımda üretilen hash değerleri 0 ile 255 arasındadır. Yani $0 \leq H \leq 255$.

Ayrıca, q değerleri $3 \leq q \leq 8$ aralığındadır. Hash-q'nun hash fonksiyonu $4x[i - 2] + 2x[i - 1] + x[i]$ 'dir ve aşağıda Eşitlik 4.2'de gösterilmiştir.

$$\text{kaydır}[h(\text{altdizi})] = \{ m - 1 - i \mid i = \text{maks}\{0 \leq j \leq m - q + 1 \mid h(x[j..j + q - 1]) = H \} \quad (4.2.)$$

$$m - q \quad i \text{ yoktur (ilk } q \text{ adet karakter) } \}$$

Hash-q algoritmasının farklı q değerleri için hash çakışması sayısı Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Hash-q algoritmasının farklı hash değerlerinin sayısı.

Hash-q algoritması	Hash-q'daki farklı hash indis sayısı	Maksimum eşsiz hash indis sayısı
Hash-3	53	64
Hash-4	156	256
Hash-5	231	1024
Hash-6	256	4096
Hash-7	256	16384
Hash-8	256	65536

Hash-q algoritmasındaki hash fonksiyonu 0-255 arasında bir değer üretir. Hash çakışmalarına örnek olarak Çizelge 4.2'de görüldüğü gibi Hash-3 algoritmasındaki farklı hash indekslerinin sayısı 64 olmalıdır. Ancak farklı hash indislerinin sayısı 53'tür ve farklı diziler aynı hash değerini üretir. Örneğin, Sitozin-Adenin-Guanin (CAG) ve Adenin-Guanin-Sitozin (AGC) için hash değerleri aşağıdaki gibidir:

$$CAG = 4 * 67 + 2 * 65 + 71 = 213$$

$$AGC = 4 * 65 + 2 * 71 + 67 = 213$$

CAG ve AGC farklı diziler olmasına rağmen, Hash-q algoritması aynı hash değeri üretir ve bu hash çakışmasına neden olur. Bu nedenle, Çizelge 4.1. ve Çizelge 4.2'de gösterildiği gibi Skip-q ve Hash-q algoritmaları için yapılan deneysel çalışmalar sonucu, farklı q değerleri için farklı sayıda hash çakışması olduğu anlaşılmıştır.

Yapılan bir çalışmada (Ho et al., 2018), A, C, G ve T harflerinin her birini 2-bitlik bir kod kullanarak tam sayıya dönüştürme önerilmiştir. Karakter sırasıyla 00, 01, 10, 11 olarak temsil edilmiştir. Ancak yaptıkları çalışmada bu dönüşüm süreci anlatılmamıştır.

Bu tez çalışmasında, literatürdeki hash tabanlı algoritmalarındaki eksiklikler dikkate alınarak, DNA sekanslarında hash çakışmalarını ortadan kaldırmak için, tüm q değerlerinde eşsiz hash değerleri üreten, yeni ve verimli bir hash fonksiyonunun geliştirilmesi hedeflenmiştir. Önerilen hash fonksiyonunu sayesinde daha verimli hash tabanlı algoritmalar geliştirilmiştir. Böylece, hash çakışmalarını kaldırılarak çalışma zamanı, karakter karşılaştırma sayısı ve hash karşılaştırma sayısı azalacaktır. Önerilen hash fonksiyonu kullanılarak, hash tabanlı kesin dizi eşleştirme algoritmaları geliştirilmiştir.

4.1.2. Önerilen Mükemmel Hash Fonksiyonu

DNA dizilerindeki hash çakışmalarını ortadan kaldırmak için verilen dizilerin bit gösterimleri ile hash değerlerini hesaplanmıştır. A, C, G, T için ASCII kodları şu şekilde temsil edilmiştir: A: 01000001, C:01000011, G: 01000111, T: 01010100

Arama süresinden kazanç elde etmek için, dizideki her karaktere değer atamak yerine DNA dizi harflerini eşsiz 2'li bitlerle temsil edebiliriz. Bu nedenle, bitlere bakarak, verilen DNA karakterlerini farklı bitlerle temsil etmek için ASCII kodlarından (sağdan 2. ve 3. bit) bir örüntü yakalanmıştır. DNA karakterleri için kullanılan ASCII kodlarının 2. ve 3. bitleri (sağdan) aşağıda gösterilmiştir:

A: 01000001 C: 01000011 G: 01000111 T: 01010100

Böylece A: 00, C: 01, G: 11, T: 10 karakterlerinin 2 bitlik karşılıkları kullanılırken farklı bitleri temsil ederek farklı hash değerlerinin üretilmesi sağlanmıştır. Önerilen mükemmel hash fonksiyonu (UniqueFNG) Eşitlik 4.3'te gösterilmiştir ve Algoritma 4.1'de sözde kodu verilmiştir.

$$UniqueFNG = (\sum_{i=0}^{q-2} (x[i] \& 0b110) \ll 2(q-i-1) - 1) + (x[q-1] \& 0b110) \gg 1 \quad (4.3.)$$

Önerilen hash fonksiyonun sözde kodu Algoritma 4.1’de gösterilmiştir.

Algoritma 4.1. Önerilen hash fonksiyonunun sözde kodu (UniqueFNG).

```

1. Algoritma   UniqueFNG( $x, i, q$ )
2.    $v \leftarrow 0$ 
3.   for  $i \leftarrow 0$  to  $q-2$  do
4.      $v \leftarrow v + ((x[i] \& 0b110) \ll 2(q-i-1) - 1)$ 
5.    $v \leftarrow v + (x[q-1] \& 0b110) \gg 1$ 
6.   return  $v$ 

```

Aşağıdaki örnek, Şekil 4.1'deki Hash-3 algoritması için verilen bir dizi için bir hash değerini nasıl bulunduğunu göstermektedir. CAG ve AGC dizilerinin hash değerleri, önerilen hash fonksiyonu ile sırasıyla 19 ve 13 olarak elde edilirken, Hash-q algoritmasında kullanılan hash fonksiyonunda bu dizilerin her ikisi için hash değerleri 213 elde edilir.

YeniHash-3 $\rightarrow ((x[0] \& 0b110) \ll 3) + ((x[0] \& 0b110) \ll 3) + ((x[0] \& 0b110) \ll 3)$

CAG: C = 01000011, A = 01000001, G = 01000111

C&0b110 = 00000010, A&0b110 = 00000000, G&0b110 = 00000110

$((C \& 0b110) \ll 3) + ((A \& 0b110) \ll 1) + ((G \& 0b110) \gg 1) = 00010000 + 00000000 + 00000011 = 16 + 0 + 3 = 19$

AGC: A = 01000001, G = 01000111, C = 01000011

A&0b110 = 00000000, G&0b110 = 00000110, C&0b110 = 00000010

$((A \& 0b110) \ll 3) + ((G \& 0b110) \ll 1) + ((C \& 0b110) \gg 1) = 00000000 + 00001100 + 00000001 = 0 + 12 + 1 = 13$

Şekil 4.1. Önerilen hash fonksiyonunun $q=3$ için hash değeri hesaplaması.

Çizelge 4.3’te gösterildiği gibi, önerilen hash fonksiyonu maksimum sayıda farklı hash değeri üretmektedir. Yani, herhangi bir hash çakışmasının meydana gelmediği gösterilmiştir.

Çizelge 4.3. Önerilen hash fonksiyonunun farklı hash değerlerinin sayısı.

q değeri	UniqueFNG'deki farklı hash indis sayısı	Maksimum eşsiz hash indis sayısı
2	16	16
3	64	64
4	256	256
5	1024	1024
6	4096	4096
7	16384	16384
8	65536	65536

4.1.3. UniqueFNG'li Hash-q Algoritması (HqUF)

HqUF algoritmasının ön işleme aşamasında, q uzunluğundaki tüm olası diziler için kaydırma değerleri hesaplanır. Ardından aranan dizi, arama aşamasında metnin başlangıcına hizalanır. Daha sonra hem metnin hem de aranan dizinin son q karakterinin hash değerleri hesaplanır. Hash değerleri eşit değilse, dizi $m - q + 1$ kadar sağa kaydırılır. Ancak hash değerleri birbirine eşitse, karakter karşılaştırması aranan dizinin en soldaki karakterinden $m - q$ konumundaki karaktere kadar yapılır. Bir dizinin tamamı eşleşirse, ön işleme adımında hesaplanan kaydırma değeri kadar aranan dizi metin üzerinde sağa kaydırılır. Bu işlem metnin sonuna kadar devam eder.

Hash-q algoritmasında $m - q + 1$ farklı alt dizi üretilebilir. Örneğin, $m = 8$ ve $q = 3$ olduğunda, $8-3+1=6$ farklı alt dizi vardır. Önerilen hash fonksiyonu ile, uzunluğu q olan her olası alt dizinin hash değeri $0 \leq c \leq 4^q$ arasında değerler alır. Kaydırma tablosundaki tüm değerler ilk olarak $m - q + 1$ birim kadar kaydırma değeri ile ilklendir. Bir q -uzunluklu alt dizinin kaydırma değeri, $m - 1 - i$ kadardır, burada i indeksi, q -uzunluklu alt dizinin son indis değerini temsil eder. Bu nedenle, her q -uzunluklu alt dizi için, hash değeri hesaplanır ve bu indis değerine karşılık gelen kaydırma değeri güncellenir. Elde edilen kaydırma değeri 0 (sıfır) ise 1 (bir) olarak ayarlanır. Bu durum, dizideki ardışık q -uzunluklu alt dizi eşit olduğunda gerçekleşir. Arama işleminde Hash-q algoritmasında olduğu gibi son q karakter için karakter karşılaştırması yapmamıza gerek yoktur. Çünkü her q -uzunluklu alt dizi için eşsiz hash değerleri elde edilmektedir. Böylece, dizi ve metindeki q -uzunluklu dizilerin her ikisinin de birbirine eşit olduğu anlamına gelir. HqUF'nin sözde kodu Algoritma 4.2'de gösterilmiştir.

Algoritma 4.2. HqUF algoritmasının sözde kodu ($q = 3$ için).

```

1.  $q = 3$ 
2.  $Hash\_Boyutu \leftarrow 4^q$ 
3.  $UniqueFNG(x, i, q)$ 
4.
5. Algoritma  $HqUF\_3(x, m, y, n)$ 
6. /* Ön İşleme Adımı */
7. for  $a \in Hash\_Boyutu$  do  $shift[a] \leftarrow m - 2$ 
8.  $h \leftarrow FNG(x, 0)$ 
9.  $shift[h] \leftarrow m - 3$ 
10. for  $i \leftarrow 3$  to  $m - 2$  do
11.    $h \leftarrow UniqueFNG(x, i - 2)$ 
12.    $shift[h] \leftarrow m - 1 - i$ 
13.  $h \leftarrow UniqueFNG(x, m - 3)$ 
14.  $sh1 \leftarrow shift[h], shift[h] \leftarrow 0$ 
15. if  $sh1=0$  then  $sh1 \leftarrow 1$ 
16. /* Arama Adımı */
17.  $y[n..n + m - 1] \leftarrow x, j \leftarrow m - 1$ 
18. while  $TRUE$  do
19.    $sh \leftarrow 1$ 
20.   while  $sh \neq 0$  do
21.      $h \leftarrow UniqueFNG(y, i - 2)$ 
22.      $sh \leftarrow shift[h]$ 
23.      $i \leftarrow i + sh$ 
24.   if  $i < n$  then
25.      $j \leftarrow 0$ 
26.     while  $(j < m - q \text{ and } x[j] == y[i - mMinus1 + j])$  do  $j++$ 
27.     if  $(j \geq m - q)$  then  $REPORT(i - m + 1)$ 
28.      $i \leftarrow i + sh1$ 
29. return

```

4.1.4. UniqueFNG'li Hash-q Boyer-Moore Algoritması (HqUF-BM)

Önceki bölümlerde, Hash-q algoritmasında dizi eşleştiğinde, $m - q + 1$ kadar dizinin kaydırıldığı gösterilmiştir. Bu bölümde, dizi eşleştiğinde Boyer-Moore algoritmasının iyi sonek kaydırma tablosunu HqUF'ye entegre edilmiş versiyonu olan HqUF-BM önerilmiştir. Böylece, iyi sonek kaydırma tablosundaki $(m - 1)$ 'inci girişinde belirtilen kaydırma miktarına göre metin üzerindeki dizi kaydırılır. Ayrıca, son q -uzunluklu alt dizi eşleştiğinde, dizinin ilk $m - q$ karakteri, en kötü durumda hizalandığında kendisine karşılık gelen metnin karakterleriyle karşılaştırılır. Çünkü önerilen hash fonksiyonu, q -uzunluklu diziler için eşsiz hash değerleri üretir. Bu, dizinin son q karakterinin hizalandığında kendisine karşılık gelen metin karakterleriyle eşleştiği anlamına gelir. Bu nedenle, başarılı tam eşleşme durumunda, q karakter karşılaştırmalarını da kaydedebiliriz. HqUF-BM'nin sözde kodu Algoritma 4.3'te gösterilmiştir.

Algoritma 4.3. HqUF-BM algoritmasının sözde kodu ($q = 3$ için).

```

1.  $q = 3$ 
2.  $QSIGMA \leftarrow 4^q$ 
3.  $UniqueFNG(x, i, q)$ 
4.
5. Algoritma  $sonekler(x, m, sonek[])$ 
6.    $sonek[m - 1] \leftarrow m$ 
7.    $g = m - 1$ 
8.   for  $i \leftarrow m - 2$  downto 0
9.     if  $i > g$  and  $sonek[i + m - 1 - f] < i - g$  then
10.        $sonek[i] \leftarrow sonek[i + m - 1 - f]$ 
11.     else
12.       if  $i < g$  then
13.          $g \leftarrow i$ 
14.          $f \leftarrow i$ 
15.         while  $g \geq 0$  and  $x[g] = x[g + m - 1 - f]$  do
16.            $g \leftarrow g - 1$ 
17.            $sonek[i] \leftarrow f - g$ 
18.
19. Algoritma  $preBmGs(x, m, bmGs[])$ 
20.    $sonek[m]$ 
21.    $sonekler(x, m, sonek)$ 
22.   for  $i \leftarrow 0$  to  $m - 1$  do
23.      $bmGs[i] \leftarrow m$ 
24.   for  $i \leftarrow m - 1$  down to 0 do
25.     if  $sonek[i] = i + 1$  then
26.       for  $j \leftarrow 0$  to  $m - 2 - i$  do
27.         if  $bmGs[j] = m$  then
28.            $bmGs[j] \leftarrow m - 1 - i$ 
29.   for  $i \leftarrow 0$  to  $m - 2$  do
30.      $bmGs[m - 1 - sonek[i]] \leftarrow m - 1 - i$ 
31.
32. Algoritma  $HqUF\_BM\_3(x, m, y, n)$ 
33.    $bmGs[m]$ 
34.   /* Ön İşleme Adımı */
35.    $preBmGs(x, m, bmGs)$ 
36.   for  $a \in QSIGMA$  do  $shift[a] \leftarrow m - 2$ 
37.    $h \leftarrow FNG(x, 0)$ 
38.    $shift[h] \leftarrow m - 3$ 
39.   for  $i \leftarrow 3$  to  $m - 2$  do
40.      $h \leftarrow UniqueFNG(x, i - 2)$ 
41.      $shift[h] \leftarrow m - 1 - i$ 
42.    $h \leftarrow UniqueFNG(x, m - 3)$ 
43.    $sh1 \leftarrow shift[h], shift[h] \leftarrow 0$ 
44.   if  $sh1 = 0$  then  $sh1 \leftarrow 1$ 
45.   /* Arama Adımı */
46.    $y[n..n + m - 1] \leftarrow x, j \leftarrow m - 1$ 
47.   while TRUE do
48.      $sh \leftarrow 1$ 
49.     while  $sh \neq 0$  do
50.        $h \leftarrow UniqueFNG(y, i - 2)$ 
51.        $sh \leftarrow shift[h]$ 
52.        $i \leftarrow i + sh$ 

```

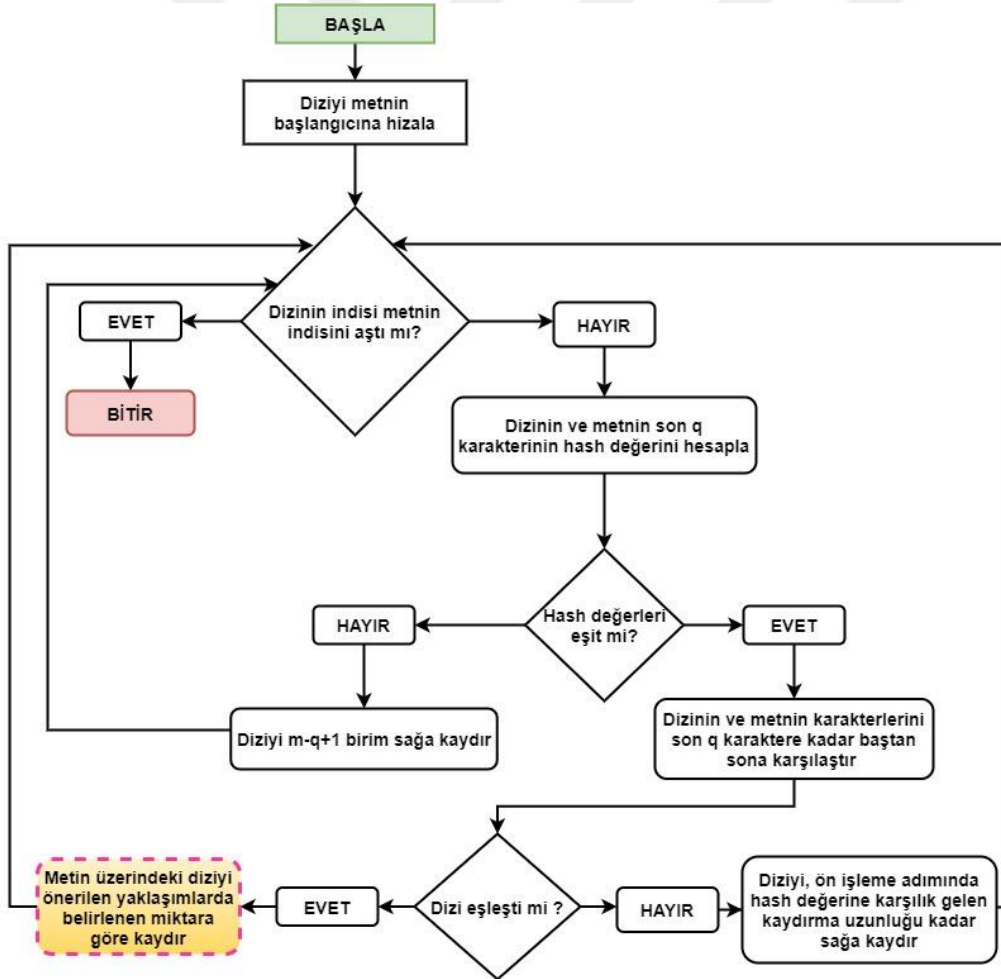
```

53.   if  $i < n$  then
54.        $j \leftarrow 0$ 
55.       while ( $j < m - q$  and  $x[j] = y[i - mMinus1 + j]$ ) do  $j++$ 
56.       if ( $j \geq m - q$ ) then
57.            $i \leftarrow i + bmGs[0]$ 
58.           REPORT ( $i - m + 1$ )
59.       else  $i \leftarrow i + sh1$ 
60.   return

```

4.1.5. Önerilen Yaklaşımların Genel Çerçevesi

Önerilen yaklaşımların genel çerçevesi, Şekil 4.2'de gösterilmiştir. Şekil 4.2'de, dizi önerilen yaklaşımlar tarafından belirlenen uzunluk kadar kaydırılır. HqUF algoritmasında, dizi metin üzerinde $m - q + 1$ birim kaydırılır. HqUF-BM algoritmasında, dizi, iyi sonek kaydırma tablosundaki $(m - 1)$ 'inci girişinde belirtilen uzunluk kadar kaydırılır.



Şekil 4.2. HqUF ve HqUF-BM yaklaşımların genel çerçevesi.

Çizelge 4.4, literatürde iyi bilinen tekli kesin dizi eşleştirme algoritmalarından KK, BM, BR, Skip-q, BMH-q ve Hash-q algoritmalarının ve

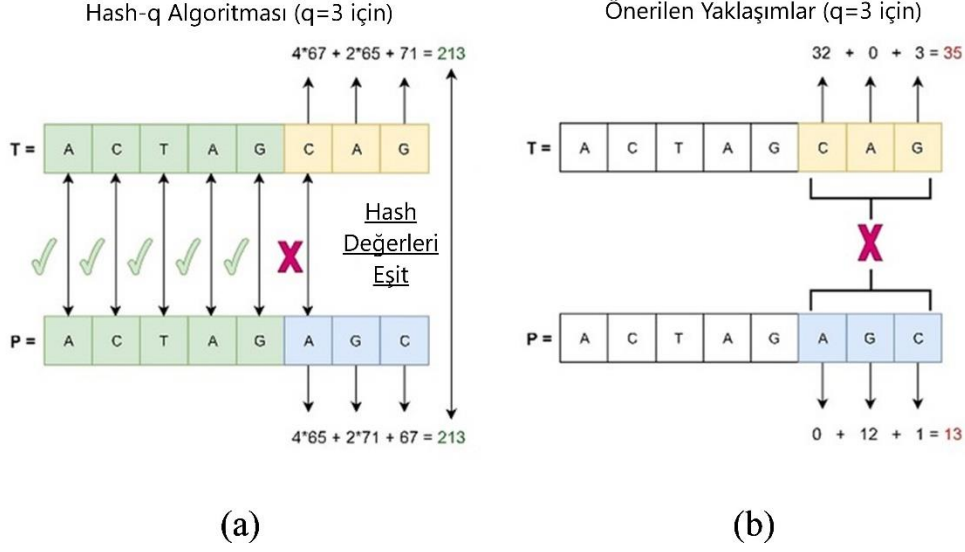
önerilen HqUF ve HqUF-BM algoritmalarının zaman karmaşıklıklarını göstermektedir. BF, BM, BR, Skip-q, BMH-q ve Hash-q algoritmalarının en kötü durum zaman karmaşıklıkları $O(nm)$ 'dir. Buna ek olarak, Hash-q algoritmasının ön işleme aşaması, q uzunluğundaki tüm olası diziler için bir kaydırma değeri hesaplar. Önerilen HqUF ve HqUF-BM algoritmalarının en kötü durum zaman karmaşıklıkları, son q karakteri karşılaştırılmadığından $O(n(m - q))$ 'dir. Ayrıca, Skip-q, BMH-q, Hash-q için en iyi durum zaman karmaşıklıkları gösterilmiştir ve HqUF ve HqUF-BM algoritmalarının en iyi durum zaman karmaşıklığı $O(n/(m - q))$ 'dir. Ancak, bu algoritmaların ortalama çalışma süresi, metnin ve aranan dizinin içeriğine bağlıdır.

Çizelge 4.4. Kesin dizi eşleştirme algoritmalarının zaman karmaşıklığı.

Algoritmalar	Zaman Karmaşıklığı	Veri Seti
KK	$O(nm)$	Tüm Metinler
BM	$O(nm)$	Tüm Metinler
BR	$O(nm)$	İngilizce Metinler
Skip-q	En İyi: $O(n/(m - q))$ En Kötü: $O(nm)$	DNA & Protein Sekansları, Doğal Dil Metinleri
BMH-q	$O(nm)$	DNA Sekansları
Hash-q	En İyi: $O(n/(m - q))$ En Kötü: $O(nm)$	DNA Sekansı, İngilizce Metinler
HqUF	En İyi: $O(n/(m - q))$ En Kötü: $O(n(m - q))$	DNA Sekansları
HqUF-BM	En İyi: $O(n/(m - q))$ En Kötü: $O(n(m - q))$	DNA Sekansları

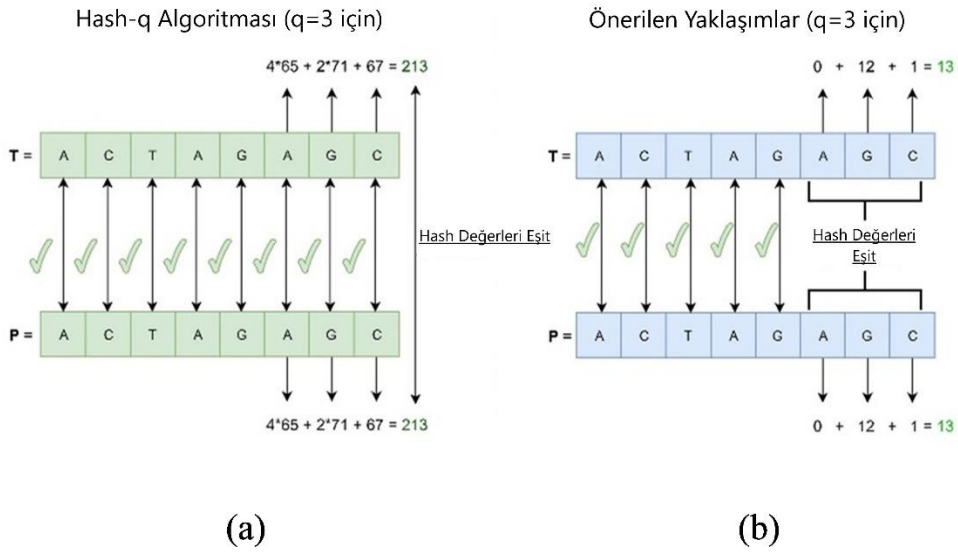
4.1.6. Önerilen Hash Fonksiyonunun Karakter Karşılaştırma Sayısı Üzerindeki Etkisine Örnekler

Şekil 4.3'te, CAG ve AGC dizileri birbirinden farklı olmasına rağmen, Hash-q algoritması $q = 3$ için aynı hash değerlerini üretir. Sonra, dizinin başından itibaren karakterleri karşılaştırmaya başlar. Örnekte görüldüğü gibi 6. karakter eşleşmemiştir. Yani, eşleşmeye olana kadar altı karakter karşılaştırması yapılmıştır. Ancak önerilen hash fonksiyonu kullanıldığında, her benzersiz dizi için farklı hash değerleri üretildiğinden son q karakter karşılaştırması yapılmaz. Böylece çok sayıda karakter karşılaştırmasından kazanç elde etmiş oluruz.



Şekil 4.3. Eşleşme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karakter karşılaştırma örneği (q=3 için).

Şekil 4.4'te başarılı bir eşleşme örnek olarak gösterilmiştir. Hash-q algoritmasında, kullanılan hash fonksiyonu çakışmalara neden olacağından, her durumda dizinin son q karakteri karşılaştırılır. Bununla birlikte, önerilen hash fonksiyonu, her benzersiz q -uzunluklu alt dizi için eşsiz hash değerleri ürettiğinden, önerilen yaklaşımlarda son q -karakter karşılaştırmaya gerek yoktur. Bu nedenle, önerilen yaklaşımlar daha az karakter karşılaştırması gerçekleştirmiştir.



Şekil 4.4. Eşleşme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karakter karşılaştırma örneği (q=3 için).

4.1.7. Önerilen Yaklaşımlar İçin Çalışan Örnekler

Hash-q, HqUF ve HqUF-BM algoritmaları için ($q = 3$) çalışılan örnekler Şekil 4.5, 4.6, 4.7'de gösterilmiştir. Bu bölümde metin ve aranan dizi sırasıyla *agtcagatacgagaccattacgagattacagattacgagattcagtcgaga* ve *acgagatt* olarak seçilmiştir. Her adımda, hash karşılaştırma sayısı, karakter karşılaştırma sayısı ve bulunan toplam dizi sayısı gösterilmiştir. Başarısız hash ve karakter karşılaştırmaları açık kırmızı renkte gösterilmiştir. Başarılı hash karşılaştırmaları yeşil renkte, başarılı karakter karşılaştırmaları ise açık yeşil renkte gösterilmiştir.

Hash-q algoritmasının $q = 3$ için çalışan örneği Şekil 4.5'te gösterilmiştir. Hash karşılaştırma sayısı 10, karakter karşılaştırma sayısı 17 ve eşleşen dizi sayısı 2 olarak elde edilmiştir.

Adım 1:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 1 Kaydır 1 hash karşılaştırması
Adım 2:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash karşılaştırması
Adım 3:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash karşılaştırması
Adım 4:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash karşılaştırması
Adım 5:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash, 8 karakter karşılaştırması
Adım 6:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 1 Kaydır 1 hash karşılaştırması
Adım 7:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash, 1 karakter karşılaştırması
Adım 8:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 2 Kaydır 1 hash karşılaştırması
Adım 9:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash, 8 karakter karşılaştırması
Adım 10:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydır 1 hash karşılaştırması
Adım 11:	agtcagatacgagaccattacgagattacagattacgagattcagtcgaga metnin boyutunu aştı acgagattt

Sonuç: hash karşılaştırma sayısı = 10, karakter karşılaştırma sayısı = 17, eşleşen dizi sayısı = 2

Şekil 4.5. Hash-q algoritmasının $q=3$ için çalışan örneği.

HqUF algoritmasının $q = 3$ için çalışan örneği Şekil 4.6'da gösterilmiştir. Hash karşılaştırma sayısı 10, karakter karşılaştırma sayısı 11 ve eşleşen dizi sayısı 2 olarak elde edilmiştir.

Adım 1:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 1 Kaydır 1 hash karşılaştırması
Adım 2:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash karşılaştırması
Adım 3:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash karşılaştırması
Adım 4:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash karşılaştırması
Adım 5:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash, 5 karakter karşılaştırması
Adım 6:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 1 Kaydır 1 hash karşılaştırması
Adım 7:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash, 1 karakter karşılaştırması
Adım 8:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 2 Kaydır 1 hash karşılaştırması
Adım 9:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash, 5 karakter karşılaştırması
Adım 10:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a a c g a g a t t → 6 Kaydır 1 hash karşılaştırması
Adım 11:	a g t c a g a t a c g a g a c c a t t a c g a g a t t a c a g a t t a c g a g a t t c a g t c g a g a metnin boyutunu aştı a c g a g a t t
Sonuç:	hash karşılaştırma sayısı = 10, karakter karşılaştırma sayısı = 11, eşleşen dizi sayısı = 2

Şekil 4.6. HqUF algoritmasının $q=3$ için çalışan örneği.

HqUF-BM algoritmasının $q = 3$ için çalışan örneği Şekil 4.7'da gösterilmiştir. Hash karşılaştırma sayısı 9, karakter karşılaştırma sayısı 10 ve eşleşen dizi sayısı 2 olarak elde edilmiştir.

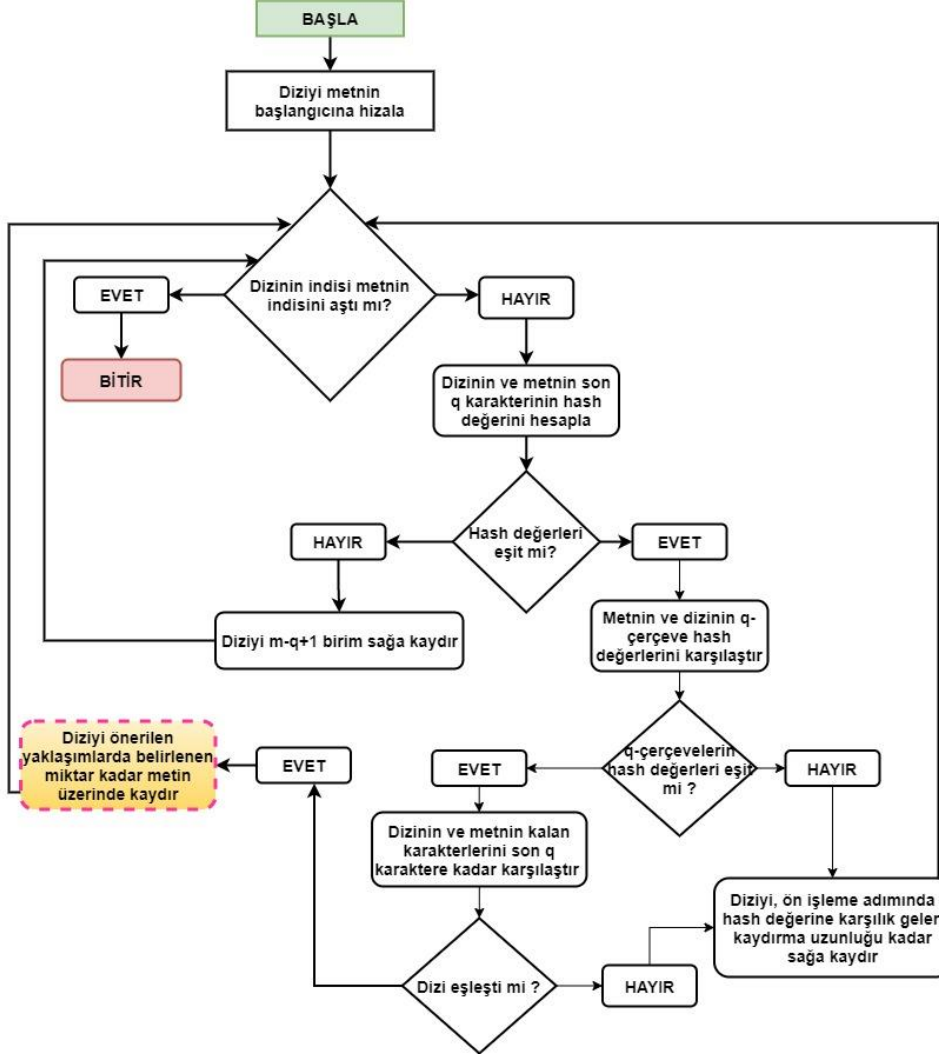
Adım 1:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 1 Kaydırma 1 hash karşılaştırması
Adım 2:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydırma 1 hash karşılaştırması
Adım 3:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydırma 1 hash karşılaştırması
Adım 4:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydırma 1 hash karşılaştırması
Adım 5:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 8 Kaydırma 1 hash, 5 karakter karşılaştırması
Adım 6:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 6 Kaydırma 1 hash karşılaştırması
Adım 7:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 1 Kaydırma 1 hash karşılaştırması
Adım 8:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 8 Kaydırma 1 hash, 5 karakter karşılaştırması
Adım 9:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga acgagattt → 3 Kaydırma 1 hash karşılaştırması
Adım 10:	agtcagat acgagaccattacgagattacagattacgagattcagtcgaga metnin boyutunu aştı acgagattt
Sonuç:	hash karşılaştırma sayısı = 9, karakter karşılaştırma sayısı = 10, eşleşen dizi sayısı = 2

Şekil 4.7. HqUF-BM algoritmasının $q=3$ için çalışan örneği.

Bu örnek için önerilen yaklaşımların hash ve karakter karşılaştırma sayıları Hash- q algoritmasından daha azdır. Bu durum, HqUF-BM algoritmasının, hash sayısı ve karakter karşılaştırma sayıları açısından HqUF algoritmasından daha iyi performans elde edebileceğini göstermiştir.

4.2. DNA Sekansları İçin q -Çerçeve Hash Karşılaştırmasına Dayalı Kesin Dizi Eşleştirme Algoritmaları

DNA sekanslarında tüm q değerleri için eşsiz hash değerleri üreten önerilen hash fonksiyonunu kullanarak bu çalışmada q -çerçeve hash karşılaştırmasına dayalı iki algoritma daha önerilmiştir. Önerilen algoritmalar, ortalama çalışma süresini ve karakter karşılaştırma sayısını azaltırken, trade-off olarak hash karşılaştırma sayısında artış meydana gelmektedir. Önerilen yaklaşımların genel çerçevesi Şekil 4.8'de gösterilmiştir. Tez çalışmasının bu kısmında iki farklı kesin dizi eşleştirme algoritması önerilmiştir ve bunlar i) Eşsiz q -çerçeve hash karşılaştırmalı Hash- q algoritması (Hq-QF) ve ii) Eşsiz q -çerçeve hash karşılaştırmalı Hash- q Boyer-Moore algoritması (HqBM-QF). Her iki algoritma da sırasıyla HqUF ve HqUF-BM algoritmalarının geliştirilmiş versiyonudur. Önerilen algoritmaların her ikisi de karakter karşılaştırma sayısını azaltmak için q -çerçeve hash karşılaştırmalarını içerir.



Şekil 4.8. Önerilen Hq-QF ve HqBM-QF yaklaşımların genel çerçevesi.

4.2.1. Eşsiz q -Çerçeve Hash Karşılaştırmalı Hash- q Algoritması (Hq-QF)

HqUF algoritması, dizinin ve metnin son q karakterlerinin hash değerlerini karşılaştırır. Hash değerleri eşleşirse, dizinin ilk karakterinden son q karakterine kadar karakter karşılaştırması yapar. Ancak, hash fonksiyonu eşsiz değerler ürettiği için, karakterleri karşılaştırmak yerine diğer alt diziler için de hash karşılaştırmaları yapılabilir. Bu nedenle, karakter karşılaştırma sayısını azaltmak için Eşsiz q -çerçeve hash karşılaştırmalı Hash- q (Hq-QF) algoritması önerilmiştir. Belirli bir m uzunluklu dizi için q -çerçeve sayısı $\lfloor m/q \rfloor$ 'dir.

m , q 'nun katı değilse, geriye kalan karşılaştırılacak karakter sayısı $m - (\lfloor m/q \rfloor * q)$ adettir. Bu algoritma, dizi ile metnin q -çerçevelerinin hash değerlerini soldan sağa doğru karşılaştırır. Dizinin son q karakterinin hash

değeri eşleşmezse, dizi, metnin son q karakterinin hash değerine karşılık gelen kaydırma miktarı kadar kaydırılır. Dizinin son q karakterinin hash değeri eşleşirse, q -çerçevesinin hash değeri hesaplanır ve tüm $\lfloor m/q \rfloor - 1$ adet q -çerçeveleri için hash karşılaştırması yapılır. Tüm q -çerçeve hash değerleri başarılı bir eşleşmeyle sonuçlanırsa ve herhangi bir karakter kalırsa, kalan karakterler soldan sağa doğru karşılaştırılır. Herhangi bir ara hash karşılaştırması veya karakter karşılaştırması eşleşmezse, model $m - q + 1$ miktarı kadar kaydırılır. Dizi başarılı bir şekilde eşleşirse, $m - q + 1$ miktarı kadar kaydırılır. Hq-QF algoritmasının sözde kodu Algoritma 4.4'te verilmiştir.

Algoritma 4.4: Hq-QF algoritmasının sözde kodu ($q = 3$ için).

```

1.   $q \leftarrow 3$ 
2.   $QSIGMA \leftarrow 4^q$ 
3.  Algoritma FNG( $x, i, q$ )
4.  Algoritma Hq_QF_3( $x, m, y, n$ )
5.  → Ön İşleme Adımı
6.    for  $a \in QSIGMA$  do  $shift[a] \leftarrow m - 2$ 
7.     $h \leftarrow FNG(x, 0)$ 
8.     $shift[h] \leftarrow m - 3$ 
9.    for  $i \leftarrow 3$  to  $m - 2$  do
10.      $h \leftarrow FNG(x, i - 2)$ 
11.      $shift[h] \leftarrow m - 1 - i$ 
12.     $h \leftarrow FNG(x, m - 3)$ 
13.     $sh1 \leftarrow shift[h], shift[h] \leftarrow 0$ 
14.     $maks\_eslesme\_say \leftarrow \lfloor m/q \rfloor$ 
15.  → Arama Adımı
16.     $y[n..n + m - 1] \leftarrow x, j \leftarrow m - 1$ 
17.    while  $TRUE$  do
18.       $sh \leftarrow 1$ 
19.      while  $sh \neq 0$  do
20.         $h \leftarrow FNG(y, i - 2)$ 
21.         $sh \leftarrow shift[h]$ 
22.         $i \leftarrow i + sh$ 
23.      if  $i < n$  then
24.         $eslesen\_say \leftarrow 1$ 
25.        while  $m - eslesen\_say * RANK3 \geq RANK3$  and  $(HASH(y, i - mMinus1 + eslesen\_say * RANK3 - 3) = HASH(x, m - mMinus1 + eslesen\_say * RANK3 - 4))$  do  $eslesen\_say \leftarrow eslesen\_say + 1$ 
26.        if  $maks\_eslesme\_say = eslesen\_say$  then
27.           $j \leftarrow 0$ 
28.          while  $j < m - eslesen\_say * RANK3$  and  $x[j + (eslesen\_say - 1) * RANK3] = y[i - mMinus1 + j + (eslesen\_say - 1) * RANK3]$  do  $j \leftarrow j + 1$ 
29.          if  $j \geq m - eslesen\_say * q$  then REPORT ( $i - m + 1$ )
30.           $i \leftarrow i + sh1$ 
31.  return

```

4.2.2. Eşsiz q -Çerçeve Hash Karşılaştırmalı Hash- q Boyer-Moore Algoritması (HqBM-QF)

Önerilen Hq-QF algoritmasında, bir dizi eşleşmesi meydana geldiğinde, dizi metin üzerinde $m - q + 1$ sabit uzunluk kadar kaydırılır. Ancak HqUF ve Hq-QF algoritmalarında eşleşen son eke ilişkin bilgi kaydırma değerine dahil edilmez. HqUF-BM, eşleşen son ekin kaydırma değerinden oluşur. Bu nedenle, HqUF-BM algoritmasını geliştirmek için, Eşsiz q -çerçeve hash karşılaştırmalı Hash- q Boyer-Moore (HqBM-QF) algoritması önerilmiştir. Kaydırma değerini elde etmek için q -çerçeve hash karşılaştırmalarını arama süreci ve iyi sonek bilgileriyle birleştiririz. HqBM-QF algoritmasında dizi eşleşirse iyi sonek tablosunun $(m - 1)$ 'inci girişindeki değer kaydırma değeri olarak seçilir. HqBM-QF algoritmasının diğer adımları Hq-QF algoritmasınıninkilerle aynıdır. HqBM-QF algoritmasının sözde kodu Algoritma 4.5'te gösterilmiştir.

Algoritma 4.5. HqBM-QF algoritmasının sözde kodu ($q = 3$ için).

```

1.  $q \leftarrow 3$ 
2.  $QSIGMA \leftarrow 4^q$ 
3. Algoritma  $FNG(x, i, q)$ 
4. Algoritma  $sonekler(x, m, sonek[])$ 
5.  $sonek[m - 1] \leftarrow m$ 
6.  $g = m - 1$ 
7. for  $i \leftarrow m - 2$  down to 0
8.     if  $i > g$  and  $sonek[i + m - 1 - f] < i - g$  then
9.          $sonek[i] \leftarrow sonek[i + m - 1 - f]$ 
10.    else
11.        if  $i < g$  then
12.             $g \leftarrow i$ 
13.             $f \leftarrow i$ 
14.            while  $g \geq 0$  and  $x[g] = x[g + m - 1 - f]$  do
15.                 $g \leftarrow g - 1$ 
16.             $sonek[i] \leftarrow f - g$ 
17.
18. Algoritma  $preBmGs(x, m, bmGs[])$ 
19.  $sonek[m]$ 
20.  $sonekler(x, m, sonek)$ 
21. for  $i \leftarrow 0$  to  $m - 1$  do
22.     $bmGs[i] \leftarrow m$ 
23. for  $i \leftarrow m - 1$  down to 0 do
24.    if  $sonek[i] = i + 1$  then
25.        for  $j \leftarrow 0$  to  $m - 2 - i$  do
26.            if  $bmGs[j] = m$  then
27.                 $bmGs[j] \leftarrow m - 1 - i$ 
28. for  $i \leftarrow 0$  to  $m - 2$  do
29.     $bmGs[m - 1 - sonek[i]] \leftarrow m - 1 - i$ 
30.
31. Algoritma  $HqBM\_QF\_3(x, m, y, n)$ 

```

```

32.  $bmGs[m]$ 
33. → Ön İşleme Adımı
34.  $preBmGs(x, m, bmGs)$ 
35. for  $a \in QSIGMA$  do  $shift[a] \leftarrow m - 2$ 
36.  $h \leftarrow FNG(x, 0)$ 
37.  $shift[h] \leftarrow m - 3$ 
38. for  $i \leftarrow 3$  to  $m - 2$  do
39.      $h \leftarrow FNG(x, i - 2)$ 
40.      $shift[h] \leftarrow m - 1 - i$ 
41.  $h \leftarrow FNG(x, m - 3)$ 
42.  $sh1 \leftarrow shift[h], shift[h] \leftarrow 0$ 
43.  $maks\_eslesme\_say \leftarrow \lfloor m/q \rfloor$ 
44. → Arama Adımı
45.  $y[n..n + m - 1] \leftarrow x, j \leftarrow m - 1$ 
46. while  $TRUE$  do
47.      $sh \leftarrow 1$ 
48.     while  $sh \neq 0$  do
49.          $h \leftarrow FNG(y, i - 2)$ 
50.          $sh \leftarrow shift[h]$ 
51.          $i \leftarrow i + sh$ 
52. if  $i < n$  then
53.      $eslesen\_say \leftarrow 1$ 
54.     while  $m - eslesen\_say * RANK3 \geq RANK3$  and  $(HASH(y, i - mMinus1 + eslesen\_say * RANK3 - 3) = HASH(x, m - mMinus1 + eslesen\_say * RANK3 - 4))$  do  $eslesen\_say \leftarrow eslesen\_say + 1$ 
55.     if  $maks\_eslesme\_say = eslesen\_say$  then
56.          $j \leftarrow 0$ 
57.         while  $j < m - eslesen\_say * RANK3$  and  $x[j + (eslesen\_say - 1) * RANK3] = y[i - mMinus1 + j + (eslesen\_say - 1) * RANK3]$  do  $j \leftarrow j + 1$ 
58.         if  $j \geq m - eslesen\_say * q$  then
59.             REPORT  $(i - m + 1)$ 
60.              $i \leftarrow i + bmGs[0]$ 
61.         else  $i \leftarrow i + sh1$ 
62.     else  $i \leftarrow i + sh1$ 
63. return

```

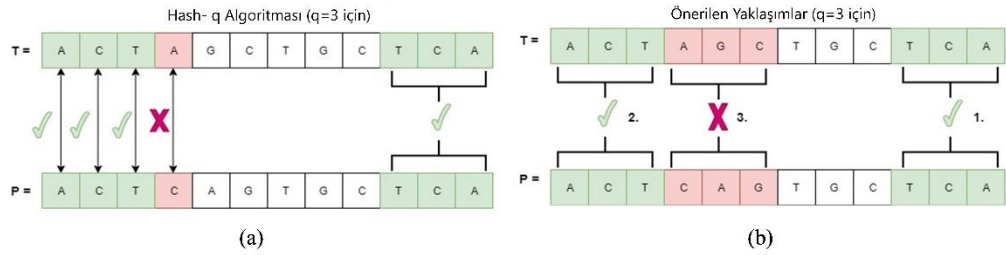
KK, BM, BR, Skip-q, BMH-q, Hash-q, HqUF, HqUF-BM algoritmalarının ve önerilen Hq-QF ve HqBM-QF algoritmalarının zaman karmaşıklığı analizi Çizelge 4.5'te gösterilmiştir. BF, BM, BR, Skip-q, BMH-q ve Hash-q algoritmalarının zaman karmaşıklığı $O(nm)$ 'dir. Ayrıca, HqUF ve HqUF-BM algoritmalarının en kötü durum zaman karmaşıklığı $O(n(m - q))$ 'dir. Önerilen Hq-QF ve HqBM-QF algoritmalarının en kötü durum zaman karmaşıklığı $O(n(m - (\lfloor m/q \rfloor q)))$. HqUF ve HqUF-BM algoritmalarında karakter karşılaştırma sayısı en aza indirildiğinden, zaman karmaşıklığı açısından daha verimli Hq-QF ve HqBM-QF algoritmaları önerilmiştir. Önerilen algoritmalarda karakter karşılaştırma sayısı $m - (\lfloor m/q \rfloor * q)$ olarak hesaplanmıştır.

Çizelge 4.5. İyi bilinen kesin dizi eşleştirme algoritmalarının zaman karmaşıklık analizi.

Algoritmalar	Zaman Karmaşıklığı	Veri Seti
KK	$O(nm)$	Tüm Metinler
BM	$O(nm)$	Tüm Metinler
BR	$O(nm)$	İngilizce Metinler
Skip-q	En İyi: $O(n/(m - q))$ En Kötü: $O(nm)$	DNA & Protein Sekansı, Doğal Dil Metinleri
BMH-q	$O(nm)$	DNA Sekansı
Hash-q	En İyi: $O(n/(m - q))$ En Kötü: $O(nm)$	DNA Sekansı, İngilizce Metinler
HqUF	En İyi: $O(n/(m - q))$ En Kötü: $O(n(m - q))$	DNA Sekansı
HqUF-BM	En İyi: $O(n/(m - q))$ En Kötü: $O(n(m - q))$	DNA Sekansı
Hq-QF	En İyi: $O\left(\frac{n}{(m - (\lfloor \frac{m}{q} \rfloor q))}\right)$ En Kötü: $O\left(n\left(m - \left(\lfloor \frac{m}{q} \rfloor q\right)\right)\right)$	DNA Sekansı
HqBM-QF	En İyi: $O\left(\frac{n}{(m - (\lfloor \frac{m}{q} \rfloor q))}\right)$ En Kötü: $O\left(n\left(m - \left(\lfloor \frac{m}{q} \rfloor q\right)\right)\right)$	DNA Sekansı

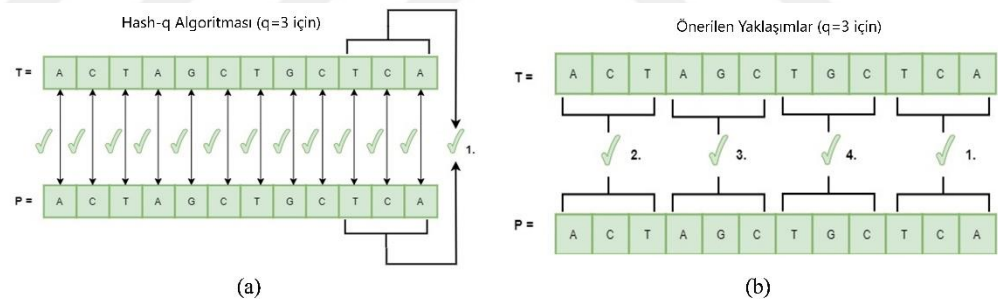
4.2.3. Karakter Sayısı ve Hash Karşılaştırmalarına İlişkin Önerilen Yaklaşımlara Örnekler

Bu bölümde algoritmalar arasındaki farklara bir örnek verilmiştir. Şekil 4.9'daki örnek için Hash-q algoritmasında, dizinin ve metnin son q karakterinin hash değerleri eşleşirse, karakterler dizinin başından sonuna kadar sırayla karşılaştırılır. Böylece, bir eşleşme olana kadar Şekil 4.9a'da gösterildiği gibi bir hash karşılaştırması ve dört karakter karşılaştırması gerçekleştirilir ve ardından dizi kaydırılır. Önerilen yaklaşımlarda, dizinin ve metnin son q karakterinin hash değerleri aynı ise, baştan sona doğru karakter karşılaştırması yapmak yerine, dizinin başlangıcından dizinin son q karakterine kadar olan q -çerçevelerin hash değerleri karşılaştırılır. Önerilen yaklaşımlarda q -çerçeve hash karşılaştırmasında eşleşme durumu Şekil 4.9b'de gösterilmiştir. Bu durumda önerilen yaklaşımlar bir hash ve dört karakter karşılaştırması yerine üç hash karşılaştırması yapar.



Şekil 4.9. Eşleşmeme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karşılaştırma örneği (q=3 için).

Başarılı bir dizi eşleşmesi için, Hash-q algoritması Şekil 4.10a'da gösterildiği gibi bir hash ve 12 karakter karşılaştırması yapar. Son q karakterin hash değeri eşleşse bile, algoritmanın bu karakterlerin hash çakışmaları nedeniyle birbiriyle uyuşup uyuşmadığını belirlemesi gerekir. Ancak, önerilen yaklaşımların bu karşılaştırmaları yapmasına gerek yoktur, çünkü hash fonksiyonu mükemmel hash değerleri üretir. Şekil 4.10b'de verilen örnek için önerilen yaklaşımlarda sadece dört hash karşılaştırma yapılmıştır. Sonuç olarak, önerilen yaklaşımlarda karakter karşılaştırması en aza indirilmiştir ve hash karşılaştırmasında olası bir artış olmuştur.



Şekil 4.10. Eşleşme durumu için a) geleneksel Hash-q Algoritmasının ve önerilen yaklaşımların karşılaştırma örneği (q=3 için).

4.2.4. Önerilen Yaklaşımlar İçin Çalışan Örnekler

Hash-q, Hq-QF ve HqBM-QF algoritmalarının çalışan örnekleri sırasıyla Şekil 4.11, 4.12 ve 4.13'te $q = 3$ için verilmiştir. Her adımda, hash karşılaştırma sayısı, karakter karşılaştırma sayısı ve dizi eşleşme sayıları gösterilmiştir. Eşleşmeyen hash değerleri ve karakter karşılaştırmaları açık kırmızı renkte gösterilmiştir. Başarılı hash karşılaştırmaları yeşil, başarılı karakter karşılaştırmaları ise açık yeşil ile gösterilmiştir. Önerilen yaklaşımlardaki hash karşılaştırmaları, q -çerçeve hash karşılaştırmaları

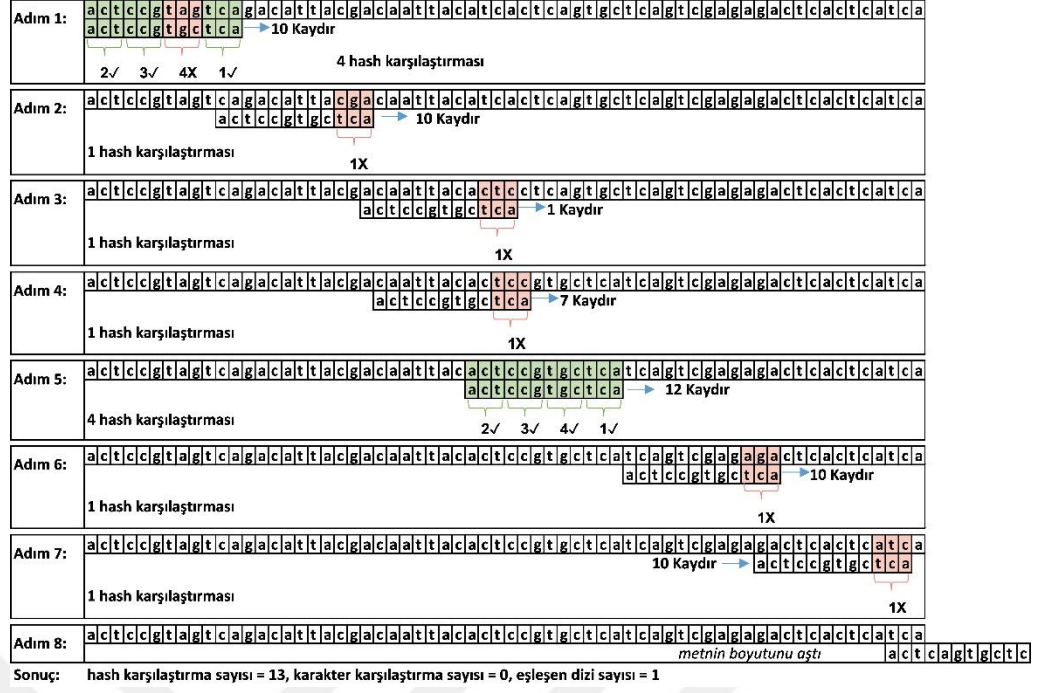
sırasıyla numaralandırılmıştır. Çalışılan örneklerin sonuçları her şeklin sonunda verilmiştir.

Adım 1:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1✓ 1 hash, 8 karakter karşılaştırması
Adım 2:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash karşılaştırması 1X
Adım 3:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 1 Kaydır 1 hash karşılaştırması 1✓
Adım 4:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 7 Kaydır 1 hash karşılaştırması 1X
Adım 5:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash, 12 karakter karşılaştırması 1✓
Adım 6:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash karşılaştırması 1X
Adım 7:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash, 1 karakter karşılaştırması 1✓
Adım 8:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a metnin boyutunu aştı a c t c a g t g c t c a
Sonuç:	hash karşılaştırma sayısı = 7, karakter karşılaştırma sayısı = 21, eşleşen dizi sayısı = 1

Şekil 4.11. Hash-q algoritmasının çalışan örneği (q=3 için).

Adım 1:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 2✓ 3✓ 4X 1✓ 4 hash karşılaştırması
Adım 2:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash karşılaştırması 1X
Adım 3:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 1 Kaydır 1 hash karşılaştırması 1✓
Adım 4:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 7 Kaydır 1 hash karşılaştırması 1X
Adım 5:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 4 hash karşılaştırması 2✓ 3✓ 4✓ 1✓
Adım 6:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 1 hash karşılaştırması 1X
Adım 7:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a → 10 Kaydır 2 hash karşılaştırması 2X 1✓
Adım 8:	a c t c c g t a g t c a g a c a t t a c g a c a a t t a c a c t c a g t g c t c a g t c g a g a g a c t c a c t c a t c a a c t c c g t a g t c a metnin boyutunu aştı a c t c a g t g c t c a
Sonuç:	hash karşılaştırma sayısı = 14, karakter karşılaştırma sayısı = 0, eşleşen dizi sayısı = 1

Şekil 4.12. Hq-QF algoritmasının çalışan örneği (q=3 için).



Şekil 4.13. HqBM-QF algoritmasının çalışan örneği ($q=3$ için).

Sonuç olarak, çalışılan örnekler için önerilen yaklaşımlarda 0 adet karakter karşılaştırması yapılırken, Hash-q algoritması 21 karakter karşılaştırması yapmıştır. Ancak, beklendiği gibi hash karşılaştırma sayısında bir artış vardır.

4.3. DNA Sekansları İçin WM-q Çoklu Kesin Dizi Eşleştirme Algoritması

4.3.1. WM Algoritması

WM algoritması çoklu kesin dizi eşleştirme problemini çözmek için kötü karakter atlama tablosunu kullanan BM algoritmasının bir versiyonudur. WM algoritmasının amacı tüm karakterleri tek tek okumak yerine B uzunluklu bloklar halinde okumaktır. Wu ve Manber (1994), B blok uzunluğunu yapmış oldukları deneysel çalışmalar sonucu 2 olarak seçmiştir. WM algoritması ön işleme ve arama olmak üzere iki aşamadan oluşmaktadır.

WM algoritmasının ön işleme aşamasında üç farklı tablo oluşturulur. Bunlar ATLAMA, ÖNEK ve HASH tablolarıdır. Tüm dizilerin B uzunluklu atlama değerleri ATLAMA tablosunda temsil edilir. Eğer B uzunluklu farklı karakter dizisi aranıyorsa $m - B + 1$ birim kadar atlama değeri atanır. HASH tablosu son B uzunluklu karakter olarak temsil edilen soneklerin hash değerlerini tutar. Hash değerleri karakterlerin ASCII değerleri kullanılarak belli

bir hash fonksiyonundan geçirildikten sonra elde edilen hash değeri (h) ile temsil edilir. WM algoritmasında kullanılan hash fonksiyonu $h = ((T[m - 2] \ll 8 + T[m - 1]) \% 65536)$ şeklindedir. ÖNEK tablosu ise aynı soneke sahip olan dizilerin listesini tutar. B' uzunluklu örneklerin hash değerleri (h') ile ÖNEK tablosunda temsil edilirler. WM algoritması İngilizcede 'ion, ing' gibi sonu aynı biten çok sayıda kelimenin tek tek karşılaştırma maliyetini en aza indirmek için ÖNEK tablosu oluşturulmuştur. Sonekleri aynı kelimeler çakışma olarak ele alınmıştır ve karakter karşılaştırma maliyetini en aza indirmek için B' uzunluklu örnekleri metnin önekiyle uyumlu kelimelerin tutulduğu ÖNEK tablosu geliştirilmiştir. Böylelikle soneki ve öneki girdi metniyle aynı olan kelimelerin kontrol edilmesi sağlanmıştır.

WM algoritmasının arama aşamasında ilk önce tüm diziler metnin başlangıcına hizalanır ve en kısa uzunluklu dizi (m) belirlenir. Tüm dizilerin ilk m karakterleri ele alınır. Tüm dizilerin ATLAMA tablosu oluşturulduktan sonra metnin sonekinin hash değeri (h) hesaplanır ve bu sonekin atlama tablosunda olup olmadığı kontrol edilir. Eğer bu sonek yoksa $m - B + 1$ birim kadar diziler sola doğru kaydırılır. Eğer bu sonek varsa atlama değeri kontrol edilir. Atlama değeri 0'dan farklı ise bu değer kadar tüm diziler sola doğru kaydırılır. Sonekin atlama değeri 0'a eşit ise girdi metninin B' uzunluklu hash değeri (h') hesaplanır ve bu değere sahip diziler döndürülür. Önek uyumsuzluğu veya karakterlerden herhangi birinin eşleşmemesi durumunda tüm diziler B birim kadar sağa doğru kaydırılır. Son olarak, sonek ve öneki uyan tüm diziler sırasıyla soldan sağa doğru tek tek karakter karşılaştırması yapılarak kontrol edilir. Herhangi bir dizi eşleşirse tüm diziler bir (1) birim sağa doğru kaydırılır. Tüm bu işlemler girdi metninin sonuna kadar devam eder. DNA sekans verilerinde WM çoklu kesin dizi eşleştirme algoritması uygulandığında yapılan deneysel çalışmalar sonucu bu algoritmanın bazı kısıtlamalara sahip olduğu gözlemlenmiştir. Bunlar;

Kısıtlama 1: Alfabe sayısı Σ belli olan farklı uzunluktaki olası üretilebilecek farklı hash değeri sayısı $\Sigma^{blok\ uzunlu\u{g}u}$ dur. DNA sekanslarında alfabe sayısı az olduğundan üretilebilecek farklı hash değeri sayısı azalacağı için DNA sekanslarındaki hash çakışma olasılığının İngilizce alfabelerindeki hash çakışma olasılığından çok yüksek olduğu görülmektedir. DNA sekanslarının alfabe sayısı $\Sigma = 4$ ve WM algoritmasının blok uzunluğu $B = 2$ olduğundan üretilebilecek farklı hash değeri sayısı $\Sigma^B = 16$ 'dır. Soneki ve öneki 16 farklı alt dizi $\{AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT\}$ olan çok sayıda dizi olması hash çakışmasına neden olur. Bu nedenle sonek

değerlerinde çakışma olduğu için önek kontrolü yapılır ve hash hesaplaması çalışma süresi artar.

Kısıtlama 2: Soneki eşleşen metnin önekinin hash değerlerinin hesaplanmasında da çakışmaya neden olur. Çok sayıda dizide B uzunluklu öneke sahip dizilerin karakter kontrolü yapılacağından çalışma süresi artar.

Kısıtlama 3: Önek eşleşmezse veya önekten sonra gelen herhangi bir karakter eşleşmezse diziler $B = 2$ kadar metin üzerinde kaydırılır. Bu yüzden diziler daha az kaydırıldığından arama süresi artmaktadır.

4.3.2. Önerilen Yaklaşım: WM-q Algoritması

DNA dizilerinde çoklu kesin dizi eşleştirmesi yaparken geleneksel WM algoritması bazı eksikliklere sahiptir. Bu eksikliklere çözüm üreten önerdiğimiz WM-q algoritması, tüm performans ölçütlerinde WM algoritmasına göre daha verimlidir.

Önerilen hash fonksiyonu, DNA'daki sonek ve önek dizilerinin hash değerlerindeki çakışmaları ortadan kaldırır. WM algoritması tarafından kullanılan sabit blok uzunluğu $B = 2$ 'yi kullanmak yerine, $q = 8$ 'e kadar daha uzun bir blok uzunluğu kullanılmaktadır. Önerilen yaklaşımda blok uzunluğu B yerine q olarak temsil edilmektedir. Maksimum blok uzunluğu için ($q = 8$), $4^8 = 65536$ adet eşsiz hash değerleri üretilebilir.

DNA dizilerinde mükemmel hash fonksiyonu kullanılarak, q -uzunluklu soneklerin 0'dan büyük hash değerlerinin olasılığı artmıştır. Blok uzunluğu arttığında (4^q adet dizide farklı q -uzunluklu sonek karakterleri yoksa), q -uzunluklu sonek eşleşse bile dizi $0 \leq \text{kaydırma_degeri} \leq m - q + 1$ kadar kaydırılır. Ayrıca, blok uzunluğu artırılarak, q -uzunluklu soneklerinin hash çakışma olasılığı azaldığından, dizinin öneki kontrol edilmeden $m - q + 1$ kadar dizi kaydırılır. Ayrıca, dizilerin sonekleri eşleşse ve önek veya q -uzunluklu önekinden sonraki herhangi bir karakter eşleşmezse bile, dizi $B = 2$ birim yerine q birim kaydırılır. Böylece önerilen yaklaşımda daha fazla kayma miktarı elde edilmiştir. q sayısı arttıkça sonek ve önekteki hash çakışma sayısı da en aza indirilmiştir. WM-q algoritmasının genel işleyişini şu şekilde özetleyebiliriz; tüm diziler arasında en kısa dizi uzunluğu bulunur ve bu uzunluk m olarak kabul edilir. İlk olarak, tüm diziler metnin başından itibaren hizalanır. Daha sonra metnin ilk m karakterinin q -uzunluk sonekinin hash

değeri hesaplanır. Bu hash değeri SHIFT tablosunda ise metnin q -uzunluklu önekinin hash değeri kontrol edilir. Aksi takdirde, diziler metin üzerinde $m - q + 1$ birim sağa kaydırılır. Metnin q -uzunluklu önekinin hash değeri ÖNEK tablosunda değilse, diziler metin üzerinde q birim sağa kaydırılır. Eğer önek değeri varsa, bu önek ile dizilerin karakterleri karşılaştırılır. Önerilen WM- q algoritmasında kullanılan mükemmel hash fonksiyonu sayesinde, dizinin tüm karakterlerini kontrol etmek

Algoritma 4.6. WM- q algoritmasının ön işleme aşamasının sözde kodu ($q = 5$ için).

```

1.  $q \leftarrow 5, \quad d \leftarrow \text{dizi\_sayisi}$ 
2. ATLAMA tablosunun tüm elemanlarına  $m - q + 1$  değerini ata
3. Algoritma ATLAMA_tablosu_olustur( $P, d$ )
4.  $\text{minUzunluk} \leftarrow \text{minimum\_dizi\_uzunlugu}$ 
5. for  $i \leftarrow 0$  to  $d - 1$  do
6.   for  $j \leftarrow 0$  to  $\text{minUzunluk} - q$  do
7.      $\text{shift} \leftarrow \text{minUzunluk} - j + q$ 
8.      $\text{sonek}[q] \leftarrow \{P[i][j], P[i][j + 1], P[i][j + 2], P[i][j + 3], P[i][j + 4]\}$ 
9.      $\text{ekle}(\text{ATLAMA}, \text{sonek}, \text{shift})$ 
10. Algoritma ÖNEK_tablosu_olustur( $P, d$ )
11. for  $i \leftarrow 0$  to  $d - 1$  do
12.    $\text{önek}[q] \leftarrow \{P[i][0], P[i][1], P[i][2], P[i][3], P[i][4]\}$ 
13.    $\text{ekle}(\text{ÖNEK}, \text{önek}, P[i])$ 

```

Algoritma 4.7. WM- q algoritmasının arama aşamasının sözde kodu ($q = 5$ için).

```

1. Algorithm ATLAMA_tablosu_olustur( $T, P, d, q, \text{minUzunluk}$ )
2.  $\text{atla} \leftarrow \text{minUzunluk} - q + 1$ 
3.  $\text{basla} \leftarrow \text{minUzunluk} - 1$ 
4.  $\text{metin\_uzunlugu} \leftarrow \text{uzunluk}(T)$ 
5. while  $\text{basla} < \text{metin\_uzunlugu}$  do
6.    $\text{altDizi}[q] \leftarrow \{T[\text{basla} - 4], T[\text{basla} - 3], T[\text{basla} - 2], T[\text{basla} - 1], T[\text{basla}]\}$ 
7.    $\text{shift} \leftarrow \text{getir}(\text{ATLAMA}, \text{altDizi})$  //  $T$ 'nin  $q$ -uzunluklu sonek atlama değeri
8.   if  $\text{shift} = -1$  then // sonek eşleşmezse diziler  $m-q+1$  birim sağa kaydırılır
9.      $\text{basla} \leftarrow \text{basla} + \text{atla}$ 
10.  else if  $\text{shift} = 0$  then // sonek eşleşirse,  $T$ 'nin  $q$ -uzunluklu önekinin kontrol et
11.     $\text{preS} \leftarrow \text{basla} - \text{minUzunluk} - 1$  // önek başlama indisi
12.     $\text{önek}[q] \leftarrow \{T[\text{preS}], T[\text{preS} + 1], T[\text{preS} + 2], T[\text{preS} + 3], T[\text{preS} + 4]\}$ 
13.     $\text{diziler} \leftarrow \text{çıkar}(\text{ÖNEK}, \text{önek})$  //  $q$ -uzunluklu öneke sahip diziler
14.    while  $\text{dizi}$  do //  $q$ -uzunluklu öneke sahip dizileri kontrol et
15.       $\text{flag} \leftarrow 0$ 
16.      for  $i \leftarrow q$  to  $\text{dizi\_uzunlugu} - q$  do // ilk-son  $q$  karakter harici kontrol et
17.        if  $\text{dizi} \rightarrow \text{dizi}[i] \neq T[\text{preS} + i]$  then
18.           $\text{gecici} \leftarrow 1$ 
19.           $\text{dizi} \leftarrow \text{sonraki\_dizi}$ 
20.         $\text{basla} \leftarrow \text{basla} + 1$  // herhangi dizi eşleşirse, dizileri 1 birim sağa kaydır
21.      else then  $\text{basla} \leftarrow \text{basla} + \text{shift}$  // önek-karakter eşleşmezse  $q$  birim kaydır

```

yerine, q -uzunluklu sonek ve önek karakterlerinin dışında kalan karakterler karşılaştırılır. Bu nedenle WM- q algoritmasında dizi uzunluğu $m \leq 2q$ olduğunda karakter karşılaştırma sayısı 0 olur. Algoritma 4.6'da WM- q algoritmasının ön işleme aşamasının sözde kodu ve Algoritma 4.7'de ve arama aşamasının sözde kodu gösterilmiştir.

4.3.3. Önerilen Yaklaşımın Başarılı ve Başarısız Eşleşme Durum Örnekleri

Bu bölümde önerilen WM- q algoritması ile WM algoritması arasındaki farklar örneklerle gösterilmiştir. İlk olarak, WM algoritması ve önerilen WM- q algoritması, ön işleme aşamasında verilen diziler için ATLAMA ve ÖNEK tablolarını üretir. Örneklerde dizi uzunlukları $m = 10$ ve WM- q algoritmasında q değeri 5 olarak seçilirken, WM algoritmasında B değeri sadece 2 olabilir.

Şekil 4.14, verilen girdi metnin aranan diziler için WM algoritmasının başarısız dizi eşleştirme durumunu göstermektedir. Diziler metnin başına hizalandıktan sonra, metnin B uzunluğundaki sonek dizilerinin (AC) h hash değeri hesaplanır. ATLAMA[h]'nin hash değeri 0'a eşit olduğu için, metnin B -uzunluklu önek karakterlerinin (TG) h' hash değeri hesaplanır. ÖNEK[h'] değeri 1, 2, 3, 4 numaralı dizileri döndürür ve bu diziler sırayla tek tek karşılaştırılır.

Sonuç olarak Şekil 4.14'te 23 karakter karşılaştırması ve 2 hash karşılaştırması yapılmıştır. Başarısız bir dizi eşleştirme durumu olduğundan, diziler metin üzerinde $B = 2$ birim sağa kaydırılır.

ÖNEK TG= Dizi 1,2,3,4										ATLAMA AC=0									
T =	T	G	C	A	T	T	G	G	A	C	A	C	C						
P1 =	T	G	C	A	A	T	G	C	A	A									
P2 =	T	G	C	A	C	T	G	C	A	C									
P3 =	T	G	C	A	G	T	G	C	A	G									
P4 =	T	G	C	A	T	T	G	C	A	T									
P5 =	C	A	C	A	T	G	G	C	A	C									

→ Tüm dizileri B=2 birim kaydır

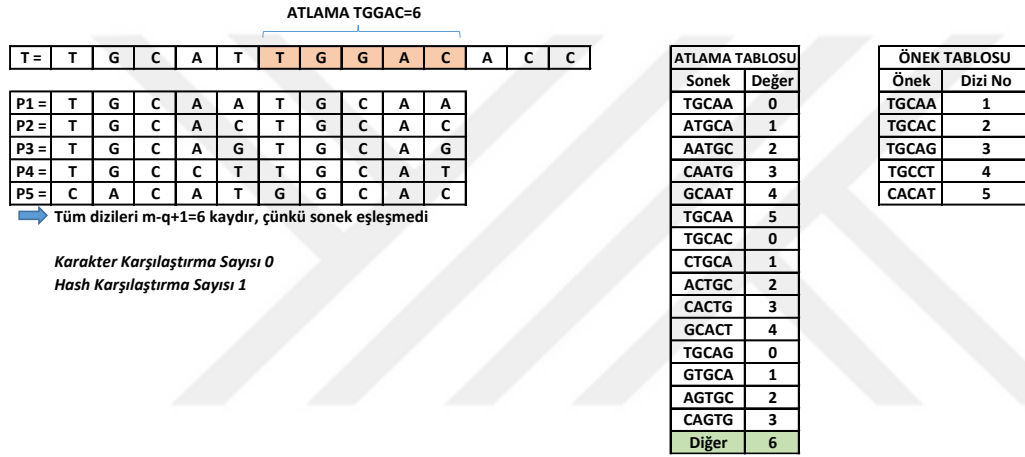
Karakter Karşılaştırma Sayısı 23
Hash Karşılaştırma Sayısı 2

ATLAMA TABLOSU	
Sonek	Değer
AA	0
AC	0
AG	0
AT	0
CA	1
CT	4
GC	2
GG	3
GT	4
TG	3
TT	4
Diğer	9

ÖNEK TABLOSU	
Önek	Dizi No
CA	5
TG	1,2,3,4

Şekil 4.14. WM algoritmasının eşleşme durumu.

Şekil 4.15, metinde aranan diziler için önerilen WM-q algoritmasının başarısız dizi eşleştirme durumunu göstermektedir. Şekil 4.15, WM-q algoritmasındaki q -uzunluklu soneki eşleşmemiştir. Diziler metnin başlangıcına hizalandıktan sonra, metnin $q = 5$ uzunluklu soneki olan TGGAC'nin h hash değeri hesaplanır. ATLAMA[h]'nin değeri 0'a eşit olmadığı için, diziler metin üzerinde $m - q + 1 = 6$ birim sağa kaydırılır. Önerilen WM-q algoritmasında q uzunluğunda bloklar kullanıldığında metnin soneklerdeki hash çakışması kaldırıldığı için, sonek uyumsuzluğu olduğundan diziler kaydırılır. Böylece, WM-q algoritması, ekstra önek hash karşılaştırması ve karakter karşılaştırması yapmadan metin üzerinde aramaya devam eder. Böylece performans metrikleri açısından daha iyi sonuçlar elde edilmektedir.



Şekil 4.15. WM-q algoritmasının eşleşmeme durumu (sonek eşleşmemesi).

Şekil 4.16, metinde aranan diziler için önerilen WM-q algoritmasının başarısız dizi eşleştirmesini göstermektedir. q -uzunluklu sonekinde bir eşleşme olmasına rağmen, önek veya karakter karşılaştırmasında bir eşleşmeme durumu vardır. Diziler metnin başlangıcına hizalandıktan sonra metnin $q = 5$ uzunluklu soneki TGCAC'nin h hash değeri hesaplanır. ATLAMA[h] değeri 0 olduğu için metnin $q = 5$ uzunluklu öneki TGCAC'nin h' hash değeri hesaplanır. ÖNEK[h'], önerilen WM-q algoritmasında önek hash çakışması farklı blok uzunlukları kullanılarak kaldırıldığından herhangi bir değer döndürmez. Geleneksel WM algoritmasından farklı olarak, başarısız bir önek eşleşmesi olduğu için, diziler metin üzerinde $B = 2$ yerine $q = 5$ birim sağa kaydırılır. WM-q algoritması, ekstra karakter karşılaştırması yapmadan metinde aramaya devam eder ve önek uyumsuzluğu olduğundan dizileri daha fazla kaydırır. Böylece performans metrikleri açısından daha iyi sonuçlar elde edilmektedir.

TGCAT ÖNEK TABLOSUNDA DEĞİL										ATLAMA TGCAC=0									
T = T G C A T T G C A C A C C																			
P1 =	T	G	C	A	A	T	G	C	A	A									
P2 =	T	G	C	A	C	T	G	C	A	C									
P3 =	T	G	C	A	G	T	G	C	A	G									
P4 =	T	G	C	C	T	T	G	C	A	T									
P5 =	C	A	C	A	T	G	G	C	A	C									

→ Tüm dizileri q=5 birim kaydır, çünkü önek eşleşmedi

Karakter Karşılaştırma Sayısı 0
Hash Karşılaştırma Sayısı 2

ATLAMA TABLOSU				ÖNEK TABLOSU	
Sonek	Değer	Sonek	Değer	Önek	Dizi No
TGCAA	0	GCAGT	4	TGCAA	1
ATGCA	1	TGCAG	5	TGCAC	2
AATGC	2	TGCAT	0	TGCAG	3
CAATG	3	TTGCA	1	TGCCT	4
GCAAT	4	ATTGC	2	CACAT	5
TGCAA	5	CATTG	3		
TGCAC	0	GCATT	4		
CTGCA	1	TGCAT	5		
ACTGC	2	GGCAC	0		
CACTG	3	TGGCA	1		
GCACT	4	ATGGC	2		
TGCAG	0	CATGG	3		
GTGCA	1	ACATG	4		
AGTGC	2	CACAT	5		
CAGTG	3	Diğer	6		

Şekil 4.16. WM-q algoritmasının eşleşmeme durumu (önek eşleşmemesi).

Şekil 4.17, metinde aranan diziler için WM algoritmasının dizi eşleştirme durumunu gösterir. Diziler metnin başına hizalandıktan sonra metnin $B = 2$ uzunluklu sonekinin (AC) h hash değeri hesaplanır. ATLAMA[h] değeri 0 olduğu için metnin $B = 2$ uzunluklu önekinin (TG) h' hash değeri hesaplanır. ÖNEK[h'] değeri 1, 2, 3, 4 numaralı dizilerini döndürür ve bu dizilerin karakterleri dizi sırasına göre tek tek karşılaştırılır. Sonuç olarak 1 başarılı dizi eşleşmesinin olduğu 25 karakter karşılaştırması ve 2 hash karşılaştırması yapılmış ve tüm diziler metin üzerinde 1 birim sağa kaydırılmıştır.

ÖNEK TG= Pattern 1,2,3,4										ATLAMA AC=0									
T = T G C A C T G C A C A C C																			
P1 =	T	G	C	A	A	T	G	C	A	A									
P2 =	T	G	C	A	C	T	G	C	A	C									
P3 =	T	G	C	A	G	T	G	C	A	G									
P4 =	T	G	C	A	T	T	G	C	A	T									
P5 =	C	A	C	A	T	G	G	C	A	C									

→ Tüm dizileri 1 birim kaydır, Dizi 2 eşleşti

Karakter Karşılaştırma Sayısı 25
Hash Karşılaştırma Sayısı 2

ATLAMA TABLOSU	
Sonek	Değer
AA	0
AC	0
AG	0
AT	0
CA	1
CT	4
GC	2
GG	3
GT	4
TG	3
TT	4
Diğer	9

ÖNEK Tablosu	
Önek	Dizi No
CA	5
TG	1,2,3,4

Şekil 4.17. WM algoritmasının eşleşme durumu.

Şekil 4.18, metinde aranan diziler için önerilen WM-q algoritmasının başarılı dizi eşleştirme durumunu göstermektedir. Diziler metnin başlangıcına hizalandıktan sonra metnin $q = 5$ uzunluklu soneki TGCAC'nin h hash değeri hesaplanır. ATLAMA[h]'nin değeri 0'a eşit olduğundan, metnin $q = 5$ uzunluklu öneki olan TGCAC'nin h' hash değeri hesaplanır. Önerilen WM-q algoritmasında farklı blok uzunlukları kullanan ÖNEK[h'] değeri Dizi 2'yi döndürür. Ancak, WM algoritmasından farklı olarak WM-q algoritması Dizi 2'deki karakterleri tek tek karşılaştırmaz. DNA dizilerinde eşsiz hash değerleri üretildiğinden, dizi uzunluğu sonek ve önek uzunlukları toplamına eşit

olduğunda, herhangi bir karakter karşılaştırılmadan dizi eşleştirilir. Sonuç olarak, 1 başarılı dizi eşleştirmesi, 0 karakter karşılaştırması ve 2 hash karşılaştırması yapılan Şekil 4.18'deki diziler metin üzerinde 1 birim sağa kaydırılmıştır. Böylece önerilen WM-q algoritmasında kullanılan hash fonksiyonu sayesinde karakter karşılaştırma sayısı minimize edilmiştir.

TGCAC ÖNEK TABLOSUNDA, Dizi 2 ATLAMA TGCAC=0

T =	T	G	C	A	C	T	G	C	A	C	A	C	C
-----	---	---	---	---	---	---	---	---	---	---	---	---	---

P1 =	T	G	C	A	A	T	G	C	A	A
P2 =	T	G	C	A	C	T	G	C	A	C
P3 =	T	G	C	A	G	T	G	C	A	G
P4 =	T	G	C	A	T	T	G	C	A	T
P5 =	C	A	C	A	T	G	G	C	A	C

→ Tüm dizileri 1 birim kaydır, Dizi 2 eşleştirdi

Karakter Karşılaştırma Sayısı 0
Hash Karşılaştırma Sayısı 2

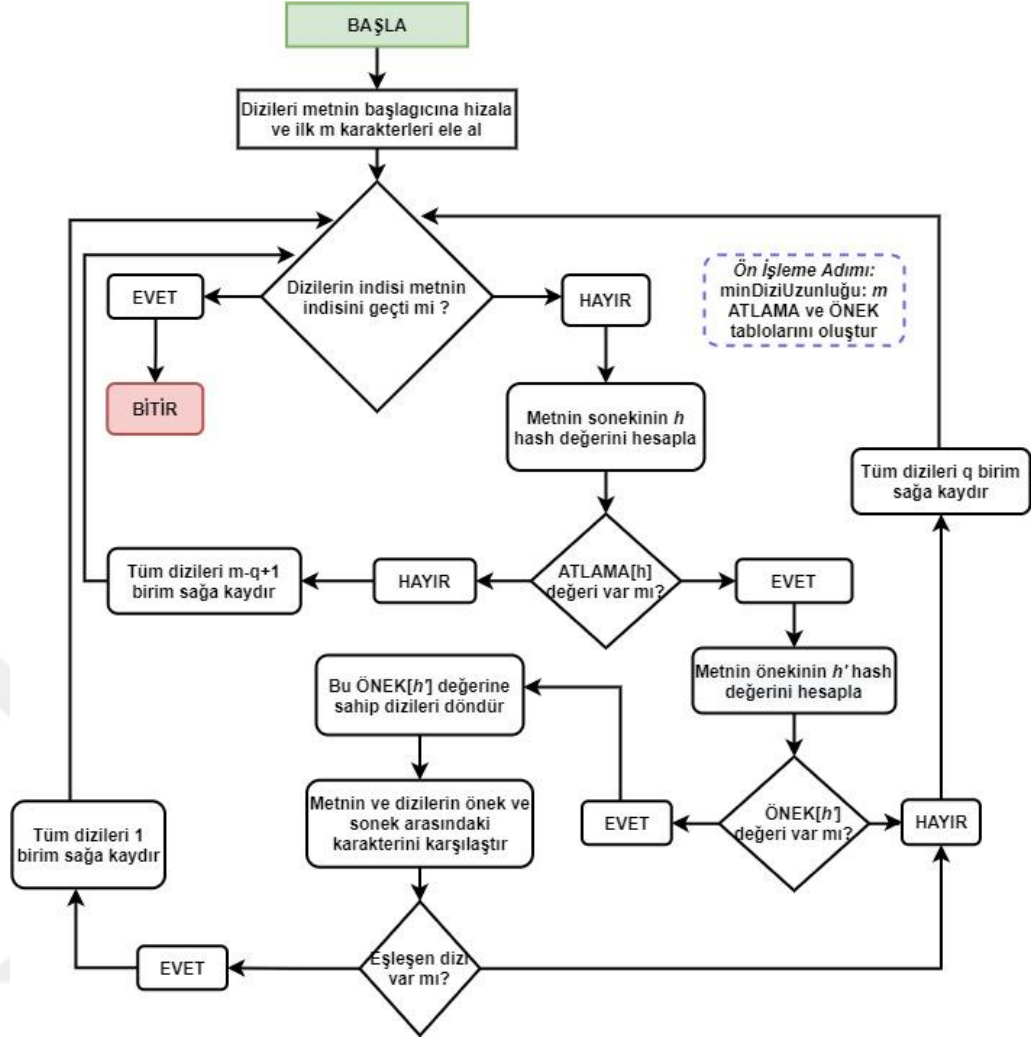
ATLAMA TABLOSU				ÖNEK TABLOSU	
Sonek	Değer	Sonek	Değer	Önek	Dizi No
TGCAA	0	GCAGT	4	TGCAA	1
ATGCA	1	TGCAG	5	TGCAC	2
AATGC	2	TGCAT	0	TGCAG	3
CAATG	3	TTGCA	1	TGCCT	4
GCAAT	4	ATTGC	2	CACAT	5
TGCAA	5	CATTG	3		
TGCAC	0	GCATT	4		
CTGCA	1	TGCAT	5		
ACTGC	2	GGCAC	0		
CACTG	3	TGGCA	1		
GCACT	4	ATGGC	2		
TGCAG	0	CATGG	3		
GTGCA	1	ACATG	4		
AGTGC	2	CACAT	5		
CAGTG	3	Diğer	6		

Şekil 4.18. WM-q algoritmasının eşleşme durumu.

4.3.4. Önerilen Yaklaşımın Genel Çerçevesi

Önerilen WM-q algoritmasının genel çerçevesi Şekil 4.19'da gösterilmiştir. WM-q algoritmasında, geleneksel WM algoritmasında B -uzunluklu bloklar yerine q -uzunluklu bloklar kullanılır. Şekil 4.18'de, sonek ve öneklerde bir eşleşme olmadığında, hash çakışma sayısı azaltılır ve gösterilen kaydırma değerleri kullanılır.

Şekil 4.19'da görüldüğü gibi ön işleme aşamasında öncelikle ATLAMA ve ÖNEK tabloları oluşturulur. Daha sonra arama aşamasında, diziler metnin başlangıcına hizalandıktan sonra metnin q -uzunluklu sonekinin hash değeri kontrol edilir. Hash değerleri eşleşirse, metnin q -uzunluklu önekinin hash değeri kontrol edilir; bu değer eşleşmezse, diziler metin üzerinde $m - q + 1$ birim sağa kaydırılır. Son olarak, metnin q -uzunluklu önekinin hash değeri eşleşirse, ilk ve son q -uzunluklu karakterler dışında kalan karakterler soldan sağa doğru kontrol edilir ve dizilerden biri eşleşirse tüm diziler metin üzerinde bir birim sağa kaydırılır; dizilerden biri eşleşmezse, diziler metin üzerinde q birim sağa kaydırılır. Tüm adımlar metnin sonuna kadar devam eder.



Şekil 4.19. Önerilen WM-q algoritmasının genel çerçevesi.

4.3.5. Önerilen Yaklaşım İçin Çalışan Örnekler

Geleneksel WM algoritması ve önerilen WM-q algoritması için $T = CTGGACAGTGTCTGACTCCAATGACCTAGCAGCTGACTGCCATA$ girdi metnindeki $P = \{ATGACCTAGC, CAGTGTCTGAC, GACTGCCATA\}$ dizi kümesini aramak için çalışılmış bir örnek bu bölümde gösterilmiştir. Hash karşılaştırma sayısı, karakter karşılaştırma sayısı ve eşleşen diziler her adımda gösterilmiştir. Başarısız hash ve karakter karşılaştırmaları açık kırmızı, başarılı hash karşılaştırmaları açık yeşil ve başarılı karakter karşılaştırmaları koyu yeşil renkte gösterilmiştir.

WM ve WM-q algoritmalarında ön işleme aşamasında oluşturulan ATLAMA ve ÖNEK tabloları sırasıyla Şekil 4.20 ve 4.21'de gösterilmiştir.

ATLAMA TABLOSU

Sonek	Değer
GC	0
AG	1
TA	0
CT	3
CC	3
AC	0
GA	1
TG	5
AT	1
CG	2
TC	3
GT	4
CA	2

ONEK TABLOSU

Onek	Dizi No
AT	1
CA	2
GA	3

Diziler

ATGACCTAGC

CAGTGTCGAC

GACTGCCATA

Şekil 4.20. WM algoritmasındaki atlama ve önek tabloları.

ATLAMA TABLOSU

Sonek	Değer
CTAGC	0
CCTAG	1
ACCTA	2
GACCT	3
TGACC	4
ATGAC	5
TCGAC	0
GTCGA	1
TGTCG	2
GTGTC	3
AGTGT	4
CAGTG	5
CCATA	0
GCCAT	1
TGCCA	2
CTGCC	3
ACTGC	4
GACTG	5

ONEK TANLOSU

Onek	Dizi No
ATGAC	1
GAGTG	2
GACTG	3

Diziler

ATGACCTAGC
CAGTGTCGAC
GACTGCCATA

Şekil 4.21. WM-q algoritmasındaki atlama ve önek tabloları.

WM algoritmasının çalışan örneği Şekil 4.22’de gösterilmiştir. Tüm modeller başarıyla eşleştirilir. Karakter karşılaştırma sayısı 31 ve hash karşılaştırma sayısı 20’dir.

Adım 1	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır TG=5 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 2	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek CA=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır AC=0 → Dizi 2 eşleşti, 1 kaydır	(10 Karakter Karşılaştırması) (2 Hash Karşılaştırması)
Adım 3	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CT=3 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 4	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CA=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 5	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır AT=1 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 6	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır TG=5 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 7	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Dizi 2'nin Öneki Eşleşti Ama 3. karakter eşleşmedi Önek CA=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır TA=0 → Kaydır B=2	(1 Karakter Karşılaştırması) (2 Hash Karşılaştırması)
Adım 8	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A (10 Karakter Karşılaştırması) (2 Hash Karşılaştırması) Önek AT=1 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır GC=0 → Dizi 1 eşleşti, 1 kaydır	
Adım 9	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CA=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 10	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek eşleşmedi → Kaydır B=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır GC=0	(2 Hash Karşılaştırması)
Adım 11	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır TG=5 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 12	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek eşleşmedi → Kaydır B=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır GC=0	(2 Hash Karşılaştırması)
Adım 13	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CA=2 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 14	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Dizi 3 Eşleşti → 1 Kaydır Önek GA=3 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A → Kaydır TA=0	(10 Karakter Karşılaştırması) (2 Hash Karşılaştırması)
Adım 15	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Sonuç: Toplam Karakter Karşılaştırma Sayısı: 31 Toplam Hash Karşılaştırma Sayısı: 20 metnin boyutunu aştı	A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A

Şekil 4.22. WM algoritmasının çalışan örnekleri.

WM-q algoritmasının çalışan örneği Şekil 4.23'te gösterilmiştir. Benzer şekilde, tüm diziler başarıyla eşleşmiştir. Şekil 4.23'te gösterildiği gibi karakter karşılaştırma sayısı 0 ve hash karşılaştırma sayısı 13'tür. Önerilen yaklaşımda, dizi uzunluğu, dizinin q -uzunluklu soneki ve önekinin toplamına eşit olduğunda. ($m \leq 2q$), dizi eşleştirme, karakterler karşılaştırılmadan gerçekleştirilir. Çalışılan örneklerde önerilen WM-q algoritması ile daha iyi sonuçlar elde edildiği gösterilmiştir.

Adım 1	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CAGTG=5 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 2	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek CAGTG=2 Kaydır TCGAC=0 → Dizi 2 eşleşti, 1 kaydır A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(2 Hash Karşılaştırması)
Adım 3	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CAGCT=6 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 4	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CAATG=6 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 5	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Kaydır CCTAG=1 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 6	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek ATGAC=1 → Kaydır CTAGC=0 → Dizi 1 eşleşti, 1 kaydır A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(2 Hash Karşılaştırması)
Adım 7	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır TAGCA=6 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 8	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Kaydır CTGAC=6 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 9	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A → Shift GCCAT=1 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(1 Hash Karşılaştırması)
Adım 10	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Kaydır CCATA=0 → Dizi 3 eşleşti, 1 kaydır A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	(2 Hash Karşılaştırması)
Adım 11	C T G G A C A G T G T C G A C T C C A A T G A C C T A G C A G C T G A C T G C C A T A Önek GACTG=3 A T G A C C T A G C C A G T G T C G A C G A C T G C C A T A	
Sonuç:	Toplam Karakter Karşılaştırma Sayısı: 0 Toplam Hash Karşılaştırma Sayısı: 13	metnin boyutunu aştı

Şekil 4.23. WM-q algoritmasının çalışan örnekleri.

AC, CW, WM ve önerilen WM-q algoritmasının zaman karmaşıklığı analizi Çizelge 4.6'da gösterilmiştir. AC algoritmasının en iyi ve en kötü durum karmaşıklıkları $O(n)$ 'dir. Ancak AC algoritmasının geliştirilmiş bir versiyonu olan CW algoritmasının en iyi durum karmaşıklığı alt lineerdir. Hash tabanlı WM algoritmasının en iyi durum zaman karmaşıklığı $O((n \log_{|\Sigma|} |P|)/m)$ ve en kötü durum zaman karmaşıklığı $O(n|P| \log_{|\Sigma|} |P|)$ (Aho and Corasick, 1975). Önerilen WM-q algoritmasında q uzunluklu sonek ve önek karakterlerinin karşılaştırması yoktur. d adet dizinin toplam uzunluğu $|P|$ 'dir. Sonuç olarak, d adet dizinin sonek ve önek karakterleri karşılaştırılmadığından, karşılaştırılan toplam karakter sayısı $|P - 2dq|$ 'dir. Dolayısıyla, önerilen WM-q algoritmasının en iyi durum zaman karmaşıklığı $O((n \log_{|\Sigma|} |P - 2dq|)/m)$ ve en kötü durum zaman karmaşıklığı $O(n|P - 2dq| \log_{|\Sigma|} |P - 2dq|)$ 'dir. Bu nedenle, önerilen WM-q algoritmasının zaman karmaşıklığı, WM algoritmasının zaman karmaşıklığından daha iyidir.

Çizelge 4.6. Çoklu dizi eşleştirme algoritmalarının zaman karmaşıklığı.

Algoritma	Zaman Karmaşıklığı	Arama Tipi	Veri Setleri
AC	En İyi: $O(n)$	Önek Tabanlı	Bibliyografik Metin
	En Kötü: $O(n)$		
CW	En İyi: Sublinear	Sonek Tabanlı	İngilizce Metin
	En Kötü: $O(nm)$		
WM	En İyi: $O((n \log_{ \Sigma } P)/m)$	Hash Tabanlı	Wall Street Dergisi
	En Kötü: $O(n P \log_{ \Sigma } P)$		
WM-q	En İyi: $O((n \log_{ \Sigma } P - 2dq)/m)$	Hash Tabanlı	DNA Sekansları
	En Kötü: $O(n P - (2dq) \log_{ \Sigma } P - 2dq)$		

4.4. DNA Sekansları İçin q -Gram Hash Karşılaştırmasına Dayalı Çoklu Kesin Dizi Eşleştirme Algoritması

DNA sekanslarında çoklu dizi eşleştirme problemini çözmek için kullanılan WM algoritmasındaki çözüm üreten mükemmel hash fonksiyonunu kullanarak yeni bir yöntem önerilmiştir. Kullanılan mükemmel hash fonksiyonuna ek olarak q -gram hash karşılaştırmasına dayalı çoklu kesin dizi eşleştirme yaklaşımı önerilmiştir. Bu bölümde önerilen yaklaşımdan ve çalışma örneklerinden bahsedilerek WM algoritmasındaki eksiklere yönelik yapılan katkılardan da bahsedilmiştir.

4.4.1. Önerilen Yaklaşım: q -Gram WM Algoritması

DNA sekanslarında hash çakışmasını ortadan kaldıran hash fonksiyonunu kullanarak sonek ve önek dizilerinin hash değerlerindeki çakışma ortadan kaldırılmıştır. Geleneksel WM algoritmasındaki sabit blok uzunluğu $B=2$ yerine mükemmel hash fonksiyonu kullanarak blok uzunluğu artırılmıştır. Bunun nedeni, bilgisayar mimarilerinde 8-bit kayıtcı kullanıldığından 4^8 farklı değer üretildiği için blok uzunluğu maksimum 8'dir. Önerilen yaklaşımda blok uzunluğu B yerine q ile ifade edilir. Bölüm 5.1'de elde edilen deneysel çalışmalar sonucu en etkili blok uzunluğu olarak belirlenen değer bu çalışma için $q = 8$ olarak seçilmiştir.

Aranan dizilerin soneklerinde hash çakışması ortadan kaldırılmıştır ve böylece önek kontrolü yapmadan diziler kaydırılır. Soneklerde olduğu gibi öneklerdeki hash çakışması da ortadan kaldırılmıştır ve kontrol edilecek dizi sayısı optimize edilmiştir. Ayrıca, q -uzunluklu önek veya önekten sonra gelen herhangi bir karakter eşleşmezse, WM algoritmasının aksine, $B = 2$ birim kadar değil $q = 8$ birim kadar diziler metin üzerinde kaydırılarak önerilen yaklaşımda daha fazla kaydırma miktarı elde edilmiştir.

Önerilen yaklaşımın genel işleyişi, ön işleme adımı WM algoritmasının temeline dayanmaktadır ancak ATLAMA ve ÖNEK tablolarındaki değerleri $q = 8$ göz önünde bulundurularak hesaplanır. Arama aşaması da WM algoritmasına benzer işlemektedir. Yani, metnin q -uzunluklu sonekinin hash değeri ATLAMA tablosunda varsa metnin q -uzunluklu öneki kontrol edilir, aksi halde diziler $m - q + 1$ birim kadar sağa kaydırılır. Metnin q -uzunluklu önekinin hash değeri ÖNEK tablosunda yoksa diziler q birim sağa kaydırılır. Eğer öneki varsa bu öneke sahip dizilerin ilk olarak önek karakterinden sonra gelen ilk karakter karşılaştırılır. Buradaki amaç, önerdiğimiz q -gram hash karşılaştırmasını yapmadan önce karakter uyumsuzluğu varsa dizinin eşleşmediğini erkenden tespit edip hash karşılaştırma maliyetinden kazanç elde etmektir. Eğer bu karakter eşleşirse, dizilerin tüm karakterlerini karşılaştırmak yerine önerilen yaklaşımda kullanılan mükemmel hash fonksiyonu sayesinde q -uzunluklu sonek ve önek karakterleri arasındaki karakterlerin q -gram hash karşılaştırması yapılır.

Verilen m uzunluklu dizide toplam $\left(\left\lfloor \frac{m}{q} \right\rfloor - 2\right)$ adet q -gram hash karşılaştırması yapılır. Sonek ve önek mükemmel hash fonksiyonu kullanılarak eşleştiği için tekrardan kontrol etmemize gerek yoktur ve toplam q -gram

uzunluklu karakter sayısından iki (2) adet q -gram çıkarmış oluruz. Yapılan sonek ve önek hash karşılaştırma sayılarına ek olarak bu q -gram hash karşılaştırma sayıları da dahil edilir. Algoritma 4.8’de önerilen yaklaşımın ön işleme aşamasının, Algoritma 4.9’da arama aşamasının sözde kodları gösterilmiştir. ATLAMA (h) ve ÖNEK (h') tablolarının hash değerleri ($q = 8$ için) mükemmel hash fonksiyonu kullanılarak Eşitlik 4.4 ve 4.5’te gösterildiği gibi hesaplanır.

$$h = ((T[m-1] \& 0b110) \ll 13) + ((T[m-2] \& 0b110) \ll 11) + ((T[m-3] \& 0b110) \ll 9) + ((T[m-4] \& 0b110) \ll 7) + ((T[m-5] \& 0b110) \ll 5) + ((T[m-6] \& 0b110) \ll 3) + ((T[m-7] \& 0b110) \ll 1) + ((T[m-8] \& 0b110) \gg 1) \quad (4.4)$$

$$h' = ((T[0] \& 0b110) \ll 13) + ((T[1] \& 0b110) \ll 11) + ((T[2] \& 0b110) \ll 9) + ((T[3] \& 0b110) \ll 7) + ((T[4] \& 0b110) \ll 5) + ((T[5] \& 0b110) \ll 3) + ((T[6] \& 0b110) \ll 1) + ((T[7] \& 0b110) \gg 1) \quad (4.5)$$

Algoritma 4.8. q -Gram WM algoritmasının ön işleme aşamasının sözde kodu.

-
1. $q \leftarrow 8, \quad d \leftarrow \text{desen_sayisi}$
 2. ATLAMA tablosunun tüm elemanlarına $m - q + 1$ ata
 3. **Algoritma** ATLAMA_tablosu(P, d)
 4. $\text{minUzunluk} \leftarrow \text{min_desen_uzunlugu}$
 5. **for** $i \leftarrow 0$ **to** $d - 1$ **do**
 6. **for** $j \leftarrow 0$ **to** $\text{minUzunluk} - q$ **do**
 7. $\text{shift} \leftarrow \text{minUzunluk} - j + q$
 8. $\text{önek}[q] \leftarrow \{P[i][j], P[i][j+1], P[i][j+2], P[i][j+3], P[i][j+4], P[i][j+5], P[i][j+6], P[i][j+7]\}$
 9. $\text{ekle}(\text{ATLAMA}, \text{sonek}, \text{shift})$
 10. **Algoritma** ÖNEK_tablosu(P, d)
 11. **for** $i \leftarrow 0$ **to** $d - 1$ **do**
 12. $\text{önek}[q] \leftarrow \{P[i][0], P[i][1], P[i][2], P[i][3], P[i][4]\}$
 13. $\text{ekle}(\text{ÖNEK}, \text{önek}, P[i])$
-

Algoritma 4.9. q -Gram WM algoritmasının arama aşamasının sözde kodu.

```

1. Algoritma ATLAMA_tablosu( $T, P, d, q, minUzunluk$ )
2.  $atla \leftarrow minUzunluk - q + 1$ 
3.  $basla \leftarrow minUzunluk - 1$ 
4.  $metin\_uzunluđu \leftarrow uzunluk(T)$ 
5. while  $basla < metin\_uzunluđu$  do
6.    $altDizi[q] \leftarrow \{T[basla - 7], T[basla - 6], T[basla - 5], T[basla - 4], T[basla - 3], T[basla - 2], T[basla - 1], T[basla]\}$ 
7.    $shift \leftarrow getir(ATLAMA, altDizi)$ 
8.   if  $atlama = -1$  then
9.      $basla \leftarrow basla + atla$ 
10.  else if  $kaydır = 0$  then
11.     $preS \leftarrow basla - minUzunluk - 1$ 
12.     $önek[q] \leftarrow \{T[preS], T[preS + 1], T[preS + 2], T[preS + 3], T[preS + 4], T[preS + 5], T[preS + 6], T[preS + 7]\}$ 
13.     $düğüm\_ler \leftarrow düğüm\_döndür(ATLAMA, önek)$ 
14.    while  $düğüm\_ler$  do
15.       $gecici \leftarrow 0, eslesen \leftarrow 1, max\_eslesen \leftarrow yuvarla(m/q), i \leftarrow q$ 
16.      if  $(m[i + 1] = T[preS + 1])$  then
17.        while  $i < m - (eslesen * q)$  AND  $(FNG(önek, preS + (q * (eslesen - 1))) = FNG(desen, preS + (q * (eslesen - 1))))$  do
18.           $eslesen \leftarrow eslesen + 1$ 
19.          if  $(i \geq m - (eslesen * q))$  then
20.             $gecici \leftarrow 1$ 
21.          else  $gecici \leftarrow 0$ 
22.           $basla \leftarrow basla + 1$ 
23.        else then  $basla \leftarrow basla + shift$ 

```

4.4.2. WM ve q -Gram WM Algoritmalarının Karşılaştırma Sayısı Örnekleri

WM ve q -gram WM algoritması için örnek bir girdi metni ve dizi kümesi vererek bu algoritmalarındaki başarılı ve başarısız dizi eşleştirme durumlarının örneklerle gösterilmiştir. Girdi metni ve dizi uzunlukları 24 ve önerilen yaklaşım için $q = 8$ seçilmiştir. Başarılı karşılaştırmalar yeşil ile, başarısız karşılaştırmalar ise açık kırmızı ile gösterilmiştir.

Şekil 4.26 verilen T metninde aranan $D1, D2, D3$ dizilerinin başarısız dizi eşleştirme durumunu göstermektedir. Diziler T metninin başına hizalandıktan sonra metnin $B = 2$ uzunluklu AT sonek dizisinin h hash değeri hesaplanır. ATLAMA[h] değeri 0'a eşit olduğu için metnin $B = 2$ uzunluklu CT önek dizisinin h' hash değeri hesaplanır. ÖNEK[h'] değeri 1,2,3 nolu dizileri döndürür ve bu dizilerde baştan sona doğru karakterler tek tek karşılaştırılır. Sonuç olarak, 2 hash karşılaştırması ve 39 karakter karşılaştırması yapılan bu örnekte başarısız dizi eşleştirme olduğu için diziler $B = 2$ kadar metin üzerinde sağa kaydırılır.

Önek CT, (1-2-3)		Sonek AT atlama değeri=0	
T=	C T T G C A T C G G A C T C A T T A G T C C A T	Atlama Tablosu	Önek Tablosu
D1=	C T T G C A T C G G A C T C A T T A G T C C A T	Sonek	Önek
D2=	C T T G C A T A T C T G A A T C C A T G C G A T	Değeri	Desen No
D3=	C T T G C T T G A G G A C T G C A A C G T T A C	AA	6
Toplam Hash Karşılaştırma Sayısı: 2, Toplam Karakter Karşılaştırma Sayısı: 39		AC	0
B=2 birim kadar desenleri sağa kaydır.		AG	14
		AT	0
		CA	6
		CC	2
		CG	1
		CT	11
		GA	1
		GC	3
		GG	13
		GT	0
		TA	1
		TC	3
		TG	4
		TT	2

Şekil 4.24. WM algoritmasının başarısız dizi eşleştirme durumu.

Şekil 4.25 önerilen q -gram WM algoritmasındaki sonek ve önek eşleşme durumunu göstermektedir. T dizisinin soneki ATLAMA tablosunda olmadığından $m - q + 1 = 17$ birim diziler sağa kaydırılır. Soneki eşleşmesine rağmen önek eşleşme durumunda ise WM algoritmasındaki $B = 2$ aksine $q = 8$ birim kadar diziler kaydırılır. Herhangi bir karakter karşılaştırması yapmadan önerilen yaklaşımda sonek ve önek eşleşme durumlarında performans sağlanır.

Sonek TAGTCCAT atlama değeri=17		Sonek TATTCAT atlama değeri=0	
T=	C T T T G C A T C G G A C T C A T T A G T C C A T	Atlama Tablosu	Önek Tablosu
D1=	C T T T G C A T C G G A C T C A T T A T T C C A T	Sonek	Önek
D2=	C T T T G C A T A T C T G A A T C C A T G C G A T	Değeri	Desen No
D3=	C T T T G C T T G A G G A C T G C A A C G T T A C	TATTCAT	0
Toplam Hash Karşılaştırma Sayısı: 1, Toplam Karakter Karşılaştırma Sayısı: 0		TTATCCA	1
17 birim kadar desenleri sağa kaydır.		ATTATCC	2
		CATTATC	3
		TCATTAT	4
		CTCATTAT	5
		ACTCATTA	6
		GACTCATT	7
		GGACTCAT	8
		CGGACTCA	9
		TCGGACTC	10
		ATCGGACT	11
		CATCGGAC	12
		GCATCGGA	13
		TGCATCGG	14
		TTGCATCG	15
		CTTGCATC	16
		Diğerleri	17

Şekil 4.25. q -Gram WM algoritmasının başarısız dizi eşleştirme durumu.

Şekil 4.26 WM algoritmasındaki başarılı dizi eşleştirme durumunu göstermektedir. Sonek ve önek eşleştikten sonra önek tablosunun döndürdüğü (Dizi 1-2-3) dizilerinin karakterleri baştan sona kontrol edilir. Dizi 1 eşleştiği için Diziler sağa 1 birim kaydırılır ve 2 hash karşılaştırması ve 40 karakter karşılaştırması yapılır.

Önek CT, (1-2-3)	Sonek AT atlama değeri=0	Atlama Tablosu	Önek Tablosu
T= C T T G C A T C G G A C T C A T T A G T C C A T		Sonek Değeri	Önek Desen No
D1= C T T G C A T C G G A C T C A T T A T T C C A T		AA 6	CT 1,2,3
D2= C T T G C A T A T C T G A A T C C A T G C G A T		AC 0	
D3= C T T G C T T G A G G A C T G C A A C G T T A C		AG 14	
		AT 0	
		CA 6	
		CC 2	
		CG 1	
		CT 11	
		GA 1	
		GC 3	
		GG 13	
		GT 3	
		TA 1	
		TC 3	
		TG 4	
		TT 2	

Toplam Hash Karşılaştırma Sayısı: 2, Toplam Karakter Karşılaştırma Sayısı: 40
Desen 1 eşleştiği için desenleri 1 birim sağa kaydır.

Şekil 4.26. WM algoritmasının başarılı dizi eşleştirme durumu.

Şekil 4.27 önerilen q -gram WM algoritmasındaki başarılı dizi eşleşme durumunu göstermektedir. Sonek eşleştikten sonra önek eşleşmiştir ve ÖNEK tablosundan Dizi 1'i döndürmüştür. Sonek ve önek karakterlerinde çakışma olmadığından önekden sonra gelen ilk karakter ile metne denk gelen karakter kontrol edilir. Karakter uyumu sağlandığı için önekden sonra gelen q -gram'ların hash kontrolü yapılır. Sonuç olarak toplam 3 hash karşılaştırması ve 1 karakter karşılaştırması yapılarak karakter karşılaştırma sayısı minimuma indirgenmiştir.

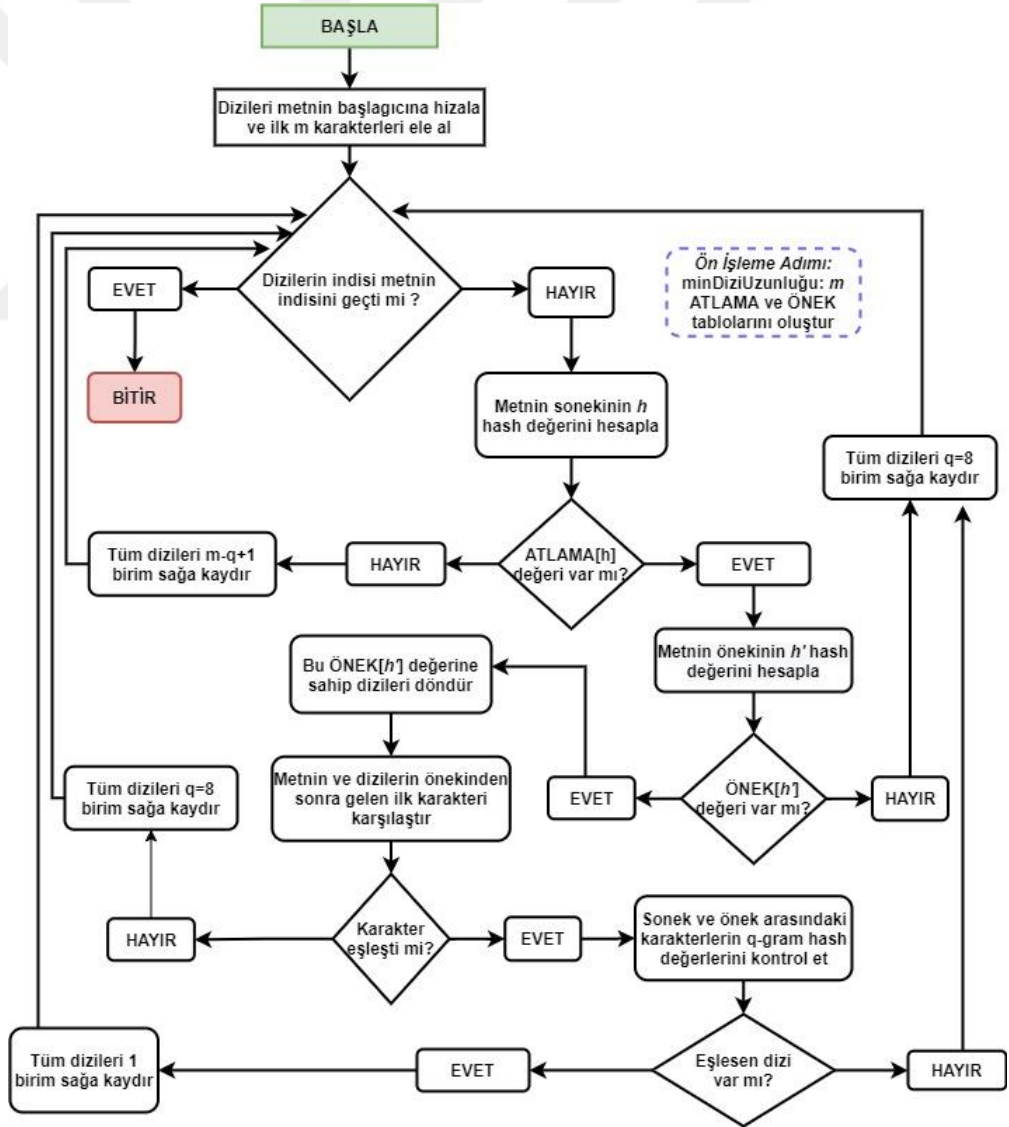
Önek CTTCATC, (1)	Sonek TAGTCCAT atlama değeri=0	Atlama Tablosu	Önek Tablosu
T= C T T G C A T C G G A C T C A T T A T T C C A T		Sonek Değeri	Önek Desen No
D1= C T T G C A T C G G A C T C A T T A T T C C A T		TATTCAT 0	CTTCATC 1
D2= C T T G C A T A T C T G A A T C C A T G C G A T		TTATCCA 1	CTTCATA 2
D3= C T T G C T T G A G G A C T G C A A C G T T A C		ATTATTC 2	CTTCATG 3
		CATTATC 3	
		TCATTAT 4	
		CTCATTAT 5	
		ACTCATTAT 6	
		GACTCATT 7	
		GGACTCAT 8	
		CGGACTCA 9	
		TCGGACTC 10	
		ATCGGACT 11	
		CATCGGAC 12	
		GCATCGGA 13	
		TGCATCGG 14	
		TTGCATCG 15	
		CTTCATC 16	
		Diğerleri 17	

Toplam Hash Karşılaştırma Sayısı: 3, Toplam Karakter Karşılaştırma Sayısı: 1
Desen 1 eşleştiği için desenleri 1 birim sağa kaydır.

Şekil 4.27. q -Gram WM algoritmasının başarılı dizi eşleştirme durumu.

4.4.3. Önerilen Yaklaşımın Genel Çerçevesi

Önerilen q -gram WM algoritmasının akış diyagramı Şekil 4.28'de gösterilmiştir. Ön işleme aşamasında öncelikle ATLAMA ve ÖNEK tabloları oluşturulur. Arama aşamasında, diziler metnin başlangıcına hizalandıktan sonra metnin q -uzunluklu sonekinin hash değeri kontrol edilir. Hash değerleri varsa, metnin q -uzunluklu önekinin hash değeri kontrol edilir. Eğer bu hash değeri de varsa bu öneke sahip diziler döndürülür. Sonra, önekten sonra gelen ilk karakter karşılaştırılır. Karakter eşleşirse sonek ve önek karakterleri arasında kalan karakterler soldan sağa q -gram hash karşılaştırmasıyla kontrol edilir. Herhangi bir eşleşme sağlanırsa, diziler metin üzerinde bir birim sağa kaydırılır; eşleşme sağlanmazsa, q birim sağa kaydırılır. Tüm adımlar metnin sonuna kadar devam eder.



Şekil 4.28. Önerilen q -Gram WM yaklaşımının genel çerçevesi.

4.4.4. Önerilen Yaklaşımın Çalışan Örneği

T girdi dizisi ve P aranan desenler kümesi olarak sırasıyla $T = TACTGGAATCGATATCTGCACGCGATTCTGACTAACGGTCTTCACCATT$ ve $P = \{D1 = TACTGGAATCGATATCTGCACGCG, D2 = ATTCGACTAACGGTCTTCACCATT\}$ dizi kümesi seçilmiştir. Her adımda hash karşılaştırma sayısı, karakter karşılaştırma sayısı ve eşleşen dizi gösterilmiştir. Başarısız hash ve karakter karşılaştırmaları açık kırmızı ile, başarılı hash karşılaştırmaları açık yeşil ile, başarılı karakter karşılaştırmaları ise koyu yeşil ile gösterilmiştir. Şekil 4.29'da WM algoritmasının, Şekil 4.30'da önerilen q -gram WM algoritmasının verilen diziler için ATLAMA ve ÖNEK tablolarını gösterilmiştir.

Atlama Tablosu			
Sonek	Değer	Sonek	Değer
AA	14	GA	12
AC	3	GC	1
AG	17	GG	11
AT	1	GT	10
CA	2	TA	10
CC	3	TC	6
CG	0	TG	6
CT	7	TT	0

Önek Tablosu	
Önek	Desen No
AT	1
TA	2

Şekil 4.29. WM algoritmasındaki sonek ve önek tabloları.

Atlama Tablosu			
Sonek	Değer	Sonek	Değer
TCACCATT	0	TGCACGCG	0
TTCACCAT	1	CTGCACGC	1
CTTCACCA	2	TCTGCACG	2
TCTTCACC	3	ATCTGCAC	3
GTCTTCAC	4	TATCTGCA	4
GGTCTTCA	5	ATATCTGC	5
CGGTCTTC	6	GATATCTG	6
ACGGTCTT	7	CGATATCT	7
AACGGTCT	8	TCGATATC	8
TAACGGTC	9	ATCGATAT	9
CTAACGGT	10	AATCGATA	10
ACTAACGG	11	GAATCGAT	11
GACTAACG	12	GGAATCGA	12
CGACTAAC	13	TGGAATCG	13
TCGACTAA	14	CTGGAATC	14
TTCGACTA	15	ACTGGAAT	15
ATTCGACT	16	TACTGGAA	16
Diğerleri	17		

Önek Tablosu	
Önek	Desen No
ATTCGACT	1
TACTGGAA	2

Şekil 4.30. q -Gram WM algoritmasındaki sonek ve önek tabloları.

Şekil 4.31 ve 4.32 ön işleme adımı sonucu oluşturulan tablolardaki bilgileri kullanarak geleneksel WM algoritmasında ve önerilen q -gram WM algoritması için girdi dizisinde aranacak dizilerin işleyişi gösterilmiştir. Şekil 4.32 WM algoritmasının işleyiş örneğini göstermektedir. Tüm dizilerin eşleştiği örnekte hash karşılaştırma sayısı 6, karakter karşılaştırma sayısı 48'dir. Şekil 4.33 önerilen q -gram WM algoritmasının işleyen örneğini göstermektedir. Tüm dizilerin eşleştiği örnekte hash karşılaştırma sayısı 8, karakter karşılaştırma sayısı 2'dir. Önerilen yaklaşımda, karakter karşılaştırması yapmadan dizi eşleştirme yapılmıştır. Çalışan örneklerde, önerilen q -gram WM algoritmasının WM algoritmasından daha iyi performans sonuçları elde ettiği gösterilmiştir.

Adım 1	Önek TA → Sonek CG=0 Desen 1 Bulundu Desenleri 1 birim kaydır (2 hash karşılaştırması) (24 karakter karşılaştırması) desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T
Adım 2	(1 hash karşılaştırması) Sonek GA=12 Desenleri 12 birim kaydır desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T
Adım 3	(1 hash karşılaştırması) Sonek GG=11 Desenleri 11 birim kaydır desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T
Adım 4	Desen 2 Bulundu Desenleri 1 birim kaydır (2 hash karşılaştırması) (24 karakter karşılaştırması) Önek AT → Sonek TT desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G
Adım 5	(metnin uzunluğu aşıldı) Toplam Karakter Karşılaştırma Sayısı: 48 Toplam Hash Karşılaştırma Sayısı: 6 desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G

Şekil 4.31. WM algoritmasının çalışan örneği.

Adım 1	Önek TACTGGAA → Sonek TGCACGCG=0 (1 karakter karşılaştırması) Desen 1 Bulundu Desenleri 1 birim kaydır (2 hash + 1 q-gram hash karşılaştırması) desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T
Adım 2	desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G Sonek GCACGCGA=17 Desenleri 17 birim kaydır (1 hash karşılaştırması)
Adım 3	Sonek CGGTCTTC=6 Desenleri 6 birim kaydır (1 hash karşılaştırması) desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T
Adım 4	(1 karakter karşılaştırması) Desenleri 1 birim kaydır Desen 2 Bulundu (2 hash + 1 q-gram hash karşılaştırması) Önek ATTCGACT → Sonek TCACCATT=0 desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G
Adım 5	(metnin uzunluğu aşıldı) Toplam Karakter Karşılaştırma Sayısı: 2 Toplam Hash Karşılaştırma Sayısı: 8 desen 2 = A T T C G A C T A A C G G T C T T C A C C A T T desen 1 = T A C T G G A A T C G A T A T C T G C A C G C G

Şekil 4.32. q-Gram WM algoritmasının çalışan örneği.

AC algoritmasının en iyi ve en kötü durum zaman karmaşıklığı $O(n)$ 'dir. Ancak AC algoritmanın geliştirilmiş bir versiyonu olan CW algoritmasının en iyi durum karmaşıklığı alt lineerdir. Hash tabanlı WM algoritmasının en iyi durum zaman karmaşıklığı $O((n \log_{|\Sigma|} |P|)/m)$ ve en kötü durum zaman karmaşıklığı $O(n|P| \log_{|\Sigma|} |P|)$ 'dir. Önerilen q -gram WM algoritmasında q uzunluklu sonek ve önek karakterlerinin karşılaştırması yapılmamaktadır. Sonuç olarak, d adet dizinin toplam uzunluğu $|P|$ olduğundan d adet dizinin sonek ve önek karakterleri karşılaştırılmadığı için toplam karşılaştırılacak karakter sayısı $|P - 2dq|$ 'dur. Bu nedenle önerilen q -gram WM algoritmasının en iyi durum zaman karmaşıklığı $O((n \log_{|\Sigma|} |P - 2dq|)/(m - 2q))$ ve en kötü durum zaman karmaşıklığı $O(n|P - 2dq| \log_{|\Sigma|} |P - 2dq|)$ 'dir. Bu çalışmada zaman karmaşıklığı WM algoritmasından daha verimli q -gram WM algoritması önerilmiştir.

Literatürde sıkça kullanılan çoklu dizi eşleştirme algoritmalarından AC algoritmasının en iyi ve en kötü durum zaman karmaşıklığı $O(n)$ 'dir. CW algoritması ise AC algoritmasının geliştirilmiş bir versiyonu olup zaman karmaşıklığı alt lineerdir. WM algoritmasının zaman karmaşıklığı $O(n|P| \log_{|\Sigma|} |P|)$ 'dir. Formüldeki $|P|$, d adet dizinin toplam uzunluğudur. Bu çalışmada önerilen q -gram WM algoritması mükemmel hash fonksiyonunu kullanarak sonek ve önek kontrolü yapmadan dizi eşleştirme işlemini gerçekleştirdiği için karşılaştırılacak karakter sayısı $|P - 2dq|$ 'dur. Sonuç olarak önerilen yaklaşımın zaman karmaşıklığı $O(n|P - 2dq| \log_{|\Sigma|} |P - 2dq|)$ olarak elde edilmiştir ve WM algoritmasından daha verimli olduğu gösterilmiştir.

5. DENEYSEL ÇALIŞMALAR

5.1. Hq-UF ve HqUF-BM Algoritmalarının Deneysel Çalışma Sonuçları

Deneysel çalışmalarda Çizelge 5.1'de gösterildiği gibi üç farklı veri seti kullanılmıştır. E. Coli (Durfee et al., 2008) ve Human Chromosome 1 (Gregory et al., 2004) veri setleri NCBI sitesinden metin dosyası formatında indirilmiştir. Tablo 4.8'de gösterildiği gibi, E. Coli veri setinde 4 milyondan fazla baz bulunurken, Human Chromosome1 veri setinde 230 milyondan fazla baz bulunmaktadır. Bu iki veri seti, küçük ve büyük ölçekli veri kümelerinde algoritmaları karşılaştırmak için seçilmiştir. “Sentetik Veri” dediğimiz veri seti dengeli dağılımla rastgele bilgisayar tarafından üretilmiştir.

Çizelge 5.1. HqUF ve HqUF-BM algoritmalarında kullanılan veri setleri.

Veri Seti	NCBI Referans No	Uzunluk (bp)
E. Coli	NC_010473.1	4,753,183
Sentetik	-	4,686,208
Human Chromosome1	NC_000001.11	248,956,422

Her bir algoritma için, veri setlerinden rastgele m uzunluklu diziler elde edilmiştir. Dizi uzunlukları 4, 8, 16, 32, 64, 128, 256, 512, 1024 olarak belirlenmiştir. Her bir dizi uzunluğu için rastgele 100 farklı dizi seçilmiş ve 100 kez çalıştırılmıştır. Ardından, ortalama çalışma süresi (milisaniye cinsinden), ortalama karakter karşılaştırma sayısı ve ortalama hash karşılaştırma sayısı elde edilmiştir. Hash fonksiyonumuz q 'nın tüm değerleri ($3 \leq q \leq 8$) için tüm veri setlerine uygulanmıştır ve sonuçlar aşağıdaki tablolarda listelenmiştir. Önerilen yaklaşımların sonuçları Hash- q algoritmasının sonuçlarıyla karşılaştırılmıştır. Her dizi uzunluğu için en iyi performansı veren q değeri seçilmiştir ve parantez içinde gösterilmiştir. Ayrıca, ortalama çalışma zamanındaki hızlanma yüzdeleri, ortalama karakter karşılaştırma sayısı ve ortalama hash karşılaştırma sayısı için performans artış yüzdeleri gösterilmiştir.

Algoritmalar C programlama dilinde derlenmiştir. Makine konfigürasyonu Intel Core i7-2670 QM CPU, 2.20 GHz, 6 GB RAM, Intel HD Graphics 3000 + Nvidia GeForce GT 555M, 750 GB 5400 RPM HDD'dir.

5.1.1. E. Coli Veri Seti Sonuçları

Çizelge 5.2, her dizi uzunluğundaki ortalama çalışma süresini gösterir. HqUF-BM algoritmasında ($m = 1024, q = 7$ için) Hash-q algoritmasına göre %83,11 hızlanma elde edilmiştir. Dizi uzunluğu 4 ile 128 arası olduğunda HqUF algoritmasında daha iyi sonuçlar elde edilmiştir. Ayrıca, dizinin uzunluğu arttıkça hızlanma yüzdeleri de artmıştır.

Çizelge 5.2. Ortalama çalışma süresi (e. coli).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	53.45(3)	42.68(3)	20.14%	44.22(3)	17.26%
8	18.52(3)	15.18(3)	18.03%	15.62(3)	15.65%
16	9.11(3)	7.04(3)	22.72%	7.09(3)	22.17%
32	4.84(4)	3.28(5)	32.23%	3.29(4)	32.02%
64	2.65(4)	1.09(5)	58.87%	1.70(6)	35.85%
128	1.60(4)	0.32(8)	80.00%	0.47(5)	70.62%
256	0.82(6)	0.16(7)	80.48%	0.15(6)	81.70%
512	0.70(5)	0.12(6)	82.85%	0.15(8)	78.57%
1024	0.77(8)	0.16(8)	79.22%	0.13(7)	83.11%

Çizelge 5.3, E. Coli veri kümesindeki ortalama karakter karşılaştırma sayısını göstermektedir. HqUF-BM algoritmasında ($m = 64, q = 6$ için) Hash-q algoritmasına göre %90.93 hızlanma elde edilmiştir. Önerilen yaklaşımların daha az karakter karşılaştırmasına sahip olduğu gösterilmiştir. Çizelge 5.4, E. Coli veri kümesi için ortalama hash karşılaştırma sayısını göstermektedir. HqUF algoritmasında ($m = 1024, q = 8$ için) Hash-q algoritmasına göre %97.52 hızlanma elde edilmiştir. Önerilen yaklaşımlar, tüm dizi uzunluklarında daha az hash karşılaştırma sayısı elde edilmiştir.

Çizelge 5.3. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (e. coli).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	117644.76(3)	45219.28(3)	61.56%	44954.94(3)	61.79%
8	33714.20(3)	26278.16(3)	22.06%	26266.00(3)	22.09%
16	13836.08(3)	11439.88(3)	17.32%	11439.45(3)	17.32%
32	2486.07(4)	411.48(5)	83.45%	1469.78(4)	40.88%
64	1573.63(4)	280.42(5)	82.18%	142.80(6)	90.93%
128	984.03(4)	147.99(8)	84.96%	244.85(5)	75.12%
256	512.28(6)	252.00(7)	50.81%	261.73(6)	48.91%
512	735.96(5)	515.26(6)	29.99%	507.57(8)	31.03%
1024	1346.81(8)	1138.95(8)	15.43%	1145.23(7)	14.97%

Çizelge 5.4. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (e. coli).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	2369242.89(3)	2360631.66(3)	0.36%	2346728.05(3)	0.95%
8	824362.51(3)	813171.23(3)	1.36%	813126.89(3)	1.36%
16	382784.91(3)	372572.89(3)	2.67%	372572.69(3)	2.67%
32	180653.97(4)	170158.84(5)	5.81%	172067.12(4)	4.75%
64	97237.21(4)	80872.87(5)	16.83%	80149.91(6)	17.57%
128	58027.98(4)	21380.73(8)	63.15%	40580.28(5)	30.07%
256	32300.13(6)	13999.71(7)	56.66%	18995.63(6)	41.19%
512	25021.59(5)	9072.05(6)	63.74%	1818.23(8)	92.73%
1024	20353.27(8)	505.41(8)	97.52%	1600.68(7)	92.14%

5.1.2. Sentetik Veri Seti Sonuçları

Çizelge 5.5, sentetik veri kümesindeki her bir dizi uzunluğundaki ortalama çalışma süresini göstermektedir. HqUF-BM algoritmasında ($m = 1024, q = 6$ için) Hash-q algoritmasına göre %87.17 hızlanma elde edilmiştir. Önerilen yaklaşımlar, tüm dizi uzunlukları için daha iyi ortalama çalışma zamanı sonuçları üretmiştir.

Çizelge 5.5. Ortalama çalışma süresi (sentetik veri).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	54.78(3)	43.49(3)	20.61%	44.62(3)	18.55%
8	19.28(3)	15.15(3)	21.42%	15.31(3)	20.59%
16	9.14(3)	7.03(3)	23.09%	6.87(3)	24.84%
32	4.52(3)	3.41(4)	24.56%	3.23(5)	28.54%
64	2.67(4)	0.62(5)	76.78%	1.55(7)	41.95%
128	1.72(6)	0.91(6)	47.09%	0.78(6)	54.65%
256	1.10(7)	0.47(7)	57.27%	0.32(6)	70.91%
512	1.08(6)	0.15(7)	86.11%	0.16(8)	85.19%
1024	0.78(4)	0.15(8)	80.77%	0.10(6)	87.17%

Çizelge 5.6, sentetik veri kümesindeki ortalama karakter karşılaştırma sayısını göstermektedir. HqUF-BM algoritmasında ($m = 32, q = 5$ için) Hash-q algoritmasına göre %94.78 hızlanma elde edilmiştir. Önerilen yaklaşımların daha az karakter karşılaştırmasına sahip olduğu gösterilmiştir. Çizelge 5.7, sentetik veri kümesi için ortalama hash karşılaştırma sayısını göstermektedir. HqUF-BM algoritmasında ($m = 1024, q = 8$ için) Hash-q algoritmasına göre %93.94 hızlanma elde edilmiştir. Önerilen yaklaşımlarda, tüm dizi uzunluklarında daha az hash karşılaştırma sayısı elde edilmiştir.

Çizelge 5.6. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (sentetik veri).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	121547.59(3)	45506.11(3)	62.56%	45274.55(3)	62.75%
8	30266.81(3)	22214.05(3)	26.61%	22210.81(3)	26.62%
16	13477.40(3)	10269.67(3)	23.80%	10268.52(3)	23.81%
32	7170.62(3)	1264.96(4)	82.36%	374.60(5)	94.78%
64	1426.52(4)	278.34(5)	80.49%	138.11(7)	90.32%
128	727.07(6)	315.38(6)	56.62%	315.16(6)	56.65%
256	862.60(7)	568.74(7)	34.07%	577.74(6)	33.03%
512	1390.68(6)	1193.36(7)	14.19%	1186.86(8)	14.66%
1024	2810.54(4)	2344.60(8)	16.58%	2359.31(6)	16.05%

Çizelge 5.7. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (sentetik veri).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	2369997.42(3)	2361486.13(3)	0.36%	2347018.98(3)	0.97%
8	823330.52(3)	811271.31(3)	1.46%	811245.96(3)	1.47%
16	384312.15(3)	370018.94(3)	3.72%	370018.86(3)	3.72%
32	209365.57(3)	170662.53(4)	18.49%	169589.89(5)	19.00%
64	96760.56(4)	80380.68(5)	16.93%	80258.05(7)	17.05%
128	51573.54(6)	38654.94(6)	25.05%	38655.98(6)	25.05%
256	32610.83(7)	15194.56(7)	53.41%	19081.20(6)	41.49%
512	24344.26(6)	6541.08(7)	73.13%	4571.15(8)	81.22%
1024	30395.88(4)	1841.27(8)	93.94%	4394.01(6)	85.54%

5.1.3. Human Chromosome 1 Veri Seti Sonuçları

Çizelge 5.8, her bir dizi uzunluğundaki ortalama çalışma sürelerini göstermektedir. HqUF algoritmasında ($m = 1024, q = 8$ için) Hash-q algoritmasına göre %72.85 hızlanma elde edilmiştir. Dizi uzunluğu $m = 32, 128, 256, 512$ olduğunda HqUF-BM algoritması ile daha iyi sonuçlar elde edilmiştir. Diğer dizi uzunluklarında HqUF algoritması daha iyi sonuçlar üretmiştir. Ayrıca dizinin uzunluğu arttıkça ($m = 256$ hariç) hızlanma yüzdeleri de artmıştır.

Çizelge 5.8. Ortalama çalışma süresi (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	2628.16(3)	2116.51(3)	19.47%	2159.13(3)	17.85%
8	926.99(3)	741.49(3)	20.01%	761.10(3)	17.90%
16	443.05(3)	346.78(3)	21.73%	351.39(4)	20.69%
32	226.11(4)	168.84(4)	25.33%	166.10(4)	26.54%
64	124.11(4)	84.14(5)	32.21%	84.87(5)	31.62%
128	80.16(4)	48.26(5)	39.80%	47.67(5)	40.53%
256	59.05(5)	41.62(5)	29.52%	40.46(5)	31.48%
512	55.14(5)	25.31(8)	54.10%	24.67(8)	55.26%
1024	47.81(5)	12.98(8)	72.85%	14.36(8)	69.96%

Çizelge 5.9, Human Chromosome 1 veri kümesindeki ortalama karakter karşılaştırma sayısını göstermektedir. HqUF-BM algoritmasında ($m = 512, q = 8$ için) Hash-q algoritmasına göre %94.95 hızlanma elde edilmiştir. Önerilen yaklaşımların daha az karakter karşılaştırmasına sahip olduğu gösterilmiştir. Çizelge 5.10, Human Chromosome 1 veri kümesi için ortalama hash karşılaştırma sayısını göstermektedir. HqUF algoritmasında ($m = 1024, q = 8$ için) Hash-q algoritmasına göre %78.03 hızlanma elde edilmiştir. Önerilen yaklaşımlarda, tüm dizi uzunluklarında daha az hash karşılaştırma sayısı elde edilmiştir.

Çizelge 5.9. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	5181509.47(3)	1936270.24(3)	62.63%	1923569.84(3)	62.88%
8	1538182.03(3)	1131480.42(3)	26.44%	1131117.96(3)	26.46%
16	846529.25(3)	632021.30(3)	25.34%	181167.59(4)	78.60%
32	144342.46(4)	102983.90(4)	28.65%	102983.81(4)	28.65%
64	76680.61(4)	12202.53(5)	84.09%	12202.53(5)	84.09%
128	42162.55(4)	7375.02(5)	82.51%	7374.89(5)	82.51%
256	15607.58(5)	3695.11(5)	76.32%	3695.06(5)	76.33%
512	11348.32(5)	573.91(8)	94.94%	573.65(8)	94.95%
1024	11943.34(5)	1095.81(8)	90.82%	1096.68(8)	90.82%

Çizelge 5.10. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	116350760.72(3)	115987223.92(3)	0.31%	115418558.17(3)	0.80%
8	40513577.46(3)	39940339.60(3)	1.41%	39938328.40(3)	1.42%
16	19096583.42(3)	18453491.05(3)	3.37%	18253684.74(4)	4.41%
32	8839059.01(4)	8480793.75(4)	4.05%	8480790.79(4)	4.05%
64	4702911.94(4)	3977846.89(5)	15.42%	3977846.92(5)	15.42%
128	2827672.60(4)	2004057.49(5)	29.13%	2004056.83(5)	29.13%
256	1635021.44(5)	1067112.19(5)	34.73%	1067112.56(5)	34.73%
512	1196853.60(5)	452438.45(8)	62.20%	452443.51(8)	62.20%
1024	1007867.96(5)	221411.73(8)	78.03%	221406.87(8)	78.03%

5.1.4. İyi Bilinen Algoritmalarla Deneysel Karşılaştırma

Literatürden hash tabanlı algoritma olan Skip-q, BMH-q ve Hash-q algoritmalarına ek olarak KK algoritması, BM ve BR algoritmaları olmak üzere 6 iyi bilinen algoritma seçilmiştir. Bu algoritmalar, önerilen iki yaklaşım olan HqUF ve HqUF-BM algoritmaları ile üç veri kümesi üzerinde ortalama çalışma süreleri açısından karşılaştırılmıştır. Hash tabanlı algoritmalar için en iyi q değeri sonuçların yanında parantez içinde verilmiştir.

5.1.4.1. E. Coli Veri Seti Sonuçları

Çizelge 5.11'de gösterildiği gibi, E. Coli veri seti için, $m = 4$ ve 8 dizi uzunlukları için, Skip-q algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF ve HqUF-BM algoritmaları ikinci ve üçüncü en iyi algoritmalar. Dizi uzunlukları, $m = 16, 32, 64, 128, 512$ için, önerilen HqUF algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF-BM algoritması, Skip-q'nun olduğu $m = 64$ dışında ikinci en iyi algoritmadır. Dizi uzunlukları, $m = 256$ ve 1024 için, HqUF-BM algoritması diğer algoritmalarından daha iyidir ve HqUF algoritması ikinci en iyi algoritmadır.

Çizelge 5.11. İyi bilinen algoritmalarla ortalama çalışma süreleri (e. coli).

	4	8	16	32	64	128	256	512	1024
BF	520.37	521.48	519.65	521.44	524.13	520.96	522.45	521.78	520.32
BM	68.65	35.52	28.43	12.56	9.65	8.85	7.67	7.11	6.81
BR	59.61	33.12	23.42	14.53	6.88	6.02	5.23	4.86	4.12
Skip-q	23.55(3)	13.74(4)	7.79(4)	3.89(4)	1.32(4)	1.08(6)	0.54(8)	0.32(8)	0.25(8)
BMH-q	49.15(4)	17.26(4)	9.56(4)	5.21(4)	2.92(4)	1.71(4)	0.94(4)	0.86(4)	0.81(4)
Hash-q	53.45(3)	18.52(3)	9.11(3)	4.84(4)	2.65(4)	1.60(4)	0.82(6)	0.70(5)	0.77(8)
HqUF	42.68(3)	15.18(3)	7.04(3)	3.28(5)	1.09(5)	0.32(8)	0.16(7)	0.12(6)	0.16(8)
HqUF-BM	44.22(3)	15.62(3)	7.09(3)	3.29(4)	1.70(6)	0.47(5)	0.15(6)	0.15(8)	0.13(7)

5.1.4.2. Sentetik Veri Seti Sonuçları

Çizelge 5.12'de gösterildiği gibi, sentetik veri seti için, $m = 4$ ve 8 dizi uzunlukları için, Skip-q algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF ve HqUF-BM algoritmaları ikinci ve üçüncü en iyi algoritmalar. Dizi uzunlukları, $m = 64$ ve 512 için, önerilen HqUF algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF-BM algoritması, Skip-q'nun olduğu $m = 64$ dışında üçüncü en iyi algoritmadır.

Dizi uzunlukları, $m = 16, 32, 128, 256$ ve 1024 için, HqUF-BM algoritması diğer algoritmalarından daha iyidir ve HqUF algoritması ikinci en iyi algoritmadır.

Çizelge 5.12. İyi bilinen algoritmalarda ortalama çalışma süreleri (sentetik).

	4	8	16	32	64	128	256	512	1024
BF	543.11	545.36	541.58	544.65	545.74	543.88	544.31	543.63	542.26
BM	69.34	37.12	29.06	11.68	9.72	9.38	8.71	8.32	7.75
BR	60.75	35.83	25.64	14.13	6.94	7.44	6.79	5.61	5.13
Skip-q	24.41(3)	13.85(4)	7.89(4)	3.83(4)	1.41(4)	1.12(6)	0.57(8)	0.34(8)	0.26(8)
BMH-q	51.21(4)	18.36(4)	9.86(4)	5.31(4)	3.08(4)	1.78(4)	0.96(4)	0.87(4)	0.81(4)
Hash-q	54.78(3)	19.28(3)	9.14(3)	4.52(3)	2.67(4)	1.72(6)	1.10(7)	1.08(6)	0.78(4)
HqUF	43.49(3)	15.15(3)	7.03(3)	3.41(4)	0.62(5)	0.91(6)	0.47(7)	0.15(7)	0.15(8)
HqUF-BM	44.62(3)	15.31(3)	6.87(3)	3.23(5)	1.55(7)	0.78(6)	0.32(6)	0.16(8)	0.10(6)

5.1.4.3. Human Chromosome 1 Veri Seti Sonuçları

Çizelge 5.13'te gösterildiği gibi, Human Chromosome 1 veri seti için, $m = 4$ ve 8 dizi uzunlukları için, Skip-q algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF ve HqUF-BM algoritmaları ikinci ve üçüncü en iyi algoritmalar. Dizi uzunlukları, $m = 16, 64$ ve 1024 için, önerilen HqUF algoritması diğer algoritmalarından daha iyi iken, önerilen HqUF-BM algoritması, Skip-q'nun olduğu $m=64$ dışında üçüncü en iyi algoritmadır. Dizi uzunlukları, $m = 32, 128, 256$ ve 512 için, HqUF-BM algoritması diğer algoritmalarından daha iyidir ve HqUF algoritması ikinci en iyi algoritmadır.

Çizelge 5.13. İyi bilinen algoritmalarda ortalama çalışma süreleri (human chromosome 1).

	4	8	16	32	64	128	256	512	1024
BF	27342.45	27645.52	26945.12	27632.56	28310.74	27486.31	27630.55	27261.69	26557.93
BM	4215.54	2461.87	1932.62	873.65	526.31	451.89	413.52	374.12	343.22
BR	3591.19	2181.55	135.74	723.65	348.62	312.42	295.63	273.67	248.53
Skip-q	1326.38(3)	687.59(4)	413.74(4)	217.91(4)	98.17(4)	74.56(6)	45.84(8)	28.23(8)	20.21(8)
BMH-q	2812.11(4)	902.56(4)	585.66(4)	362.85(4)	178.65(4)	89.95(4)	64.87(4)	60.12(4)	55.36(4)
Hash-q	2628.16(3)	926.99(3)	443.05(3)	226.11(4)	124.11(4)	80.16(4)	59.05(5)	55.14(5)	47.81(5)
HqUF	2116.51(3)	741.49(3)	346.78(3)	168.84(4)	84.14(5)	48.26(5)	41.62(5)	25.31(8)	12.98(8)
HqUF-BM	2159.13(3)	761.10(3)	351.39(4)	166.10(4)	84.87(5)	47.67(5)	40.46(5)	24.67(8)	14.36(8)

5.1.5. Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada, DNA dizilerindeki hash çakışmalarını ortadan kaldıran hash fonksiyonu ve Hash-q algoritmasına dayalı iki farklı yaklaşım önerilmiştir. HqUF algoritmasında, önerilen hash fonksiyonunu kullanarak, dizinin son q karakterini karşılaştırmaya gerek kalmamıştır. HqUF algoritmasına ek olarak, HqUF-BM algoritmasında başarılı bir arama için iyi sonek kaydırma tablosu dahil edilmiştir. Önerilen yaklaşımlarda, tüm q değerleri için ortalama çalışma zamanında, ortalama karakter karşılaştırma sayısında ve ortalama hash karşılaştırma sayısında daha iyi sonuçlar elde edilmiştir.

Hash-q algoritmasına kıyasla ortalama çalışma sürelerinde E. Coli veri setinde HqUF-BM algoritması ile %83.11 hızlanma, sentetik veri setinde HqUF-BM algoritması ile %87.17 hızlanma ve Human Chromosome1 veri setinde HqUF algoritması ile %72.5 hızlanma elde edilmiştir.

Ayrıca Hash-q algoritmasına kıyasla, ortalama karakter karşılaştırma sayılarında E. Coli veri setinde HqUF-BM algoritması ile %90.93, sentetik veri setinde HqUF-BM algoritması ile %94.78 iyileşme, Human Chromosome1 veri setinde HqUF-BM algoritması ile %94.95'lik bir iyileştirme elde edilmiştir. Ayrıca, dizi uzunluğu q değerine eşit olduğunda, ortalama karakter karşılaştırma sayısı sıfır (0) olarak elde edilmiştir.

Ek olarak, Hash-q algoritmasına kıyasla, ortalama hash karşılaştırma sayılarında, E. Coli veri setinde HqUF algoritması ile %97.52, sentetik veri setinde HqUF algoritması ile %93.94 ve HqUF algoritması ile Human Chromosome 1 veri setinde %78.03 iyileştirme elde edilmiştir.

Önerilen hash fonksiyonu sayesinde hem HqUF hem de HqUF-BM algoritmaları için Hash-q algoritmasına göre daha iyi performans sonuçları elde edilmiştir. Bu çalışmanın katkılarını şu şekilde özetleyebiliriz:

- Önerilen hash fonksiyonu sayesinde DNA dizilerindeki hash çakışmaları ortadan kaldırılmıştır.
- Dizinin ve metnin son q karakterini karşılaştırmaya gerek kalmamıştır.
- Dizi eşleştğinde iyi sonek tablosunun $(m - 1)$ 'inci girişindeki kaydırma boyutu kullanılarak performans iyileştirmesi sağlanmıştır.

Gelecekteki çalışmalarda önerilen hash fonksiyonuna dayalı yeni yaklaşımlar geliştirilebilir. Önerilen hash fonksiyonunun zaman açısından verimli hash değerleri üretir. Bu hash fonksiyonunu kullanarak, karakter karşılaştırmalarının sayısını azaltmak için q -çerçeve yaklaşımına dayalı hash tabanlı kesin dizi eşleştirme algoritmaları geliştirilebilir.

5.2. Hq-QF ve HqBM-QF Algoritmalarının Deneysel Çalışma Sonuçları

Deneysel çalışmalarda Çizelge 5.14'te gösterildiği gibi üç farklı veri seti kullanılmıştır. E. Coli (Durfee et al., 2008), Human Chromosome 1 (Gregory et al., 2004), Human Chromosome 2 (Hillier et al., 2005) ve Human Chromosome 3 (Muzny et al., 2006) veri setleri NCBI sitesinden metin dosyası formatında indirilmiştir. Human Chromosome 2 ve Human Chromosome 3 veri setleri birleştirilerek daha büyük bir veri seti elde etmek için Human Chromosome 2+3 oluşturulmuştur. Çizelge 5.14'te gösterildiği gibi, E. Coli veri setinde 4 milyondan fazla baz bulunurken, Human Chromosome 1 ve 2'de 243 milyondan fazla baz ve Human Chromosome 2+3 veri setlerinde 430 milyondan fazla baz bulunmaktadır.

Çizelge 5.14. Hq-QF ve HqBM-QF algoritmalarında kullanılan veri setleri.

Veri Setleri	NCBI Referans No	Uzunluk
E. Coli	NC_010473.1	4,753,183
Human Chromosome 1	NC_000001.11	248,956,422
Human Chromosome 2	NC_000002.12	243,199,373
Human Chromosome 3	NC_000003.12	198,295,559
Human Chromosome 2+3	-	441,494,932

Her bir algoritma için, veri setlerinden rastgele m uzunluklu diziler elde edilmiştir. Dizi uzunlukları 4, 8, 16, 32, 64, 128, 256, 512, 1024 olarak belirlenmiştir. Her bir dizi uzunluğu için rastgele 100 farklı dizi seçilmiş ve 100 kez çalıştırılmıştır. Ortalama çalışma süresi (milisaniye cinsinden), ortalama karakter karşılaştırma sayısı ve ortalama hash karşılaştırma sayısı elde edilmiştir. Ayrıca, performans metriklerinin hızlanma yüzdeleri tablolarda gösterilmiştir. Algoritmalar C programlama dilinde derlenmiştir. 64 bit işlemci üzerinde Intel Core i7-2670 QM CPU, 2.20 GHz, 6 GB RAM, Intel HD Graphics 3000 + Nvidia GeForce GT 555M, 750 GB 5400 RPM HDD özellikli bilgisayarda sonuçlar elde edilmiştir. Önerilen yaklaşımlar Hash-q, HqUF ve HqUF-BM algoritmaları ile karşılaştırılmıştır. Her dizi uzunluğu için en iyi performansa sahip q değeri parantez içinde gösterilmiştir. Hash-q algoritmasına göre performans artış yüzdeleri de gösterilmiştir.

5.2.1. E. Coli Veri Seti Sonuçları

Çizelge 5.15, E. Coli veri setindeki her bir algoritmanın ortalama çalışma süresini göstermektedir. Hash-q algoritmasına kıyasla Hq-QF algoritmasında ($m = 1024, q = 8$ için) %84.41'e varan hızlanma sağlanmıştır. $m = 4,128,256,512,1024$ için Hq-QF algoritması daha iyi sonuçlar üretmiştir. HqBM-QF algoritması diğer algoritmalarla göre $m = 8,16,32$ için daha iyi sonuçlar üretmiştir. Bu nedenle, önerilen yaklaşımlar, E. Coli veri seti için ortalama çalışma süresinde daha verimli olduğu gösterilmiştir.

Çizelge 5.15. Ortalama çalışma süreleri (e. coli).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	53.45(3)	42.68(3)	20.14%	44.22(3)	17.26%	42.38(3)	20.71%	44.83(3)	16.12%
8	18.52(3)	15.18(3)	18.03%	15.62(3)	15.65%	15.47(3)	16.46%	15.02(3)	18.89%
16	9.11(3)	7.04(3)	22.72%	7.09(3)	22.17%	7.20(3)	20.96%	6.88(4)	24.48%
32	4.84(4)	3.28(5)	32.23%	3.29(4)	32.02%	3.26(5)	32.64%	3.12(4)	35.53%
64	2.65(4)	1.09(5)	58.87%	1.70(6)	35.85%	1.70(6)	35.85%	1.56(4)	41.13%
128	1.60(4)	0.32(8)	80.00%	0.47(5)	70.62%	0.28(8)	82.50%	0.47(8)	70.62%
256	0.82(6)	0.16(7)	80.48%	0.15(6)	81.70%	0.15(7)	81.70%	0.47(8)	42.68%
512	0.70(5)	0.12(6)	82.85%	0.15(8)	78.57%	0.11(7)	84.28%	0.14(7)	80%
1024	0.77(8)	0.16(8)	79.22%	0.13(7)	83.11%	0.12(8)	84.41%	0.16(5)	79.22%

Çizelge 5.16, E. Coli veri setindeki ortalama karakter karşılaştırma sayısını göstermektedir. Önerilen yaklaşımlarda Hash-q algoritmasına kıyasla %100'e varan iyileştirme sağlanmıştır. Dizi uzunluğu q değerinin katı olduğunda, ortalama karakter karşılaştırma sayısı sıfır (0) olarak elde edilmiştir. Bu nedenle, önerilen yaklaşımlarda daha az karakter karşılaştırması yapıldığı gösterilmiştir.

Çizelge 5.16. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (e. coli).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	117644,76(3)	45219.28(3)	61.56%	44954.94(3)	61.79%	45219.28(3)	61.56%	44954.94(3)	61.79%
8	33714,2(3)	26278.16(3)	22.06%	26266(3)	22.09%	1616.85(3)	95.20%	1615.88(3)	95.20%
16	13836,08(3)	11439.88(3)	17.32%	11439.45(3)	17.32%	1.26(3)	99.99%	0(4)	100%
32	2486,07(4)	411.48(5)	83.45%	1469.78(4)	40.88%	2.28(5)	99.90%	0(4)	100%
64	1573,63(4)	280.42(5)	82.18%	142.8(6)	90.93%	6.61(6)	99.58%	0(4)	100%
128	984,03(4)	147.99(8)	84.96%	244.85(5)	75.12%	0(8)	100%	0(8)	100%
256	512,28(6)	252(7)	50.81%	261.73(6)	48.91%	4(7)	99.22%	0(8)	100%
512	735,96(5)	515.26(6)	29.99%	507.57(8)	31.03%	1(7)	99.86%	1(7)	99.86%
1024	1346,81(8)	1138.95(8)	15.43%	1145.23(7)	14.97%	0(8)	100%	4.48(5)	99.66%

Çizelge 5.17, E. Coli veri setindeki ortalama hash karşılaştırma sayısını göstermektedir. Hq-QF algoritmasında ($m = 1024, q = 8$ için) %96.80'e varan iyileştirme sağlanmıştır ve önerilen yaklaşımlarda Hash-q algoritmasına kıyasla tüm dizi uzunluklarında hash karşılaştırma sayısı azalmıştır. HqBM-QF algoritması $m = 4$ ve 256 için daha iyi sonuç üretmiştir. Önerilen yaklaşımlar, E. Coli veri setinde, hash karşılaştırma sayısını artırarak karakter karşılaştırma sayısını azaltmıştır.

Çizelge 5.17. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (e. coli).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	2369242.89(3)	2360631.66(3)	0.36%	2346728.05(3)	0.95%	2360631.66(3)	0.36%	2346728.05(3)	0.95%
8	824362.51(3)	813171.23(3)	1.36%	813126.89(3)	1.36%	814440.82(3)	1.20%	831912.49(3)	-0.91%
16	382784.91(3)	372572.89(3)	2.67%	372572.69(3)	2.67%	373136.69(3)	2.52%	372644.22(4)	2.65%
32	180653.97(4)	170158.84(5)	5.81%	172067.12(4)	4.75%	170450.54(5)	5.64%	173160.42(4)	4.15%
64	97237.21(4)	80872.87(5)	16.83%	80149.91(6)	17.57%	80165.84(6)	17.55%	87997.58(4)	9.5%
128	58027.98(4)	21380.73(8)	63.15%	40580.28(5)	30.07%	21400.21(8)	63.12%	21393.71(8)	63.13%
256	32300.13(6)	13999.71(7)	56.66%	18995.63(6)	41.19%	14037.05(7)	56.54%	6473.07(8)	79.96%
512	25021.59(5)	9072.05(6)	63.74%	1818.23(8)	92.73%	5012.95(7)	79.97%	5007.26(7)	79.98%
1024	20353.27(8)	505.41(8)	97.52%	1600.68(7)	92.14%	651.22(8)	96.80%	7718.46(5)	62.08%

5.2.2. Human Chromosome 1 Veri Seti Sonuçları

Çizelge 5.18, Human Chromosome 1 veri setindeki her bir algoritmanın ortalama çalışma süresini göstermektedir. Hash-q algoritmasına kıyasla Hq-QF algoritmasında ($m = 1024, q = 8$ için) %71.86'ya varan hızlanma sağlanmıştır. $m = 4$ ve 64 için Hq-QF algoritması daha iyi sonuçlar üretmiştir. HqBM-QF algoritması $m = 32, 128, 256, 512$ için diğer algoritmalarından daha iyi sonuçlar üretmiştir. Bu nedenle, önerilen

yaklaşımlar, Human Chromosome 1 için ortalama çalışma süresi için daha verimli olduğu gösterilmiştir. Önerilen yaklaşımlarda çoğu dizi uzunluğunda daha iyi sonuçlar elde edilmiştir. Diğer sonuçlar da en iyi sonuçlara yakındır. Ek olarak, önerilen yaklaşımların zaman karmaşıklığı diğerlerinden daha verimli olmasına rağmen, sonuçlar dizilerin ve metnin içeriğine bağlıdır.

Çizelge 5.18. Ortalama çalışma süreleri (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	2628.16(3)	2116.51(3)	19.47%	2159.13(3)	17.85%	2103.22(3)	19.97%	2171.19(3)	17.39%
8	926.99(3)	741.49(3)	20.01%	761.10(3)	17.90%	752.69(3)	18.80%	767.26(3)	17.17%
16	443.05(3)	346.78(3)	21.73%	351.39(4)	20.69%	352.6(3)	20.54%	351.17(4)	20.74%
32	226.11(4)	168.84(4)	25.33%	168.08(4)	25.66%	170.73(4)	24.49%	166.10(4)	26.54%
64	124.11(4)	84.14(5)	32.21%	84.87(5)	31.62%	83.36(5)	32.83%	84.55(5)	31.87%
128	80.16(4)	48.26(5)	39.80%	47.67(5)	40.53%	48.20 (5)	40%	47.37(5)	40.90%
256	59.05(5)	41.62(5)	29.52%	40.69(5)	31.09%	41.05(5)	30.48%	40.46(5)	31.48%
512	55.14(5)	25.31(8)	54.10%	24.67(8)	55.26%	24.84(7)	54.95%	24.37(7)	55.80%
1024	47.81(5)	12.98(8)	72.85%	14.36(8)	69.96%	13.45(8)	71.86%	14.17(8)	70.36%

Tablo 5.19, Human Chromosome 1 veri setindeki ortalama karakter karşılaştırma sayısını göstermektedir. E. Coli veri seti sonuçlarında olduğu gibi, önerilen yaklaşımlarda da Hash-q algoritmasına kıyasla %100'e varan iyileştirme sağlanmıştır. Dizi uzunluğu, q uzunluğunun katlarına eşit olduğunda, ortalama karakter karşılaştırma sayısı sıfır (0) olarak elde edilmiştir. Ayrıca önerilen yaklaşımlarda karakter karşılaştırma sayısı en aza indirildiği için tüm q değerlerinde iyileştirmeler sağlanmıştır.

Çizelge 5.19. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	5181509.47(3)	1936270.24(3)	62.63%	1923569.84(3)	62.87%	1936270.24(3)	62.63%	1923569.84(3)	62.87%
8	1538182.03(3)	1131480.42(3)	26.44%	1131117.96(3)	26.46%	62424.59(3)	95.94%	62206.49(3)	95.95%
16	846529.25(3)	632021.30(3)	25.34%	181167.59(4)	78.60%	19.75(3)	99.99%	0(4)	100%
32	144342.46(4)	102983.90(4)	28.65%	102983.81(4)	28.65%	0(4)	100%	0(4)	100%
64	76680.61(4)	12202.53(5)	84.09%	12202.53(5)	84.09%	4.2(5)	99.83%	4.2(5)	99.83%
128	42162.55(4)	7375.02(5)	82.51%	7374.89(5)	82.51%	3(5)	99.99%	3(5)	99.99%
256	15607.58(5)	3695.11(5)	76.32%	3695.06(5)	76.33%	1(5)	99.99%	1(5)	99.99%
512	11348.32(5)	573.91(8)	94.94%	573.65(8)	94.95%	1(7)	99.99%	1(7)	99.99%
1024	11943.34(5)	1095.81(8)	90.82%	1096.68(8)	90.82%	0(8)	100%	0(8)	100%

Tablo 5.20, Human Chromosome 1 veri setindeki ortalama hash karşılaştırma sayısını göstermektedir. HqBM-QF algoritmasında ($m = 1024$, $q = 8$ için) Hash-q algoritmasına kıyasla %78.01'e varan iyileştirme sağlanmıştır. HqUF ve HqUF-BM algoritmaları tüm durumlar için daha iyi sonuçlar üretmiştir.

Çizelge 5.20. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 1).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	116350760.72(3)	115987223.92(3)	0.31%	115418558.17(3)	0.80%	115987223.92(3)	0.31%	115418558.17(3)	0.8%
8	40513577.46(3)	39940339.60(3)	1.41%	39938328.40(3)	1.42%	39990501.29(3)	1.29%	407693548.94(3)	-0.63%
16	19096583.42(3)	18453491.05(3)	3.37%	18253684.74(4)	4.41%	18488748.87(3)	3.18%	18382336.65(4)	3.74%
32	8839059.01(4)	8480793.75(4)	4.05%	8480790.79(4)	4.05%	8554462.01(4)	3.21%	8554458.28(4)	3.22%
64	4702911.94(4)	3977846.89(5)	15.42%	3977846.92(5)	15.42%	3986348.5(5)	15.23%	3986348.49(5)	15.23%
128	2827672.60(4)	2004057.49(5)	29.13%	2004056.83(5)	29.13%	2009316.64(5)	28.94%	2009315.97(5)	28.94%
256	1635021.44(5)	1067112.19(5)	34.73%	1067112.56(5)	34.73%	1069864.86(5)	34.56%	1069865.01(5)	34.56%
512	1196853.60(5)	452438.45(8)	62.20%	452443.51(8)	62.20%	468145.94(7)	60.88%	468146.16(7)	60.88%
1024	1007867.96(5)	221411.73(8)	78.03%	221406.87(8)	78.03%	221610.31(8)	78.01%	221605.23(8)	78.01%

5.2.3. E. Human Chromosome 2+3 Veri Seti Sonuçları

Çizelge 5.21, Human Chromosome 2+3 veri setindeki her bir algoritmanın ortalama çalışma süresini göstermektedir. Hash-q algoritmasına kıyasla Hq-QF algoritmasında ($m = 1024, q = 8$ için) %71.43'e varan hızlanma sağlanmıştır. Hq-QF algoritması $m = 4,64,1024$ için daha iyi sonuçlar üretmiştir. HqBM-QF algoritması $m = 16,128,256$ için diğer algoritmalara göre daha iyi sonuçlar üretmiştir. Bu nedenle, önerilen yaklaşımlar, Human Chromosome 2+3 için ortalama çalışma süresi için daha verimli olduğu gösterilmiştir.

Çizelge 5.21. Ortalama çalışma süreleri (human chromosome 2+3).

M	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)	Ortalama Çalışma Zamanı	Hızlanma (%)
4	2628.16(3)	2116.51(3)	19.47%	2159.13(3)	17.85%	2103.22(3)	19.97%	2171.19(3)	17.39%
8	926.99(3)	741.49(3)	20.01%	761.10(3)	17.90%	752.69(3)	18.80%	767.26(3)	17.17%
16	443.05(3)	346.78(3)	21.73%	351.39(4)	20.69%	352.6(3)	20.54%	351.17(4)	20.74%
32	226.11(4)	168.84(4)	25.33%	168.08(4)	25.66%	170.73(4)	24.49%	166.10(4)	26.54%
64	124.11(4)	84.14(5)	32.21%	84.87(5)	31.62%	83.36(5)	32.83%	84.55(5)	31.87%
128	80.16(4)	48.26(5)	39.80%	47.67(5)	40.53%	48.20 (5)	40%	47.37(5)	40.90%
256	59.05(5)	41.62(5)	29.52%	40.69(5)	31.09%	41.05(5)	30.48%	40.46(5)	31.48%
512	55.14(5)	25.31(8)	54.10%	24.67(8)	55.26%	24.84(7)	54.95%	24.37(7)	55.80%
1024	47.81(5)	12.98(8)	72.85%	14.36(8)	69.96%	13.45(8)	71.86%	14.17(8)	70.36%

Çizelge 5.22, Human Chromosome 1 veri setindeki ortalama karakter karşılaştırma sayısını göstermektedir. E. Coli ve Human Chromosome 1 veri setleri sonuçlarında olduğu gibi, önerilen yaklaşımlarda da Hash-q algoritmasına kıyasla %100'e varan iyileştirme sağlanmıştır. Dizi uzunluğu, q uzunluğunun katlarına eşit olduğunda, ortalama karakter karşılaştırma sayısı sıfır (0) olarak elde edilmiştir. Ayrıca önerilen yaklaşımlarda karakter karşılaştırma sayısı en aza indirildiği için tüm q değerlerinde iyileştirmeler sağlanmıştır.

Çizelge 5.23, Human Chromosome 1 veri setindeki ortalama hash karşılaştırma sayısını göstermektedir. Hq-QF algoritmasında ($m = 1024$, $q = 8$ için) Hash-q algoritmasına kıyasla %84.13'e varan iyileştirme sağlanmıştır. HqUF ve HqUF-BM algoritmaları tüm durumlar için daha iyi sonuçlar üretmiştir.

Çizelge 5.22. Ortalama çalışma süresine karşılık gelen ortalama karakter karşılaştırma sayısı (human chromosome 2+3).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)	Ortalama Karakter Karşılaştırma Sayısı	Azalma (%)
4	14559504.01(3)	4371674.01(3)	69,97%	4327937.47(3)	70,27%	4371674.01(3)	69,97%	4327937.47(3)	70,27%
8	3248268.86(3)	2292475.19(3)	29,42%	2290374.55(3)	29,48%	87635.64(3)	97,30%	163158.60(3)	94,97%
16	1602697.13(3)	1227682.08(3)	23,39%	1227679.60(3)	23,39%	0(4)	100%	297.10(3)	99,98%
32	899436.70(3)	171944.53(4)	80,88%	51339.35(5)	94,29%	0(4)	100%	0(4)	100%
64	143849.74(4)	25150.38(5)	82,51%	3176.33(7)	97,79%	5.80(5)	99,9%	5.80(5)	99,99%
128	34613.47(6)	3188.92(6)	90,78%	3188.94(6)	90,78%	3.48(5)	99,98%	3.48(5)	99,98%
256	22516.82(7)	782.45(7)	96,52%	1925.00(6)	91,45%	1.08(5)	99,9%	4.32(6)	99,98%
512	16895.52(6)	582.89(8)	96,55%	562.19(8)	96,67%	1(7)	99,99%	1(7)	99,99%
1024	43711.34(4)	1035.48 (8)	97,63%	1450.12(6)	96,68%	0(8)	100%	0(8)	100%

Çizelge 5.23. Ortalama çalışma süresine karşılık gelen ortalama hash karşılaştırma sayısı (human chromosome 2+3).

m	Hash-q	HqUF		HqUF-BM		Hq-QF		HqBM-QF	
		Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)	Ortalama Hash Karşılaştırma Sayısı	Azalma (%)
4	222152507.20(3)	220991625.40(3)	0,52%	219688218.70(3)	1,10%	220991625.40(3)	0,52%	219688218.70(3)	1,10%
8	77548894.25(3)	76359001.24(3)	1,53%	76352237.79(3)	1,54%	76359001.24(3)	1,53%	76352237.79(3)	1,54%
16	36227988.02(3)	35163246.65(3)	2,93%	35163227.45(3)	2,93%	34736598.35(4)	4,11%	35163227.45(3)	2,93%
32	19889920.53(3)	16208094.3(4)	18,51%	15960727.94(5)	19,75%	16208094.30(4)	18,51%	16208092.97(4)	18,51%
64	9013415.43(4)	7616629.50(5)	15,49%	7588079.90(7)	15,81%	7616629.50(5)	15,49%	7616629.40(5)	15,49%
128	4656621.15(6)	3656184.69(6)	21,48%	3656184.78(6)	21,48%	3852979.78(5)	17,25%	3852979.72(5)	17,25%
256	2886598.67(7)	1780034.78(7)	38,33%	1835516.27(6)	36,41%	2054285.08(5)	28,83%	1835516.27(6)	36,41%
512	2137742.22(6)	875714.87(8)	59,03%	875716.09(8)	59,03%	891113.86(7)	58,31%	1780034.87(7)	16,73%
1024	2807751.05(4)	445457.47 (8)	84,13%	524142.69(6)	81,33%	445457.47(8)	84,13%	445458.17(8)	84,13%

5.2.4. İyi Bilinen Algoritmalarla Deneysel Karşılaştırmalar

Geliştirilen algoritmaları karşılaştırmak için literatürden sekiz farklı kesin dizi eşleştirme algoritması seçilmiştir. Seçilen algoritmalar KK, BM ve BR algoritmaları ve hash tabanlı algoritmalar: Hash-q, BMH-q, Skip-q, HqUF HqUF-BM algoritmalarıdır. Algoritmaların ortalama çalışma süreleri cinsinden sonuçları karşılaştırılmıştır. Hash tabanlı algoritmaların sonuçlarını veren en iyi q değeri, sonuçların yanında parantez içinde verilmiştir. Seçilen algoritmalar, önerilen Hq-QF ve HqBM-QF algoritmaları ile karşılaştırılmıştır.

5.2.4.1. E. Coli Veri Seti Sonuçları

Çizelge 5.24'te gösterildiği gibi E. Coli veri seti için en iyi sonuçlar $m = 4,8$ için Skip-q algoritması ile elde edilmiştir. İkinci en iyi sonuçlar, $m = 4$ için Hq-UF algoritması ve $m = 8$ için HqBM-QF algoritması ile elde edilmiştir. Ayrıca $m = 16$ için HqBM-UF algoritması ile ikinci en iyi sonuç elde edilmiştir. Hq-QF algoritmasında en iyi sonuçlar $m = 128,256,512$ için elde edilmiştir. HqBM-QF algoritmasında ise en iyi sonuçlar $m = 16,32,1024$ için elde edilmiştir. Sonuç olarak, E. Coli veri seti için önerilen yaklaşımların iyi bilinen algoritmalarından daha iyi sonuçlar elde ettiği gösterilmiştir.

Çizelge 5.24. Ortalama çalışma süresi (e. coli).

	4	8	16	32	64	128	256	512	1024
BF	520.37	521.48	519.65	521.44	524.13	520.96	522.45	521.78	520.32
BM	68.65	35.52	28.43	12.56	9.65	8.85	7.67	7.11	6.81
BR	59.61	33.12	23.42	14.53	6.88	6.02	5.23	4.86	4.12
Skip-q	23.55(3)	13.74(4)	7.79(4)	3.89(4)	1.32(4)	1.08(6)	0.54(8)	0.32(8)	0.25(8)
BMH-q	49.15(4)	17.26(4)	9.56(4)	5.21(4)	2.92(4)	1.71(4)	0.94(4)	0.86(4)	0.81(4)
Hash-q	53.45(3)	18.52(3)	9.11(3)	4.84(4)	2.65(4)	1.60(4)	0.82(6)	0.70(5)	0.77(8)
HqUF	42.68(3)	15.18(3)	7.04(3)	3.28(5)	1.09(5)	0.32(8)	0.16(7)	0.12(6)	0.16(8)
HqUF-BM	44.22(3)	15.62(3)	7.09(3)	3.29(4)	1.70(6)	0.47(5)	0.15(6)	0.15(8)	0.13(7)
Hq-QF	42.38(3)	15.47(3)	7.20(3)	3.26(5)	1.70(6)	0.28(8)	0.15(7)	0.11(7)	0.16(5)
HqBM-QF	44.83(3)	15.02(3)	6.88(4)	3.12(4)	1.56(4)	0.47(8)	0.47(8)	0.14(7)	0.12(8)

5.2.4.2. Human Chromosome 1 Veri Seti Sonuçları

Çizelge 5.25'te gösterildiği gibi Human Chromosome 1 veri seti için en iyi sonuçlar $m = 4,8$ için Skip-q algoritması ile elde edilmiştir. İkinci en iyi sonuç $m = 4$ için, üçüncü en iyi sonuç ise $m = 8$ için Hq-UF algoritması ile

elde edilmiştir. Ayrıca $m = 16$ için HqBM-UF algoritması ile ikinci en iyi sonuç elde edilmiştir. En iyi sonuçlar $m = 64$ için Hq-QF algoritmasında elde edilmiştir. Yine en iyi sonuçlar $m = 32, 128, 256, 512$ için HqBM-QF algoritmasında elde edilmiştir. Sonuç olarak, Human Chromosome 1 veri setinde önerilen yaklaşımlar iyi bilinen algoritmalarından daha iyi sonuçlar üretmiştir.

Çizelge 5.25. Ortalama çalışma süresi (human chromosome 1).

	4	8	16	32	64	128	256	512	1024
BF	27342.45	27645.52	26945.12	27632.56	28310.74	27486.31	27630.55	27261.69	26557.93
BM	4215.54	2461.87	1932.62	873.65	526.31	451.89	413.52	374.12	343.22
BR	3591.19	2181.55	135.74	723.65	348.62	312.42	295.63	273.67	248.53
Skip-q	1326.38(3)	687.59(4)	413.74(4)	217.91(4)	98.17(4)	74.56(6)	45.84(8)	28.23(8)	20.21(8)
BMH-q	2812.11(4)	902.56(4)	585.66(4)	362.85(4)	178.65(4)	89.95(4)	64.87(4)	60.12(4)	55.36(4)
Hash-q	2628.16(3)	926.99(3)	443.05(3)	226.11(4)	124.11(4)	80.16(4)	59.05(5)	55.14(5)	47.81(5)
HqUF	2116.51(3)	741.49(3)	346.78(3)	168.84(4)	84.14(5)	48.26(5)	41.62(5)	25.31(8)	12.98(8)
HqUF-BM	2159.13(3)	761.10(3)	351.39(4)	168.08(4)	84.87(5)	47.67(5)	40.69(5)	24.67(8)	14.36(8)
Hq-QF	2103.22(3)	752.69(3)	352.6(3)	170.73(4)	83.36(5)	48.20 (5)	41.05(5)	24.84(7)	13.45(8)
HqBM-QF	2171.19(3)	767.26(3)	351.17(4)	166.10(4)	84.55(5)	47.37(5)	40.46(5)	24.37(7)	14.17(8)

5.2.4.3. Human Chromosome 2+3 Veri Seti Sonuçları

Çizelge 5.26'da gösterildiği gibi, Human Chromosome 2+3 veri seti için en iyi sonuçlar $m = 4, 8$ için Skip-q algoritması ile elde edilmiştir. İkinci en iyi sonuçlar $m = 4, 8$ için ve $m = 16$ için ikinci en iyi sonuç Hq-UF algoritması ile elde edilmiştir. En iyi sonuçlar $m = 64$ için Hq-QF algoritmasında elde edilmiştir. Yine en iyi sonuçlar $m = 32, 128, 256, 512$ için HqBM-QF algoritmasında elde edilmiştir. Human Chromosome 2+3 veri setinde önerilen yaklaşımlar iyi bilinen algoritmalarından daha iyi sonuçlar üretmiştir.

Çizelge 5.26. Ortalama çalışma süresi (human chromosome 2+3).

	4	8	16	32	64	128	256	512	1024
BF	52123.14	52205.13	51321.25	53159.62	55421.45	54062.31	54632.14	54121.47	52326.56
BM	8231.22	4632.44	3865.32	1602.75	976.46	852.23	805.36	712.49	661.87
BR	6916.54	4086.52	241.78	1364.58	641.12	603.78	548.62	525.39	427.85
Skip-q	2548.55(3)	1321.22(4)	803.61(4)	434.22(4)	200.21(4)	148.26(6)	89.79(8)	54.78(8)	39.11(8)
BMH-q	5412.33(4)	1781.26(4)	1026.32(4)	712.63(4)	326.52(4)	181.21(4)	126.23(4)	119.56(4)	108.77(4)
Hash-q	5129.51(3)	1822.11(3)	871.95(3)	440.26(4)	237.46(4)	162.45(4)	119.67(5)	109.78(5)	93.68(5)
HqUF	4130.78(3)	1441.45(3)	651.48(3)	319.25(4)	317.69(5)	84.64(5)	80.56(5)	48.85(8)	23.54(8)
HqUF-BM	4098.89(3)	1485.21(3)	662.47(4)	318.22(4)	317.45(5)	83.77(5)	78.95(5)	46.98(8)	27.18(8)
Hq-QF	4020.56(3)	1462.39(3)	660.18(3)	325.87(4)	312.25(5)	84.87 (5)	80.25(5)	46.79(7)	25.55(8)
HqBM-QF	4112.33(3)	1487.45(3)	663.54(4)	312.51(4)	317.27(5)	82.66(5)	78.05(5)	46.11(7)	27.47(8)

5.2.5. Sonuçlar

Bu çalışmada, DNA dizilerinde mükemmel bir hash fonksiyonuna dayanan iki algoritma önerilmiştir. Mükemmel hash, q -çerçeve hash karşılaştırmalarının uygulanmasını sağlamıştır. Önerilen algoritmalar ortalama çalışma süresini ve karakter karşılaştırma sayısını iyileştirirken, bunun karşılığında hash karşılaştırma sayısını da artırmıştır.

Ortalama çalışma süreleri açısından Hash- q algoritması ile karşılaştırıldığında, Hq-QF algoritması E. Coli veri setinde %84.41 hızlanma, Human Chromosome 1 veri setinde %71.86 hızlanma ve Human Chromosome 2+3'te %71.43 hızlanma elde etmiştir. Ayrıca önerilen algoritmaların, ortalama çalışma süresi için çoğu dizi uzunluğunda HqUF ve HqUF-BM algoritmalarından daha iyi sonuçlar ürettiği gösterilmiştir.

Ek olarak, ortalama karakter karşılaştırma sayısı açısından, Hash- q , HqUF, HqUF-BM ve HqBM-QF algoritmalarıyla karşılaştırıldığında E. Coli, Human Chromosome 1 ve Human Chromosome 2+3 veri kümeleri için Hq-QF ve HqBM-QF algoritmalarında %100 iyileştirme sağlanmıştır. Ayrıca, dizi uzunluğu q değerine eşit olduğunda, ortalama karakter karşılaştırma sayısı sıfırdır.

Hash- q algoritmasına kıyasla ortalama hash karşılaştırma sayısı açısından, Hq-QF algoritması E. Coli veri setinde %96.80, Human Chromosome 1 veri setinde %78.01 ve Human Chromosome 2+3 veri setinde %84.13 iyileşme elde etmiştir. Ancak, Hq-QF ve HqBM-QF algoritmaları için ortalama hash karşılaştırma sayısı, bazı dizi uzunlukları için Hash- q , HqUF ve HqUF-BM algoritmalarından daha yüksek elde edilmiştir.

Böylece, önerilen yaklaşımlar karakter karşılaştırmalarını q -çerçeve hash karşılaştırmaları ile değiştirirerek yeni tekli kesin dizi eşleştirme algoritmaları önerilmiştir. Deneysel bulgular önerilen yaklaşımların, literatürde iyi bilinen algoritmalarından daha iyi çalışma zamanı verimliliği ve daha az karakter karşılaştırması sağladığı göstermiştir.

Gelecekteki çalışmalarda literatürdeki çoklu kesin dizi eşleştirme algoritmalarındaki eksiklikler tespit edilerek, DNA sekanslarında mükemmel hash fonksiyonuna dayalı çoklu kesin dizi eşleştirme algoritmaları geliştirilebilir.

5.3. WM-q Algoritmasının Deneysel Çalışma Sonuçları

Çizelge 5.27'de gösterildiği gibi, deneysel çalışmalarda iki farklı veri seti kullanılmıştır. E. Coli (Durfee et al., 2008) ve Human Chromosome 1 (Gregory et al., 2004) veri setleri, verilen referans numaralarıyla açık kaynaklı NCBI sitesinden metin dosyası formatında indirilmiştir. E. Coli veri seti 4.6 milyon baz içerir ve Human Chromosome1 veri seti 230 milyon baz içerir.

Çizelge 5.27. WM-q algoritmasında kullanılan veri setleri.

Veri Seti	NCBI Referans No	Uzunluk (bp)
E. Coli	NC_010473.1	4,753,183
Human Chromosome 1	NC_000001.11	248,956,422

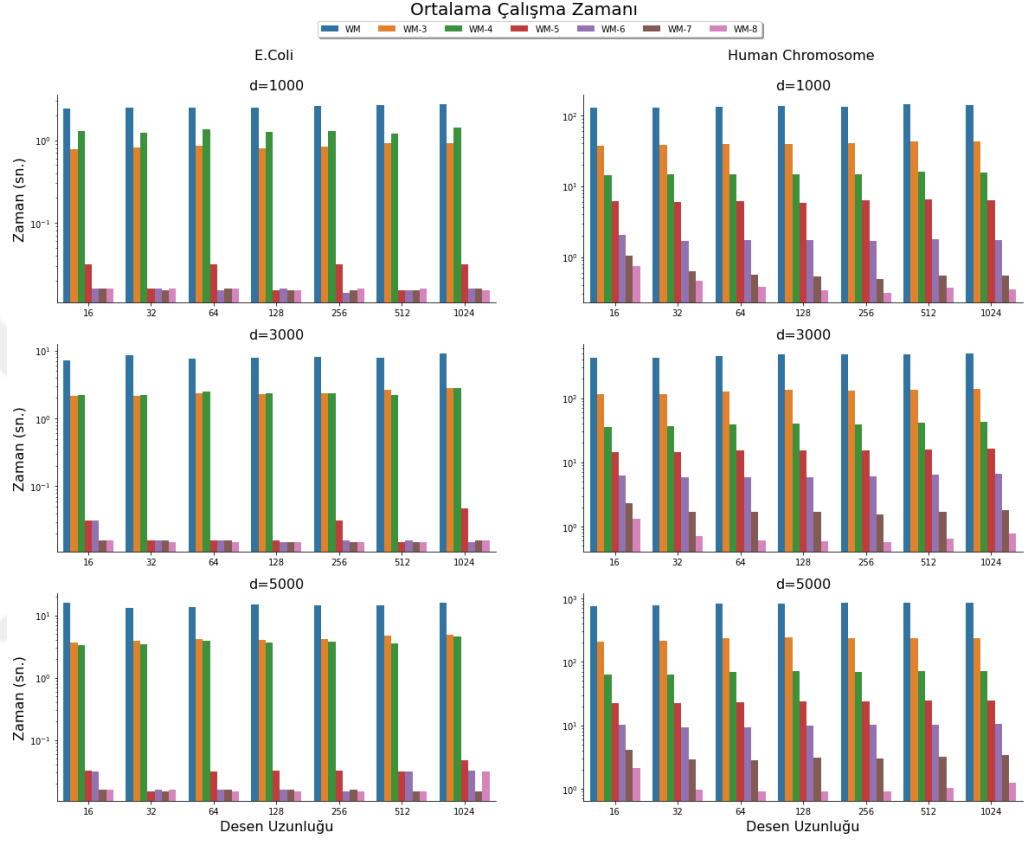
Her bir veri setinden $m = 16,32,64,128,256,512,1024$ uzunluğunda rastgele diziler seçilmiştir. P aranan dizi kümesi, m uzunluklu $d = 1000,3000,5000$ adet diziden oluşmaktadır. Her bir algoritma 10 kez çalıştırılmıştır ve ortalama çalışma zamanı (saniye cinsinden) elde edilmiştir. Ayrıca her bir algoritma için karakter karşılaştırma sayısı ve hash karşılaştırma sayısı elde edilmiştir. Önerilen WM-q algoritmasındaki q değerleri $3 \leq q \leq 8$ olarak seçilmiş ve her q değeri tüm veri setlerine uygulanmış ve sonuçlar WM algoritması ile karşılaştırılmıştır. Ortalama çalışma zamanındaki hızlanma yüzdeleri ve tüm d dizi kümesindeki her bir dizi uzunluğu için ortalama karakter ve hash karşılaştırma sayılarında iyileştirme yüzdeleri gösterilmiştir. En yüksek yüzde değerleri kalın harflerle gösterilmiştir.

Tüm algoritmalar C programlama dilinde derlenmiştir. Kullanılan bilgisayarın özellikleri Intel(R) Core (TM) 64-bit, i7-9750H CPU, 2.60 GHz, 32 GB RAM, Nvidia GeForce RTX 2060'tır.

5.3.1. Ortalama Çalışma Zamanı Sonuçları

E. Coli ve Human Chromosome 1 veri setlerindeki d adet m -uzunluklu diziler için ortalama çalıştırma zamanı sonuçları, logaritmik ölçek ile Şekil 5.1'de gösterilmektedir. WM algoritmasına kıyasen, E. Coli veri setinde, WM-q algoritması tüm $d = 1000,3000,5000$ adet m -uzunluklu dizilerde ($m = 256$ ve $q = 8$ için) sırasıyla %99.45, %99.82 ve %99.90 hızlanma elde etmiştir. Ayrıca WM-q algoritmasının tüm q değerlerinde hızlanmalar elde edilmiştir ve $q = 8$ için en iyi sonuçlar elde edilmiştir.

WM algoritmasına kıyasen, Human Chromosome 1 veri setinde, WM-q algoritması tüm $d = 1000, 3000, 5000$ adet m -uzunluklu diziler için ($m = 256$ ve $q = 8$ için) sırasıyla %99.76, %99.88 ve %99.89 hızlanma elde etmiştir. Ayrıca WM-q algoritmasının tüm q değerlerinde hızlanmalar elde edilmiştir ve $q = 8$ için en iyi sonuçlar elde edilmiştir.



Şekil 5.1. WM-q algoritmasının ortalama çalışma süresi sonuçları ($d=1000, 3000, 5000$).

5.3.2. Ortalama Karakter Karşılaştırma Sayısı Sonuçları

E. Coli ve Human Chromosome 1 veri setlerindeki d adet m -uzunluklu diziler için ortalama karakter karşılaştırma sayısı Çizelge 5.28’de gösterilmektedir. E. Coli veri setinde WM algoritmasına kıyasen, WM-q algoritmasında tüm $d= 1000, 3000, 5000$ adet m -uzunluklu dizilerde ($m = 16$ ve $q = 8$ için), %100 iyileşme sağlanmıştır. Ayrıca WM-q algoritmasının tüm q değerlerinde performans artışı elde edilmiştir ve $q = 8$ için en iyi sonuçlar elde edilmiştir. Human Chromosome 1 veri setinde WM algoritmasına kıyasen, WM-q algoritmasında, tüm $d= 1000, 3000, 5000$ adet m -uzunluklu dizilerde ($m = 16$ ve $q = 8$ için), %100 iyileşme sağlanmıştır. Ayrıca WM-q

algoritmasının tüm q değerlerinde performans artışı elde edilmiştir ve $q = 8$ için en iyi sonuçlar elde edilmiştir.

Çizelge 5.28. Ortalama karakter karşılaştırma sayısı sonuçları ($d=1000,3000,5000$).

$d=1000$	E. Coli		16	32	64	128	256	512	1024
		WM	407045960	410623080	412068230	404530510	418058000	400118770	415956560
		W=3	81637010	81300710	85825870	79948860	84064130	91217350	89430150
		W=4	4818030	4711540	5142260	4637400	4929840	4641700	6050950
		W=5	249640	160960	213010	256520	365100	617440	1244400
		W=6	25820	30770	89150	144880	246100	505400	1136540
		W=7	4090	21070	82250	139200	242120	501160	1132090
		W=8	0	18250	78800	137160	240000	498990	1129720
		%	100%	99.99%	99.98%	99.96%	99.94%	99.87%	99.72%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	21741678975	21072127896	22491502264	22029836324	21264368589	22144952500	21078145425
		W=3	5939244941	5719587808	6152827884	5960913311	5789632012	6126403724	5768475306
		W=4	1564099122	1522731259	1633907258	1548435123	1523355259	1669797482	1541629971
		W=5	279467440	261331604	277332480	262512707	270405606	295261132	265802392
		W=6	21304223	14768810	17481452	18110162	15978492	19785306	17735432
		W=7	1747015	912553	998384	982249	824238	1213248	1799986
		W=8	0	206563	290138	262293	287331	549974	1081226
		%	100%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%
$d=3000$	E. Coli		16	32	64	128	256	512	1024
		WM	1221137880	1231869240	1236204690	1213591530	1254174000	1200356310	1247869680
		W=3	244911030	243902130	257477610	239846580	252192390	273652050	268290450
		W=4	14454090	14134620	15426780	13912200	14789520	13925100	18152850
		W=5	748920	482880	639030	769560	1095300	1852320	3733200
		W=6	77460	92310	267450	434640	738300	1516200	3409620
		W=7	12270	63210	246750	417600	726360	1503480	3396270
		W=8	0	54750	236400	411480	720000	1496970	3389160
		%	100%	99.99%	99.98%	99.97%	99.94%	99.88%	99.73%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	65252283507	64756522381	66523632624	66729987205	63794694394	64925185209	64705630754
		W=3	17847864168	17673025338	18298047327	18224627685	17357512038	17767630131	17786202021
		W=4	4872288871	4838273642	4988929334	4955427078	4739861954	4910896953	4903841593
		W=5	1274972891	1267019823	1294105204	1294323002	1246388124	1294650928	1294650928
		W=6	205287236	199702709	200681060	203781494	201191365	220916192	217237739
		W=7	16751354	13244458	14222901	15021914	13042236	15878575	17108071
		W=8	0	1606662	1444452	1591944	1547863	2480339	3982228
		%	100%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%
$d=5000$	E. Coli		16	32	64	128	256	512	1024
		WM	2035229800	2053115400	2060341150	2022652550	2090290000	2000593850	2079782800
		W=3	408185050	406503550	429129350	399744300	420320650	456086750	447150750
		W=4	24090150	23557700	25711300	23187000	24649200	23208500	30254750
		W=5	1248200	804800	1065050	1282600	1825500	3087200	6222000
		W=6	129100	153850	445750	724400	1230500	2527000	5682700
		W=7	20450	105350	411250	696000	1210600	2505800	5660450
		W=8	0	91250	394000	685800	1200000	2494950	5648600
		%	99.90%	99.88%	99.82%	99.89%	99.89%	99.89%	99.90%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	109058704220	107636088848	109417623815	110881363112	107764301694	107952998214	107261739728
		W=3	29843020556	29295053553	29981561780	30379716198	29434396479	29537099272	29433404359
		W=4	8178697520	8032845938	8190296143	8341489692	8065870169	8179579644	8137498569
		W=5	2217308454	2176663920	2216262345	2260935713	2194048684	2266660609	2250850124
		W=6	482329872	452594940	453746344	480791723	479969094	507144426	511039047
		W=7	49867913	41824366	40059605	46856056	45249429	51025311	53932143
		W=8	0	4308097	3474012	3921332	3998133	5736528	8623102
		%	100%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%

5.3.3. Ortalama Hash Karşılaştırma Sayısı Sonuçları

E. Coli ve Human Chromosome 1 veri setlerindeki d adet m -uzunluklu diziler için ortalama hash karşılaştırma sayısı Çizelge 5.29'da gösterilmektedir. $d = 1000,3000,5000$ adet m -uzunluklu dizilerde ($m = 64,128,256,512,1024$ için), %99,99 iyileştirme sağlanmıştır. Ayrıca WM- q algoritmasının tüm q değerlerinde performans artışı elde edilmiştir ($q = 8$ için en iyi sonuçlar).

Çizelge 5.29. Ortalama hash karşılaştırma sayısı sonuçları (d=1000,3000,5000).

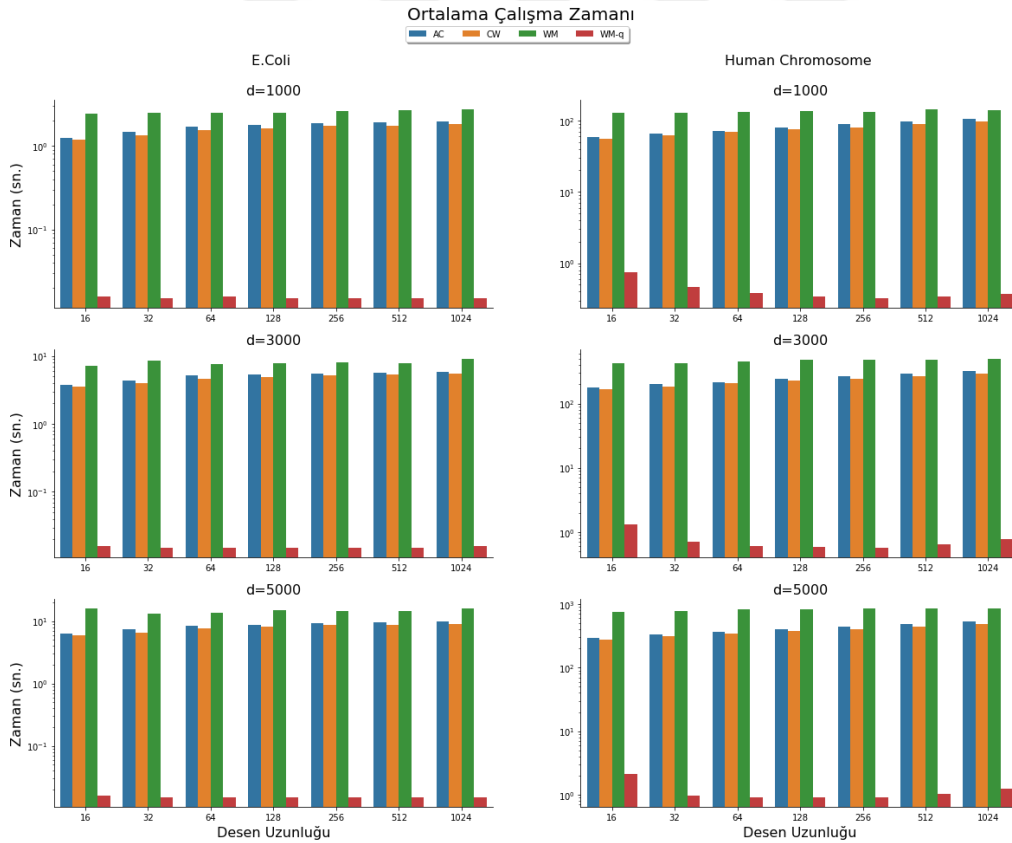
d=1000	E. Coli		16	32	64	128	256	512	1024
		WM	302854910	303186354	302057612	304696578	313148710	296888284	309030544
		W=3	63484481	63624213	66422810	62735642	66095661	71356879	68653180
		W=4	4268488	4166432	4491718	4069724	4181872	3721474	4394573
		W=5	496217	219336	171276	155771	161796	152916	146848
		W=6	399013	136857	49952	17635	9444	8255	7817
		W=7	457281	165461	67814	26358	9211	3642	1967
		W=8	518202	183761	79177	35402	15543	6810	3196
		%	99.86%	99.95%	99.98%	99.99%	99.99%	99.99%	99.99%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	16032814087	15569879429	16540313554	16270721871	15705473323	16234129970	15541185642
		W=3	4542346308	4384901521	4717813653	4537779952	4438073974	4641019917	4404785409
		W=4	1350152819	1326837492	1401688323	1324270566	1320168225	1414572111	1315002652
		W=5	338094538	326487555	330155345	311947668	331200247	341474666	310210379
		W=6	50652981	37581352	39182061	40505139	37965079	41596182	35391576
		W=7	21442280	7669852	4385867	4197457	3155342	3445959	3217621
		W=8	24497831	7742131	2717341	1211115	555243	534806	498356
		%	99.85%	99.95%	99.98%	99.99%	99.99%	99.99%	99.99%
d = 3000	E. Coli		16	32	64	128	256	512	1024
		WM	899192490	900186854	896800692	904717718	930074370	881293604	917721404
		W=3	183434541	184083753	192248230	181039022	191111621	206157299	198473640
		W=4	11213528	10992612	11930998	10651504	10970772	9762514	11613213
		W=5	833917	415256	354816	321731	338776	317116	309048
		W=6	428753	149697	58232	24175	14424	13635	13537
		W=7	462161	168501	71514	28998	11331	5902	4327
		W=8	520922	186061	82517	37862	17543	8850	5456
		%	99.95%	99.98%	99.99%	99.99%	99.99%	99.99%	99.99%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	47633740413	47314050896	48417121945	48736261072	46668453551	47389678746	47153217797
		W=3	13189604432	13043317943	13505577980	13411704075	12820717302	13044979237	13097605374
		W=4	3735201059	3709804598	3818057776	3774406011	3638317392	3735293607	3739697577
		W=5	1121912813	1113950146	1129251677	1119274725	1090572473	1112419933	1112419933
		W=6	259770427	253621715	248272688	245637649	248905505	260106625	256121251
		W=7	44588101	31692519	31063419	31112277	27578241	29464403	29289884
		W=8	25904776	8098785	4712848	3753108	2752482	3327518	3282432
		%	99.95%	99.98%	99.99%	99.99%	99.99%	99.99%	99.99%
d = 5000	E. Coli		16	32	64	128	256	512	1024
		WM	1495530070	1497187354	1491543772	1504738858	1547000030	1465698924	1526412264
		W=3	303384601	304543293	318073650	299342402	316127581	340957719	328294100
		W=4	18158568	17818792	19370278	17233284	17759672	15803554	18831853
		W=5	1171617	611176	538356	487691	515756	481316	471248
		W=6	458493	162537	66512	30715	19404	19015	19257
		W=7	467041	171541	75214	31638	13451	8162	6687
		W=8	523642	188361	85857	40322	19543	10890	7716
		%	99.96%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%
	Human Chromosome1		16	32	64	128	256	512	1024
		WM	79444520557	78571618521	79666891178	80732838579	78561742975	78649123023	78064299616
		W=3	21882132009	21481079302	22003373349	22247559670	21582439071	21572706069	21521486122
		W=4	6119176144	6014177226	6128295160	6218237344	6032010756	6083785154	6053279849
		W=5	1801764573	1770073131	1799890925	1819692802	1773637751	1798993195	1788126297
		W=6	504047675	471614044	470454951	487358993	488035660	496139467	497728043
		W=7	92360051	69538577	67096922	75371075	72236946	72965725	73435959
		W=8	30113696	11021981	8303202	8567781	7388240	7761230	8484760
		%	99.96%	99.99%	99.99%	99.99%	99.99%	99.99%	99.99%

5.3.4. Önerilen Yaklaşımın İyi Bilinen Algoritmalarla Karşılaştırılması

Bu bölümde, önerilen WM-q algoritmasının literatürde yaygın olarak kullanılan bazı çoklu kesin dizi eşleştirme algoritmaları ile ortalama çalışma süresi açısından karşılaştırmaları verilmiştir. Önerilen WM-q algoritması ile karşılaştırma için örnek tabanlı AC algoritması, sonek tabanlı CW, hash tabanlı WM algoritması seçilmiştir. Seçilen algoritmalar iki farklı veri kümesine uygulanır. Önerilen WM-q algoritmasında, en iyi q değeri, d sayısı m uzunluklu diziler için parantez içinde gösterilmiştir. Önerilen algoritmada

hızlanma yüzdeleri ortalama çalışma süresi cinsinden gösterilmiş ve en iyi hızlanma yüzdesi koyu renkle verilmiştir.

E. Coli ve Human Chromosome 1 veri setlerindeki d adet m -uzunluklu diziler için önerilen yaklaşımın iyi bilinen algoritmalarla ortalama çalışma süresi sonuçları, logaritmik ölçekte Şekil 5.2’de gösterilmektedir. Literatürde yaygın olarak kullanılan çoklu kesin dizi eşleme algoritmalarından bazıları ile karşılaştırıldığında, E. Coli veri setinde, WM-q algoritmasında tüm $d = 1000, 3000, 5000$ adet m -uzunluklu dizilerde sırasıyla %99.18, %99.72 ve %99.84 hızlanma elde edilmiştir. WM-q algoritmasının tüm q değerlerinin ortalama çalışma süresi (sn.) açısından literatürdeki algoritmalarından daha iyi olduğu gösterilmiştir. Literatürde yaygın olarak kullanılan bazı çoklu kesin dizi eşleme algoritmaları ile karşılaştırıldığında, WM-q algoritmasında, tüm $d = 1000, 3000, 5000$ adet m -uzunluklu dizilerde sırasıyla %99.62, %99.76 ve %99.77 hızlanmalar elde edilmiştir.



Şekil 5.2. İyi bilinen algoritmalarla WM-q algoritmasının ortalama çalışma süresi sonuçları ($d=1000, 3000, 5000$).

AC algoritmasının, T metnindeki P dizi kümesindeki tüm oluşumlarını bulmak için arama süresi, çoğu veri türü için Σ cinsinden hesaplanır (Aho and Corasick, 1975). Böylece AC algoritmasında dizi uzunluğu arttıkça arama süresi doğrusal olarak artar. CW algoritması, AC algoritmasının geliştirilmiş bir versiyonu olduğu için daha iyi sonuçlar elde edilmiştir. WM algoritmasında, DNA dizilerindeki hash çakışmaları nedeniyle en yüksek çalışma zamanı elde edilmiştir. Önerilen WM-q algoritması, DNA dizilerinde mükemmel hash fonksiyonunu kullanarak en iyi çalışma zamanı sonuçlarını elde etmiştir.

5.3.5. Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada, DNA dizileri için mükemmel hash fonksiyonuna dayalı WM-q çoklu kesin dizi eşleştirme algoritması önerilmiştir. Bu hash fonksiyonu kullanılarak WM algoritmasındaki sabit blok uzunluğu arttırılmış ve ortalama çalışma zamanı, ortalama karakter karşılaştırma sayısı ve ortalama hash karşılaştırma sayısı açısından daha iyi sonuçlar elde edilmiştir.

Çoklu kesin dizi eşleme algoritmalarında bir problem olarak DNA dizilerindeki hash çakışmalarını ele alan bu çalışmada, çok sayıda dizinin büyük hacimli veri kümelerinde önerilen WM-q algoritması kullanılarak verimli bir şekilde aranabileceği deneysel çalışmalar sonucunda gösterilmiştir.

Ortalama çalışma süreleri için, önerilen WM-q algoritmasında E. Coli ve Human Chromosome 1 veri setlerinde sırasıyla %99,45 - %99,90 ve %99,76 - %99,89 arasında hızlanmalar elde edilmiştir. Ortalama karakter karşılaştırma sayısı için, önerilen WM-q algoritmasında, WM algoritmasına kıyasla E. Coli veri kümesi ve Human Chromosome 1 veri kümesinde %100 iyileştirme elde edilmiştir. Ortalama hash karşılaştırma sayısı için, önerilen WM-q algoritmasında WM algoritmasına kıyasla E. Coli ve Human Chromosome 1 veri setlerinde %99,99 iyileştirme elde edilmiştir.

E. Coli veri setinde, önerilen WM-3 algoritmasının ortalama çalışma süresi, $d = 1000$ adet m -uzunluklu diziler için WM-4 algoritmasından daha düşüktür. $d = 3000$ adet m -uzunluklu diziler için, WM-3 ve WM-4'ün ortalama çalışma süreleri birbirine yakındır. $q = 5,6,7,8$ ve $d = 1000,3000$ adet dizi için q değeri arttıkça ortalama çalışma süresi azalmıştır. $d = 5000$ adet m -uzunluk diziler için, q değeri arttıkça önerilen tüm algoritmalarındaki ortalama çalışma süresi azalmıştır.

Human Chromosome 1 veri setinde, tüm d adet diziler için q değeri arttıkça ortalama çalışma süresi azalmıştır. Böylece önerilen algoritmalarda en iyi ortalama çalışma süresinin $q = 8$ için elde edildiği gösterilmiştir. Önerilen algoritmalarda uzun dizi uzunlukları için büyük q değerlerinin seçilmesi performans açısından verimli oldu gösterilmiştir.

Gelecekteki çalışmalarda, mükemmel hash fonksiyonunu kullanarak, q -çerçeve hash karşılaştırmasına dayalı çoklu kesin dizi eşleştirme yaklaşımı geliştirilebilir. Böylece q -uzunluklu çerçevelerin performans üzerindeki etkileri incelenebilir.

5.4. q -Gram WM Algoritmasının Deneysel Çalışma Sonuçları

Çizelge 5.30'da gösterildiği gibi, deneysel çalışmalarda iki farklı veri seti kullanılmıştır. E. Coli (Durfee et al., 2008) ve Human Chromosome 1 (Gregory et al., 2004) veri setleri, verilen referans numaralarıyla açık kaynaklı NCBI sitesinden metin dosyası formatında indirilmiştir. E. Coli veri seti 4.6 milyon baz içerir ve Human Chromosome1 veri seti 230 milyon baz içerir. Human Chromosome 1 veri seti H-Chr1 olarak bu çalışmada adlandırılmıştır

Çizelge 5.30. q -Gram WM algoritmasında kullanılan veri setleri.

Veri Seti	NCBI Referans No	Uzunluk (bp)
E. Coli	NC_010473.1	4,753,183
Human Chromosome 1	NC_000001.11	248,956,422

Her bir veri seti içinden rastgele $m = 16,32,64,128,256,512,1024$ uzunluklu diziler seçilmiştir. P diziler kümesi m uzunluklu $d = 1000,3000,5000$ adet diziden oluşmaktadır. Her bir algoritma 10 kere çalıştırılarak ortalama çalışma zamanları (sn.) elde edilmiştir. Ayrıca her bir algoritma için ortalama karakter karşılaştırma sayısı ve hash karşılaştırma sayısı elde edilmiştir. Önerilen q -gram WM algoritmasındaki $q = 8$ olarak seçilmiş olup, tüm veri setlerine uygulanarak sonuçları WM algoritmasıyla ve literatürde iyi bilinen algoritmalarla karşılaştırılmıştır. Tüm d adet dizi kümesindeki her bir dizi uzunluğu için ortalama çalışma zamanındaki hızlanma yüzdeleri, ortalama karakter ve hash karşılaştırma sayılarındaki iyileşme yüzdeleri gösterilmiştir. Elde edilen en yüksek yüzde değerleri ise koyu renkle gösterilmiştir. Tüm algoritmalar C programlama dilinde derlenmiştir. Kullanılan makinenin teknik özellikleri ise Intel(R) Core (TM) 64-bit, i7-9750H CPU, 2.60 GHz, 32 GB RAM, Nvidia GeForce RTX 2060.

5.4.1. WM ve q -Gram WM Algoritmasının Karşılaştırılması

5.4.1.1. Ortalama Çalışma Süresi Sonuçları

Çizelge 5.31 veri setlerindeki d adet m uzunluklu her bir dizideki ortalama çalışma zamanı sonuçlarını göstermektedirler. E. Coli veri setinde WM algoritmasına kıyasla önerilen q -gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu dizilerde ortalama çalışma zamanında sırasıyla 98,73%, 99,48%, 99,71% ($m = 64, 32, 16$ uzunluklarında) hızlanmalar elde edilmiştir. Ayrıca önerilen yaklaşımda tüm dizi uzunluklarında hızlanmalar elde edilmiştir. Human Chromosome 1 veri setinde WM algoritmasına kıyasla q -gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu dizilerde ortalama çalışma zamanında sırasıyla 99,76%, 99,88%, 99,89% ($m = 256, 32, 256$ uzunluklarında) hızlanmalar elde edilmiştir. Ayrıca önerilen yaklaşımda tüm dizi uzunluklarında hızlanmalar elde edilmiştir.

Çizelge 5.31. Ortalama çalışma süresi.

d = 1000	E. Coli		16	32	64	128	256	512	1024
		WM	2,411	2,467	2,454	2,481	2,584	2,636	2,710
		q-gram WM	0,034	0,032	0,031	0,039	0,046	0,066	0,105
		Hızlanma	98,58%	98,70%	98,73%	98,42%	98,21%	97,49%	96,12%
	H-Chr1		16	32	64	128	256	512	1024
WM	126,82	129,85	132,76	134,87	132,55	142,34	141,11		
q-gram WM	0,663	0,465	0,377	0,335	0,314	0,367	0,385		
Hızlanma	99,47%	99,63%	99,71%	99,75%	99,76%	99,74%	99,72%		
d = 3000	E. Coli		16	32	64	128	256	512	1024
		WM	7,201	8,447	7,635	7,758	8,105	7,845	9,035
		q-gram WM	0,042	0,044	0,049	0,061	0,089	0,146	0,265
		Hızlanma	99,42%	99,48%	99,36%	99,21%	98,90%	98,14%	97,07%
	H-Chr1		16	32	64	128	256	512	1024
		WM	422,13	425,13	454,66	474,49	480,45	483,31	492,79
		q-gram WM	1,155	0,725	0,632	0,592	0,576	0,668	0,812
		Hızlanma	99,73%	99,83%	99,86%	99,88%	99,88%	99,86%	99,84%
	d = 5000	E. Coli		16	32	64	128	256	512
WM			16,052	13,395	13,669	14,937	14,314	14,444	16,038
q-gram WM			0,047	0,052	0,108	0,145	0,178	0,243	0,412
Hızlanma			99,71%	99,61%	99,21%	99,03%	98,76%	98,32%	97,43%
H-Chr1			16	32	64	128	256	512	1024
		WM	746,18	769,17	817,11	840,45	852,54	856,64	858,23
		q-gram WM	1,562	1,018	0,907	0,957	0,937	1,055	1,342
		Hızlanma	99,79%	99,87%	99,89%	99,89%	99,89%	99,88%	99,84%

5.4.1.2. Ortalama Karakter Karşılaştırma Sayısı Sonuçları

Çizelge 5.32 veri setlerindeki d adet m uzunluklu her bir dizideki ortalama karakter karşılaştırma sayısı sonuçlarını göstermektedirler. Tüm veri setlerinde WM algoritmasına kıyasla önerilen q-gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu dizilerde ortalama karakter karşılaştırma sayılarında 99,99% daha az karşılaştırma elde edilmiştir. Böylece, önerilen yaklaşımda karakter karşılaştırma sayısı en aza indirgenmiştir.

Çizelge 5.32. Ortalama karakter karşılaştırma sayısı.

$d = 1000$	E. Coli		16	32	64	128	256	512	1024
		WM	407045960	410623080	412068230	404530510	418058000	400118770	415956560
		q-gram WM	1360	1150	1670	1230	1000	1020	1130
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	21741678975	21072127896	22491502264	22029836324	21264368589	22144952500	21078145425
		q-gram WM	245523	62736	43166	35173	15869	22753	27106
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	E. Coli		16	32	64	128	256	512	1024
		WM	1221137880	1231869240	1236204690	1213591530	1254174000	1200356310	1247869680
		q-gram WM	3848	3451	4923	3690	3000	3060	3385
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	65252283507	64756522381	66523632624	66729987205	63794694394	64925185209	64705630754
		q-gram WM	1582368	468407	436603	419725	263990	335862	354549
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
$d = 3000$	E. Coli		16	32	64	128	256	512	1024
		WM	2035229800	2053115400	2060341150	2022652550	2090290000	2000593850	2079782800
		q-gram WM	6800	5750	8350	6150	5000	5100	5650
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	10905870422	10763608884	10941762381	11088136311	10776430169	10795299821	10726173972
		q-gram WM	3341788	1209957	1143475	1318083	1023647	1083551	1239708
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
$d = 5000$	E. Coli		16	32	64	128	256	512	1024
		WM	2035229800	2053115400	2060341150	2022652550	2090290000	2000593850	2079782800
		q-gram WM	6800	5750	8350	6150	5000	5100	5650
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	10905870422	10763608884	10941762381	11088136311	10776430169	10795299821	10726173972
		q-gram WM	3341788	1209957	1143475	1318083	1023647	1083551	1239708
		iyileşme	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%

5.4.1.3. Ortalama Hash Karşılaştırma Sayısı Sonuçları

Çizelge 5.33 veri setlerindeki d adet m uzunluklu her bir dizideki ortalama karakter karşılaştırma sayısı sonuçlarını göstermektedirler. Tüm veri setlerinde WM algoritmasına kıyasla önerilen q-gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu dizilerde ortalama karakter karşılaştırma sayılarında 99,99% daha az karşılaştırma elde edilmiştir. Böylece, önerilen yaklaşımda karakter karşılaştırma sayısı en aza indirgenmiştir.

Çizelge 5.33. Ortalama hash karşılaştırma sayısı.

$d = 1000$	E. Coli		16	32	64	128	256	512	1024
		WM	302854910	303186354	302057612	304696578	313148710	296888284	309030544
		q-gram WM	516842	184911	87347	51392	23843	69030	144446
		iyileşme	99,83%	99,94%	99,97%	99,98%	99,99%	99,98%	99,95%
	H-Chr1		16	32	64	128	256	512	1024
		WM	16032814087	15569879429	16540313554	16270721871	15705473323	16234129970	15541185642
		q-gram WM	24246401	7720371	2743940	1346660	717470	1005180	2156399
		iyileşme	99,84%	99,95%	99,98%	99,99%	99,99%	99,99%	99,99%
	E. Coli		16	32	64	128	256	512	1024
		WM	899192490	900186854	896800692	904717718	930074370	881293604	917721404
		q-gram WM	516842	189451	107027	85832	104543	195510	134366
		iyileşme	99,94%	99,98%	99,99%	99,99%	99,99%	99,98%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	47633740413	47314050896	48417121945	48736261072	46668453551	47389678746	47153217797
		q-gram WM	23842049	7879008	5093809	5709032	5268299	9565394	20490907
		iyileşme	99,95%	99,98%	99,99%	99,99%	99,99%	99,98%	99,96%
$d = 3000$	E. Coli		16	32	64	128	256	512	1024
		WM	1495530070	1497187354	1491543772	1504738858	1547000030	1465698924	1526412264
		q-gram WM	516842	185811	92807	46772	47543	101809	203666
		iyileşme	99,97%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	79444520557	78571618521	79666891178	80732838579	78561742975	78649123023	78064299616
		q-gram WM	25169961	10281457	9236167	13751304	1023647	27963012	65447603
		iyileşme	99,97%	99,99%	99,99%	99,98%	99,99%	99,96%	99,92%
$d = 5000$	E. Coli		16	32	64	128	256	512	1024
		WM	1495530070	1497187354	1491543772	1504738858	1547000030	1465698924	1526412264
		q-gram WM	516842	185811	92807	46772	47543	101809	203666
		iyileşme	99,97%	99,99%	99,99%	99,99%	99,99%	99,99%	99,99%
	H-Chr1		16	32	64	128	256	512	1024
		WM	79444520557	78571618521	79666891178	80732838579	78561742975	78649123023	78064299616
		q-gram WM	25169961	10281457	9236167	13751304	1023647	27963012	65447603
		iyileşme	99,97%	99,99%	99,99%	99,98%	99,99%	99,96%	99,92%

5.4.2. İyi Bilinen Algoritmalarla Karşılaştırma

Literatürde iyi bilinen çoklu dizi eşleştirme algoritmalarından bazılarıyla önerilen q -gram WM algoritmasının ortalama çalışma zamanı açısından karşılaştırması gerçekleştirilmiştir. Önerilen yaklaşımla karşılaştırma yapmak için örnek tabanlı AC algoritması, sonek tabanlı CW, hash tabanlı WM algoritması seçilmiştir ve veri setlerine uygulanmışlardır. Deneysel çalışmalar sonucu ortalama çalışma zamanı açısından elde edilen hızlanma yüzdeleri gösterilmiştir ve en iyi hızlanma yüzdeleri koyu renkle gösterilmiştir. Çizelge 5.34, önerilen yöntem ile iyi bilinen algoritmaların her bir veri seti için d adet m uzunluklu her bir dizideki ortalama çalışma zamanı sonuçlarını göstermektedirler.

E. Coli veri setinde, literatürde iyi bilinen algoritmalarla kıyasla q -gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu ($m = 64, 64, 32$ uzunluklarında) dizilerde ortalama çalışma zamanında sırasıyla 98,01%, 98,95%, 99,22% olarak hızlanmalar elde edilmiştir. Önerilen q -gram WM algoritmasının literatürdeki algoritmalarından daha verimli olduğu gösterilmiştir. Human Chromosome 1 veri setinde, literatürde iyi bilinen algoritmalarla kıyasla q -gram WM algoritmasında tüm $d = 1000, 3000, 5000$ adet m uzunluklu dizilerde ortalama çalışma zamanında sırasıyla 99,61%, 99,76%, 99,77% olarak hızlanmalar elde edilmiştir. Önerilen q -gram WM algoritmasının literatürdeki algoritmalarından daha verimli olduğu gösterilmiştir.

Çizelge 5.34. İyi bilinen algoritmada ortalama çalışma süresi.

d = 1000	E. Coli		16	32	64	128	256	512	1024
		AC	1,255	1,462	1,712	1,765	1,845	1,905	1,953
		CW	1,196	1,332	1,554	1,611	1,715	1,756	1,824
		WM	2,411	2,467	2,454	2,481	2,584	2,636	2,710
		q-gram WM	0,034	0,032	0,031	0,039	0,046	0,066	0,105
		Hızlanma	97,16%	97,60%	98,01%	97,58%	97,32%	96,24%	94,24%
	H-Chr1		16	32	64	128	256	512	1024
		AC	58,75	66,23	72,15	80,56	89,63	97,81	105,72
		CW	55,21	61,71	68,63	75,65	80,21	89,56	96,32
		WM	126,82	129,85	132,76	134,87	130,55	142,34	141,11
		q-gram WM	0,663	0,465	0,377	0,335	0,314	0,367	0,385
		Hızlanma	98,80%	99,25%	99,45%	99,56%	99,61%	99,59%	99,60%
d = 3000	E. Coli		16	32	64	128	256	512	1024
		AC	3,765	4,386	5,136	5,295	5,535	5,715	5,859
		CW	3,588	3,996	4,662	4,833	5,145	5,268	5,472
		WM	7,201	8,447	7,635	7,758	8,105	7,845	9,035
		q-gram WM	0,042	0,044	0,049	0,061	0,089	0,146	0,265
		Hızlanma	98,83%	98,90%	98,95%	98,74%	98,27%	97,23%	95,16%
	H-Chr1		16	32	64	128	256	512	1024
		AC	176,25	198,69	216,45	241,68	268,89	293,43	317,16
		CW	165,63	185,13	205,89	226,95	240,63	268,68	288,96
		WM	422,13	425,13	454,66	474,49	480,45	483,31	492,79
		q-gram WM	1,155	0,725	0,632	0,592	0,576	0,668	0,812
		Hızlanma	99,30%	99,61%	99,69%	99,74%	99,76%	99,75%	99,72%
d = 5000	E. Coli		16	32	64	128	256	512	1024
		AC	6,275	7,31	8,56	8,825	9,225	9,525	9,765
		CW	5,98	6,66	7,77	8,055	8,575	8,78	9,12
		WM	16,052	13,395	13,669	14,937	14,314	14,444	16,038
		q-gram WM	0,047	0,052	0,108	0,145	0,178	0,243	0,412
		Hızlanma	99,21%	99,22%	98,61%	98,20%	97,92%	97,23%	95,48%
	H-Chr1		16	32	64	128	256	512	1024
		AC	293,75	331,15	360,75	402,8	448,15	489,05	528,6
		CW	276,05	308,55	343,15	378,25	401,05	447,8	481,6
		WM	746,18	769,17	817,11	840,45	852,54	856,64	858,23
		q-gram WM	1,562	1,018	0,907	0,957	0,937	1,055	1,342
		Hızlanma	99,43%	99,67%	99,74%	99,75%	99,77%	99,76%	99,72%

AC algoritmasının P dizi kümesindeki herhangi bir dizinin tüm oluşumlarını T girdi dizisinde arama süresi, çoğu veri türü için alfabenin boyutu olarak $(|\Sigma|)$ hesaplanır. Bu yüzden dizi uzunluğu artmasına rağmen AC algoritmasında çalışma zamanı artsa da lineer bir artış göstermektedir. CW algoritması ise AC algoritmasının gelişmiş bir versiyonu olduğu için daha iyi sonuçlar elde edilmiştir. WM algoritması DNA sekanslarındaki hash çakışmalarından dolayı çalışma zamanı olarak en yavaş algoritmadır. Önerilen q -gram WM algoritması ise DNA sekanslarında eşsiz hash fonksiyonunu ve önerilen yaklaşımı kullandığından en verimli sonuçlar elde edilmiştir.

5.4.3. Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada DNA sekanslarında hash çakışmasını kaldıran eşsiz hash fonksiyonunu ve q -gram hash karşılaştırma yöntemini kullanarak WM algoritmasına dayalı q -gram WM çoklu dizi eşleştirme algoritması önerilmiştir. Kullanılan hash fonksiyonu sayesinde geleneksel WM algoritmasındaki sabit blok uzunluğu artırılarak ortalama çalışma zamanında, ortalama karakter karşılaştırma sayısında ve ortalama hash karşılaştırma sayısında daha iyi sonuçlar elde edilmiştir.

Ortalama çalışma süreleri için, WM algoritmasına kıyasen önerilen q -gram WM algoritmasında E. Coli veri setinde 98,73%-99,71% arasında hızlanma, Human Chromosome 1 veri setinde 99,76%-99,89% arasında hızlanma elde edilmiştir. WM algoritmasına kıyasen önerilen q -gram WM algoritmasında ortalama karakter ve hash karşılaştırma sayılarında 99,99% azalma elde edilmiştir. Böylelikle, tüm performans metriklerinde daha verimli bir algoritma önerildiği gösterilmiştir. Bu çalışmadaki katkılarımız şöyledir;

- DNA sekansları için WM çoklu dizi eşleştirme algoritmasında meydana gelen hash çakışmalarını ortadan kaldıran mükemmel hash fonksiyonu kullanılmıştır.
- Mükemmel hash fonksiyonu sayesinde WM algoritmasındaki sabit $B = 2$ blok uzunluğunu kullanmak yerine $q = 8$ aralığında daha büyük blok uzunluğu kullanılmıştır.
- Önek dizisi veya karakter eşleşmediğinde WM algoritmasındaki $B = 2$ birim dizileri kaydırma yerine $q = 8$ birim dizilerin kaydırılması sağlanmıştır.
- DNA sekanslarında aranan dizilerin q uzunluklu sonek ve önek dizilerinin karşılaştırılmasına gerek kalmamış ve bu ekler arasındaki karakterler q -gram hash karşılaştırılmasıyla kontrol edilmiştir.
- Önek dizisinden sonra ilk karakter kontrolü yapılarak olası eşleşmeme olduğunda fazladan q -gram hash karşılaştırma maliyeti en aza indirilmiştir.

Gelecekteki çalışmalarda dizi eşleştirme algoritmaları ile önerilen yaklaşım birleştirilerek hibrit modeller oluşturulabilir. Mükemmel hash fonksiyonunu kullanarak oluşturulan yeni modelin başarısı deneysel çalışmalarla doğrulanabilir.

6. SONUÇ

Kesin dizi eşleştirme algoritmaları, T metninde P dizisinin tüm oluşumlarını bulmayı içerir. Bu algoritmalar, özellikle metin arama ve hesaplamalı biyoloji gibi çeşitli alanlardaki uygulamaları nedeniyle bilgisayar bilimlerinde kapsamlı bir şekilde incelenmiştir. Kesin dizi eşleştirme algoritmalarında temel amaç, mümkün olan en kısa zaman diliminde tüm dizi eşleşmelerini doğru bir şekilde bulmaktır. Hash tabanlı dizi eşleştirme algoritmaları hızlı çalışmasına rağmen, hash çakışmaları gibi eksiklikleri vardır.

Tez çalışmasında, DNA dizilerinde hash çakışmalarını ortadan kaldıran yeni bir hash fonksiyonu önerilmiştir. Bu fonksiyon hash değerlerini zaman açısından verimli bir şekilde üretir. Önerilen hash fonksiyonuna dayalı, ilk çalışmamızda UniqueFNG'li Hash-q algoritması ve UniqueFNG'li Hash-q Boyer-Moore algoritması isminde iki adet kesin dizi eşleştirme algoritması önerilmiştir. İlk yaklaşımda, geleneksel Hash-q algoritmasının hash fonksiyonu önerilen ile değiştirilmiştir. İkinci yaklaşımda, kesin bir eşleşme bulunduğunda Boyer-Moore algoritmasındaki iyi sonek kaydırma tablosunun $(m - 1)$ 'inci girişte belirtilen kaydırma değeri kullanarak birinci yaklaşım geliştirilmiştir.

İkinci çalışmada, eşsiz q -çerçeve Hash karşılaştırmalı Hash-q algoritması ve eşsiz q -çerçeve Hash karşılaştırmalı Hash-q Boyer-Moore algoritması olmak üzere iki kesin dizi eşleştirme algoritması önerilmiştir. Birinci çalışmadaki mükemmel hash fonksiyonuna dayalı olan bu algoritmalarda karakter karşılaştırmalarını tek tek yapmak yerine q -çerçeve hash karşılaştırmaları önerilmiştir. Böylece, ilk çalışmadaki her iki algoritmaya bu yöntem uygulanarak algoritmalar geliştirilmiştir ve karakter karşılaştırma sayısı minimize edilmiştir.

Üçüncü çalışmada, mükemmel hash fonksiyonu kullanılarak DNA dizilerinde çoklu kesin dizi eşleştirme gerçekleştirilmiştir. Çoklu kesin dizi eşleştirme algoritmalarından biri olan WM algoritmasının yeni versiyonu olan WM-q algoritması önerilmiştir. Önerilen algoritmada, mükemmel hash fonksiyonu kullanılarak DNA dizilerindeki hash çakışmaları ortadan kaldırılmıştır. Böylece metin ve aranan dizilerin sonek ve önek karakterlerini karşılaştırmaya gerek kalmamıştır. Önerilen WM-q algoritmasında, WM algoritmasındaki sabit blok uzunluğu $B = 2$ yerine daha büyük blok uzunluğu

olan $3 \leq q \leq 8$ aralığında bloklar kullanılmıştır. Ayrıca, WM algoritmasından farklı olarak, diziler metin üzerinde daha fazla kaydırılmıştır.

Son çalışmada, üçüncü çalışmada geliştirilen WM- q çoklu kesin dizi eşleştirme algoritmasındaki karakter karşılaştırmalarını tek tek yapmak yerine q -gram hash karşılaştırmasına dayalı karşılaştırma önerilmiştir. Böylece, WM- q algoritmasına bu yöntem uygulanarak yeni bir algoritma olan q -gram WM algoritması geliştirilmiştir ve karakter karşılaştırma sayısı minimize edilmiştir.

Tüm bunların sonucunda, tez çalışmamda DNA sekanslarında hash çakışmalarını ortadan kaldıran eşsiz hash fonksiyonuna dayalı geliştirilen tekli ve çoklu kesin dizi eşleştirme algoritmalarında, literatürdeki algoritmalara kıyasen tüm performans metrikleri açısından daha iyi sonuçlar elde edilmiştir ve daha verimli algoritmalar önerildiği gösterilmiştir.

7. TARTIŞMA

Bu tez çalışmasında, DNA sekanslarına yönelik tekli ve çoklu kesin dizi eşleştirme algoritmaları geliştirilmiştir. Geliştirilen algoritmalar hash tabanlı algoritmalarlardır. DNA sekanslarında hash çakışmasını ortadan kaldıran mükemmel hash fonksiyonu geliştirilmiştir ve bu hash fonksiyonu önerilen tekli ve çoklu kesin dizi eşleştirme algoritmalarına uygulanmıştır. Mükemmel hash fonksiyonu sayesinde literatüre DNA sekanslarında dizi eşleştirme problemlerinin daha verimli bir biçimde gerçekleştirilmesine yönelik katkılar sağlanmıştır. Geliştirilen algoritmalar ile literatürdeki bazı hash tabanlı tekli ve çoklu kesin dizi eşleştirme algoritmalarının performansları artırılmıştır.

Doktora eğitim ve tez sürecinde farklı problemler için çeşitli çalışmalar gerçekleştirilmiştir. Bu çalışmalar geliştirilen yöntemlerle literatürde iyi bilinen algoritmalar belli veri setleri üzerinde çalıştırılarak performans metrikleri açısından karşılaştırılmıştır. Elde edilen başarılı sonuçlar bilimsel yayınlar ile desteklenmiştir.

KAYNAKLAR DİZİNİ

- AbdulRazzaq, A. A., Rashid, N. A. A., Hasan, A. A., and Abu-Hashem, M. A.,** 2013, The exact string matching algorithms efficiency review, *Global Journal on Technology*, 4(2).
- Aho, A. V., and Corasick, M. J.,** 1975, Efficient string matching: an aid to bibliographic search, *Communications of the ACM*, 18(6), 333-340. Doi: [10.1145/360825.360855](https://doi.org/10.1145/360825.360855)
- Aldwairi, M., Hamzah, A. Y., and Jarrah, M.,** 2019, MultiPLZW: a novel multiple pattern matching search in LZW-compressed data, *Computer Communications*, 145, 126-136. Doi: [10.1016/j.comcom.2019.06.011](https://doi.org/10.1016/j.comcom.2019.06.011)
- Alfred, V.,** 2014, Algorithms for finding patterns in strings, *Algorithms and Complexity*, 1, 255.
- Al-Qiari, A. K., and Al-Issa, Y.,** 2018, A fast improved multiple pattern matching algorithm, *International Conference on Information and Communication Systems*, 55-60. Doi: [10.1109/IACS.2018.8355441](https://doi.org/10.1109/IACS.2018.8355441)
- Bakr, N. S., and Sharawi, A. A.,** 2013, DNA lossless compression algorithms, *American Journal of Bioinformatics Research*, 3(3), 72-81. Doi: [10.5923/j.bioinformatics.20130303.04](https://doi.org/10.5923/j.bioinformatics.20130303.04)
- Ben-Bassat, I., and Chor, B.,** 2014, String graph construction using incremental hashing, *Bioinformatics*, 30(24), 3515-3523. Doi: [10.1093/bioinformatics/btu578](https://doi.org/10.1093/bioinformatics/btu578).
- Berry, T., and Ravindran, S.,** 1999, A Fast String Matching Algorithm and Experimental Results, *Stringology*, 16-28.
- Boyer, R. S., and Moore, J. S.,** 1977, A fast string searching algorithm, *Communications of the ACM*, 20(10), 762-772. Doi: [10.1145/359842.359859](https://doi.org/10.1145/359842.359859)
- Charras, C., Lecrog, T., and Pehoushek, J. D.,** 1998, A very fast string matching algorithm for small alphabets and long patterns, *Symposium on Combinatorial Pattern Matching*, 55-64. Doi: [10.1007/BFb0030780](https://doi.org/10.1007/BFb0030780)
- Chen, L., Lu, S., and Ram, J.,** 2004, Compressed pattern matching in dna sequences. *IEEE Computational Systems Bioinformatics Conference*, 62-68. Doi: [10.1109/CSB.2004.1332418](https://doi.org/10.1109/CSB.2004.1332418)

KAYNAKLAR DİZİNİ (devam)

- Cibikdiken, A. O., Buyukyilmaz, M., Nacar, M. A., Topcu, A. E., Tambag, Y., and Karadag, A.,** 2017, Development of a New Platform Through Distributed Storage System for the Bioinformatics Analysis, *DEStech Transactions on Computer Science and Engineering*.
- Commentz-Walter, B.,** 1979, A string matching algorithm fast on the average, *International Colloquium on Automata, Languages, and Programming*, 118-132. Doi: [10.1007/3-540-09510-1_10](https://doi.org/10.1007/3-540-09510-1_10)
- Dai, L.,** 2009, An aggressive algorithm for multiple string matching, *Information processing letters*, 109(11), 553-559. Doi: [10.1016/j.ipl.2009.01.022](https://doi.org/10.1016/j.ipl.2009.01.022)
- Deorowicz, S., and Grabowski, S.,** 2013, Data compression for sequencing data, *Algorithms for Molecular Biology*, 8(1), 1-13. Doi: [10.1186/1748-7188-8-25](https://doi.org/10.1186/1748-7188-8-25)
- Durfee T, Nelson R, Baldwin S, et al.,** “Escherichia coli str. K-12 substr. DH10B, complete sequence dataset”, https://www.ncbi.nlm.nih.gov/nuccore/NC_010473.1 (2008), (Erişim tarihi: 04.02.2020)
- Faro, S.,** 2016, A very fast string matching algorithm based on condensed alphabets, *International Conference on Algorithmic Applications in Management*, 65-76. Doi: [10.1007/978-3-319-41168-2_6](https://doi.org/10.1007/978-3-319-41168-2_6)
- Faro, S., and Külekci, M. O.,** 2013, Towards a Very Fast Multiple String Matching Algorithm for Short Patterns, *Stringology*, 78-91. Url: [Scholar](#)
- Faro, S., and Lecroq, T.,** 2013, The exact online string matching problem: A review of the most recent results, *ACM Computing Surveys (CSUR)*, 45(2), 1-42. Doi: [10.1145/2431211.2431212](https://doi.org/10.1145/2431211.2431212)
- Faro, S., and Scafiti, S.,** 2021), Efficient String Matching Based on a Two-Step Simulation of the Suffix Automaton. *International Conference on Implementation and Application of Automata* (65-177). Doi: [10.1007/978-3-030-79121-6_14](https://doi.org/10.1007/978-3-030-79121-6_14)
- Greenfield, P., and Roehm, U.,** 2013, Answering biological questions by querying k-mer databases, *Concurrency and Computation: Practice and Experience*, 25(4), 497-509. Doi: [10.1002/cpe.2938](https://doi.org/10.1002/cpe.2938)

KAYNAKLAR DİZİNİ (devam)

- Gregory S.G, Barlow K.F, McLay K.E, et al.**, “Homo sapiens chromosome 1, GRCh38.p13 Primary Assembly”,
https://www.ncbi.nlm.nih.gov/nuccore/NC_000001.11 (2004), (Erişim tarihi: 04.02.2020)
- Gupta, S., Gupta, B. B., and Chaudhary, P.**, 2019, A client-server JavaScript code rewriting-based framework to detect the XSS worms from online social network, *Concurrency and Computation: Practice and Experience*, 31(21). Doi: [10.1002/cpe.4646](https://doi.org/10.1002/cpe.4646)
- Hakak, S. I., Kamsin, A., Shivakumara, P., Gilkar, G. A., Khan, W. Z., and Imran, M.**, 2019, Exact string matching algorithms: Survey, issues, and future research directions, *IEEE access*, 7, 69614-69637. Doi: [10.1109/ACCESS.2019.2914071](https://doi.org/10.1109/ACCESS.2019.2914071)
- Hillier L.W, Graves T.A, Fulton R.S, et al.**, “Homo sapiens chromosome 2, GRCh38.p13 Primary Assembly”,
https://www.ncbi.nlm.nih.gov/nuccore/NC_000002.12 (2005), (Erişim tarihi: 04.02.2020)
- Ho, T., Cho, S. J., and Oh, S. R.**, 2018, Parallel multiple pattern matching schemes based on cuckoo filter for deep packet inspection on graphics processing units, *IET Information Security*, 12(4), 381-388. Doi: [10.1049/iet-ifs.2017.042](https://doi.org/10.1049/iet-ifs.2017.042)
- Horspool, R. N.**, 1980, Practical fast searching in strings, *Software: Practice and Experience*, 10(6), 501-506. Doi: [10.1002/spe.4380100608](https://doi.org/10.1002/spe.4380100608)
- Hyyrö, H., Juhola, M., and Vihinen, M.**, 2005, On exact string matching of unique oligonucleotides, *Computers in Biology and Medicine*, 35(2), 173-181. Doi: [10.1016/j.compbiomed.2004.06.002](https://doi.org/10.1016/j.compbiomed.2004.06.002)
- Jeong, I. S., Park, K. W., Kang, S. H., and Lim, H. S.**, 2010, An efficient similarity search based on indexing in large DNA databases, *Computational biology and chemistry*, 34(2), 131-136. Doi: [10.1016/j.compbiolchem.2010.03.007](https://doi.org/10.1016/j.compbiolchem.2010.03.007)
- Kanuga, P.**, 2015, Clustering oriented hashing based multiple string pattern matching algorithm. *International Conference on Circuits, Power and Computing Technologies*, 1-6. Doi: [10.1109/ICCPCT.2015.7159288](https://doi.org/10.1109/ICCPCT.2015.7159288)

KAYNAKLAR DİZİNİ (devam)

- Karcioglu, A. A., and Bulut, H.,** 2021, Improving hash-q exact string matching algorithm with perfect hashing for DNA sequences, *Computers in Biology and Medicine*, 131, 104292. Doi: [10.1016/j.compbiomed.2021.104292](https://doi.org/10.1016/j.compbiomed.2021.104292)
- Karp, R.,** 1987, Efficient randomized pattern-matching algorithms, *The IBM Journal of Research and Development*. <http://www.research.ibm.com/journal/rd/312/ibmrd3102P.pdf>, 31. Doi: [10.1147/rd.312.0249](https://doi.org/10.1147/rd.312.0249)
- Ke, X., and Yong, C.,** 2006, An improved Wu-Manber multiple patterns matching algorithm, *2006 IEEE International Performance Computing and Communications Conference*, 6. Doi: [10.1109/.2006.1629469](https://doi.org/10.1109/.2006.1629469)
- Khitmoh, N., Smachat, S., and Tongsim, S.,** 2020, Stretch Profile: A pruning technique to accelerate DNA sequence search, *Informatics in Medicine Unlocked*, 19, 100323. Doi: [10.1016/j.imu.2020.100323](https://doi.org/10.1016/j.imu.2020.100323)
- Kim, H., and Han, Y. S.,** 2016, OMPPM: online multiple palindrome pattern matching, *Bioinformatics*, 32(8), 1151-1157. Doi: [10.1093/bioinformatics/btv738](https://doi.org/10.1093/bioinformatics/btv738)
- Kim, S., and Kim, Y.,** 1999, A fast multiple string pattern matching algorithm. *17th AoM/IAoM Conference on Computer Science*, 44-49. Url: [Google Scholar](https://scholar.google.com/citations?user=...)
- Knuth, D. E., Morris, Jr, J. H., and Pratt, V. R.,** 1977, Fast pattern matching in strings, *SIAM journal on computing*, 6(2), 323-350. Doi: [10.1137/0206024](https://doi.org/10.1137/0206024)
- Kouzinopoulos, C. S., and Margaritis, K. G.,** 2014, Multiple pattern matching: Survey and experimental results, *Neural, Parallel, and Scientific Computation*, 22, 563-593. Url: <http://www.dynamicpublishers.com/Neural/NPSC2014/NPSC-21-2013-4/08-NPSC-34-SA-04.pdf>
- Kucherov, G., Salikhov, K., and Tsur, D.,** 2016, Approximate string matching using a bidirectional index, *Theoretical Computer Science*, 638, 145-158.
- Kumar, S., Singh, S., Khatoon, A., and Agarwal, S.,** 2019, A Multiple String and Pattern Matching Algorithm Using Context-Free Grammar, *Emerging Trends in Expert Applications and Security*, 97-102. Springer. Doi: [10.1007/978-981-13-2285-3_12](https://doi.org/10.1007/978-981-13-2285-3_12)

KAYNAKLAR DİZİNİ (devam)

- Lai, W. S., Wu, C. C., Lai, L. F., and Sie, M. C.,** 2019, Two-phase PFAC algorithm for multiple patterns matching on CUDA GPUs, *Electronics*, 8(3), 270. Doi: [10.3390/electronics8030270](https://doi.org/10.3390/electronics8030270)
- Lecroq, T.,** 2007, Fast exact string matching algorithms, *Information Processing Letters*, 102(6), 229-235. Doi: [10.1016/j.ipl.2007.01.002](https://doi.org/10.1016/j.ipl.2007.01.002)
- Levenshtein, V.,** 1965, Binary codes capable of correcting spurious insertions and deletion of ones, *Problems of information Transmission*, 1(1), 8-17.
- Lin, C. H., Li, J. C., Liu, C. H., and Chang, S. C.,** 2017, Perfect hashing based parallel algorithms for multiple string matching on graphic processing units, *IEEE Transactions on Parallel and Distributed Systems*, 28(9), 2639-2650. Doi: [10.1109/TPDS.2017.2674664](https://doi.org/10.1109/TPDS.2017.2674664)
- Mane, S. U., and Pangu, K. H.,** 2016, Disease diagnosis using pattern matching algorithm from DNA sequencing: a sequential and GPGPU based approach, *In Proceedings of the International Conference on Informatics and Analytics*, 1-5. Doi: [10.1145/2980258.2980392](https://doi.org/10.1145/2980258.2980392)
- Muzny D.M, Scherer S.E, Kaul R, et al.,** “Homo sapiens chromosome 3, GRCh38.p13 Primary Assembly”, https://www.ncbi.nlm.nih.gov/nuccore/NC_000003.12 (2006), (Erişim tarihi: 04.02.2020)
- Myers, G.,** 1998, A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Annual Symposium on Combinatorial Pattern Matching* (1-13). Doi: [10.1007/BFb0030777](https://doi.org/10.1007/BFb0030777)
- Najam, M., Rasool, R. U., Ahmad, H. F., Ashraf, U., and Malik, A. W.,** 2019, Pattern matching for dna sequencing data using multiple bloom filters. *BioMed research international*. Doi: [10.1155/2019/7074387](https://doi.org/10.1155/2019/7074387)
- Navarro, G., and Raffinot, M.,** 2002, Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences, *Cambridge university press*.
- Nunes, L. S. N., Bordim, J. L., Ito, Y., and Nakano, K.,** 2020, A Rabin-Karp Implementation for Handling Multiple Pattern-Matching on the GPU, *IEICE Transactions on Information and Systems*, 103(12), 2412-2420. Doi: [10.1587/transinf.2020PAP0002](https://doi.org/10.1587/transinf.2020PAP0002)

KAYNAKLAR DİZİNİ (devam)

- Oizumi, T., Kai, T., Mieno, T., Inenaga, S., & Arimura, H.,** 2022, Cartesian Tree Subsequence Matching. arXiv preprint arXiv:2202.04349.
- Policriti, A., and Prezza, N.,** 2015, Fast randomized approximate string matching with succinct hash data structures, *BMC bioinformatics*, 16(9), 1-8. Doi: [10.1186/1471-2105-16-S9-S4](https://doi.org/10.1186/1471-2105-16-S9-S4)
- Park, S. G., Bataa, M., Amir, A., Landau, G. M., and Park, K.,** 2020, Finding patterns and periods in cartesian tree matching. *Theoretical Computer Science*, 845, 181-197. Doi: [10.1016/j.tcs.2020.09.014](https://doi.org/10.1016/j.tcs.2020.09.014)
- Rasool, A., Ahmed, G. F., Barskar, R., and Khare, N.,** 2019, Efficient multiple pattern matching algorithm based on BMH: MP-BMH, *Int. Arab J. Inf. Technol.*, 16(6), 1121-1130. Url: <https://iajit.org/PDF/November%202019,%20No.%206/15585.pdf>
- SaiKrishna, V., Rasool, A., and Khare, N.,** 2012, String matching and its applications in diversified fields, *International Journal of Computer Science Issues (IJCSI)*, 9(1), 219.
- Sankoff, D.,** 1983, Time warps, string edits, and macromolecules, *The Theory and Practice of Sequence Comparison*, Reading.
- Singh, M., and Sharma, V.,** 2017, ASCII based Sequential Multiple Pattern Matching Algorithm for High Level Cloning, *International Journal Of Advanced Computer Science And Applications*, 271-276.Url: <https://pdfs.semanticscholar.org/df05/c9dda727a6ed18a3b840e1a3f53abbd71ee4.pdf>
- Song, S., Gu, G., Ryu, C., Faro, S., Lecroq, T., and Park, K.,** 2021, Fast algorithms for single and multiple pattern Cartesian tree matching, *Theoretical Computer Science*, 849, 47-63. Doi: [10.1016/j.tcs.2020.10.009](https://doi.org/10.1016/j.tcs.2020.10.009)
- Sukhanov, S., Wu, R., Debes, C., and Zoubir, A. M.,** 2020, Dynamic pattern matching with multiple queries on large scale data streams, *Signal Processing*, 171, 107402. Doi: [10.1016/j.sigpro.2019.107402](https://doi.org/10.1016/j.sigpro.2019.107402)
- Sunday, D. M.,** 1990, A very fast substring search algorithm, *Communications of the ACM*, 33(8), 132-142. Doi: [10.1145/79173.79184](https://doi.org/10.1145/79173.79184)

KAYNAKLAR DİZİNİ (devam)

- Takenaka, Y., Seno, S., and Matsuda, H.,** 2011, Perfect Hamming code with a hash table for faster genome mapping, *BMC genomics*, Vol. 12, No. 3, 1-11. Doi: [10.1186/1471-2164-12-S3-S8](https://doi.org/10.1186/1471-2164-12-S3-S8)
- Tarhio, J., and Peltola, H.,** 1997, String matching in the DNA alphabet, *Software: Practice and Experience*, 27(7), 851-861. Doi: [10.1002/\(SICI\)1097-024X\(199707\)27:7<851::AID-SPE108>3.CO;2-D](https://doi.org/10.1002/(SICI)1097-024X(199707)27:7<851::AID-SPE108>3.CO;2-D)
- Ukkonen, E.,** 1995, On-line construction of suffix trees. *Algorithmica*. 14 (3): 249–260. Doi: [10.1007/BF01206331](https://doi.org/10.1007/BF01206331)
- Upama, P. B., Khan, J. T., Zemim, F., Yasmin, Z., and Sakib, N.,** 2015, A new approach in pattern matching: codon detection in DNA and RNA using hash function (CDDRHF), *International Conference On Computer and Information Technology (ICCIT)*, 172-177. Doi: [10.1109/ICCITech.2015.7488063](https://doi.org/10.1109/ICCITech.2015.7488063)
- Wandelt, S., Bux, M., and Leser, U.,** 2014, Trends in genome compression, *Current Bioinformatics*, 9(3), 315-326. Doi: [10.2174/1574893609666140516010143](https://doi.org/10.2174/1574893609666140516010143)
- Wang, J. F., Li, Z. R., Cai, C. Z., and Chen, Y. Z.,** 2005, Assessment of approximate string matching in a biomedical text retrieval problem, *Computers in Biology and Medicine*, 35(8), 717-724. Doi: [10.1016/j.compbiomed.2004.06.002](https://doi.org/10.1016/j.compbiomed.2004.06.002)
- Wu, S., and Manber, U.,** 1994, A fast algorithm for multi-pattern searching, *University of Arizona. Department of Computer Science*, 1-11. Url: <https://www.cs.arizona.edu/sites/default/files/TR94-17.pdf>
- Xu, K., Yang, Z., Kang, P., Wang, Q., and Liu, W.,** 2019, Document-level attention-based BiLSTM-CRF incorporating disease dictionary for disease named entity recognition, *Computers in biology and medicine*, 108, 122-132. Doi: [10.1016/j.compbiomed.2019.04.002](https://doi.org/10.1016/j.compbiomed.2019.04.002)
- Yang, X., Han, G., Chen, J., and Cai, H.,** 2018, Finding correlated patterns via high-order matching for multiple sourced biological data, *IEEE Transactions on Biomedical Engineering*, 66(4), 1017-1025. Doi: [10.1109/TBME.2018.2866266](https://doi.org/10.1109/TBME.2018.2866266)
- Yildiz, B.,** 2019, High Performance Queries Using Compressed Bitmap Indexes, *European Conference on Parallel Processing*, 493-505. Doi: [10.1007/978-3-030-48340-1_38](https://doi.org/10.1007/978-3-030-48340-1_38)

KAYNAKLAR DİZİNİ (devam)

- Yildiz, B.**, 2020, Optimizing bitmap index encoding for high performance queries, *Concurrency and Computation: Practice and Experience*, e5943. Doi: [10.1002/cpe.5943](https://doi.org/10.1002/cpe.5943)
- Yu, X., and Liu, X.**, 2020, Mapping RNA-seq reads to transcriptomes efficiently based on learning to hash method, *Computers in biology and medicine*, 116, 103539. Doi: [10.1016/j.compbio.2019.103539](https://doi.org/10.1016/j.compbio.2019.103539)
- Zeng, P., Tan, Q., Meng, X., Shao, Z., Xie, Q., Yan, Y., and Xu, J.**, 2017, A multi-pattern hash-binary hybrid algorithm for URL matching in the HTTP protocol, *PloS one*, 12(4), e0175500. Doi: [10.1371/journal.pone.0175500](https://doi.org/10.1371/journal.pone.0175500)
- Zhang, B., Chen, X., Pan, X., and Wu, Z.**, 2009, High concurrence Wu-Manber multiple patterns matching algorithm, *International Symposium on Information Processing*, 404.
- Zhang, B., Chen, X., Ping, L., and Wu, Z.**, 2009, Address filtering based Wu-Manber multiple patterns matching algorithm, *Second International Workshop on Computer Science and Engineering*, 408-412. Doi: [10.1109/WCSE.2009.698](https://doi.org/10.1109/WCSE.2009.698)
- Zhang, M., Zhang, Y., and Sun, Y.**, 2021, Dictionary lookup with one genome evolution operation, *Concurrency and Computation: Practice and Experience*, 33(6). Doi: [10.1002/cpe.5840](https://doi.org/10.1002/cpe.5840)
- Zhen, C., and Di, W.**, 2008, Improving Wu-Manber: a multi-pattern matching algorithm, *IEEE International Conference on Networking, Sensing and Control*, 812-817. Doi: [10.1109/ICNSC.2008.4525327](https://doi.org/10.1109/ICNSC.2008.4525327)

TEŞEKKÜR

Tez çalışmam boyunca değerli görüş ve önerileriyle doğru yönde ilerlememi sağlayan, her konuda yardım ve ilgisini esirgemeyen değerli hocam ve doktora tez danışmanım Doç. Dr. Hasan BULUT hocama teşekkürlerimi sunarım. Tez izleme komitesi toplantılarındaki kıymetli görüş ve önerilerle çalışmama katkılar sağlayan Prof. Dr. Serdar KORUKOĞLU ve Prof. Dr. Muhammed Bahattin TANYOLAÇ hocalarıma da ayrıca teşekkür ederim. Eğitim ve iş hayatımda benden fikirlerini esirgemeyen Dr. Öğretim üyesi Mete YAĞANOĞLU hocama da ayrıca teşekkür ediyorum. Ek olarak, doktora eğitimim ve tez çalışmalarım süresince benden desteğini esirgemeyen Ege Üniversitesi Bilgisayar Mühendisliği ailesine de teşekkürü bir borç bilirim.

Her zaman olduğu gibi tüm öğrenim hayatım boyunca beni destekleyen, bu günlere gelmeme vesile olan çok kıymetli aileme teşekkürü bir borç bilirim.

ÖZGEÇMİŞ

1. KİŞİSEL BİLGİLER

Ad Soyad:
Uyruk:

Abdullah Ammar KARCIOĞLU
T.C.

2. EĞİTİM

Öğrenim Dönemi	Derece	Üniversite	Öğrenim Alanı
2018 - 2022	Doktora	Ege Üniversitesi	Bilgisayar Mühendisliği
2015 - 2018	Yüksek Lisans	Atatürk Üniversitesi	Bilgisayar Mühendisliği
2010 - 2015	Lisans	Ege Üniversitesi	Bilgisayar Mühendisliği

3. AKADEMİK DENEYİM

Görev Dönemi	Ünvan	Bölüm	Üniversite
2018 - 2022	Arş. Gör.	Bilgisayar Mühendisliği	Ege Üniversitesi
2017 - 2018	Arş. Gör.	Bilgisayar Mühendisliği	Atatürk Üniversitesi
2017 - 2017	Arş. Gör.	Bilgisayar Mühendisliği	Recep Tayyip Erdoğan Üniversitesi

4. YAYIN BİLGİLERİ

SCI veya SCI Expanded, SSCI, AHCI tarafından taranan dergilerde yayımlanan tam makale

- KARCIOĞLU, A. A., & Bulut, H. (2021). Improving Hash-q Exact String Matching Algorithm with Perfect Hashing for DNA Sequences. *Computers in Biology and Medicine*, 104292. Doi: [10.1016/j.combiomed.2021.104292](https://doi.org/10.1016/j.combiomed.2021.104292)
- Karcioglu AA, Bulut H. q-frame hash comparison based exact string matching algorithms for DNA sequences. *Concurrency Computat Pract Exper*. 2021;e6505. Doi: <https://doi.org/10.1002/cpe.6505>
- KARCIOĞLU, A. A., & Bulut, H. (2021). The WM-q multiple exact string matching algorithm for DNA sequences. *Computers in Biology and Medicine*, 104656. Doi: [10.1016/j.combiomed.2021.104656](https://doi.org/10.1016/j.combiomed.2021.104656)

Ulusal hakemli dergilerde yayımlanan tam makale

- Karcioglu, A.A., Tanışman, S. & Bulut, H. (2021). Türkiye'de COVID-19 Bulaşısının ARIMA Modeli ve LSTM Ağı Kullanılarak Zaman Serisi Tahmini. *Avrupa Bilim ve Teknoloji Dergisi*, Ejosat Özel Sayı 2021 (RDCONF), 288-297. Doi: [10.31590/ejosat.1039394](https://doi.org/10.31590/ejosat.1039394)

2. Tanışman, S., Karcioğlu, A. A., Ugur, A. & Bulut, H. (2021). LSTM Sinir Ağı ve ARIMA Zaman Serisi Modelleri Kullanılarak Bitcoin Fiyatının Tahminlenmesi ve Yöntemlerin Karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*, Ejosat Özel Sayı 2021 (RDCONF), 514-520. Doi: [10.31590/ejosat.1039890](https://doi.org/10.31590/ejosat.1039890)

BİLDİRİLER (Tam Metin)

- 1) Karcioğlu, A. A., & Aydin, T. (2019, April). Sentiment analysis of Turkish and english twitter feeds using Word2Vec model. In *2019 27th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE. Doi: 10.1109/SIU.2019.8806295 <https://ieeexplore.ieee.org/document/8806295>
- 2) Karcioğlu, A. A., & Yaşa, A. C. (2020, October). Automatic Summary Extraction in Texts Using Genetic Algorithms. In *2020 28th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE. Doi: [10.1109/SIU49456.2020.9302205](https://doi.org/10.1109/SIU49456.2020.9302205)
- 3) AA KARCIOĞLU, T AYDIN, GT ÖZYER (2018). A Comparison of Performance Metrics of Turkish Twitter Messages Using Text Representations. 1. INTERNATIONAL TECHNOLOGICAL SCIENCES AND DESIGN SYMPOSIUM 1 (1), 433-444
- 4) Karcioğlu, A. A., & Bulut, H. (2021, September). Performance Evaluation of Classification Algorithms Using Hyperparameter Optimization. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 354-358). IEEE. Doi: [10.1109/UBMK52708.2021.9559003](https://doi.org/10.1109/UBMK52708.2021.9559003)

5. HAKEMLİKLER

Mühendislik Bilimleri ve Tasarım Dergisi, 2021

Yüksek Lisans tezi İngilizce ve Türkçe twitter mesajlarının Word2Vec modeli ile sınıflandırılması, 16.08.2018

6. Bilimsel Araştırma Projelerinde (Bap) Görev Alma (Araştırmacı)

Öncelikli Alan (BAP-ÖNAP, Yüksek Bütçeli): Cicer ve Lens türlerinin kloroplast DNA dizilerinin Next Generation Sequencing ile sekanslanması ve genom organizasyonlarının belirlenerek karşılaştırmalı genom analizlerinin gerçekleştirilmesi, 09.09.2019-...