**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

**YÜKSEK LİSANS TEZİ**

**Ebubekir TOPAK**

# SERBEST LİE CEBİRLERİNDE HESAPLAMALAR

**MATEMATİK ANABİLİM DALI**

**ADANA, 2008**

**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# SERBEST LİE CEBİRLERİNDE HESAPLAMALAR

**Ebubekir TOPAK**

**YÜKSEK LİSANS TEZİ**
**MATEMATİK ANABİLİM DALI**

Bu tez ……/……/2008 Tarihinde Aşağıdaki Jüri Üyeleri Tarafından Oybirliği/Oyçokluğu İle Kabul Edilmiştir.

| | | |
|---|---|---|
| İmza……...… | İmza…….. | İmza……..……. |
| Yrd.Doç.Dr. Ahmet TEMİZYÜREK | Yrd.Doç.Dr Ela AYDIN | Yrd.Doç.Dr. Hüseyin BİLGİÇ |
| DANIŞMAN | ÜYE | ÜYE |

Bu tez Enstitümüz Anabilim Dalında hazırlanmıştır.

Kod No:

**Prof. Dr. Aziz ERTUNÇ**
**Enstitü Müdürü**
**İmza ve Mühür**

**ÖZ**

**YÜKSEK LİSANS TEZİ**

# SERBEST LİE CEBİRLERİNDE HESAPLAMALAR

**Ebubekir TOPAK**

**ÇUKUROVA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**
**MATEMATİK ANABİLİM DALI**

Danışman : Yrd.Doç.Dr. Ahmet TEMİZYÜREK
Yıl : 2008, Sayfa: 113
Jüri : Yrd.Doç.Dr. Ahmet TEMİZYÜREK
Yrd.Doç.Dr. Ela AYDIN
Yrd.Doç.Dr. Hüseyin BİLGİÇ

Bu çalışmada bir $L$ Lie cebirinin evrensel enveloping cebiri incelenerek bir bazının yapısı verilmiştir. Bundan başka Lie cebirlerinde bilinen bazı algoritmalar için daha önce yapılan bilgisayar proğramlarından farklı bir yolla bilgisayar programı yazılmıştır. Bu çalışmada bir Serbest Lie cebirlerinde bilinen bazı algoritmalar için bilgisayar programı yazılmıştır. $F$ bir serbest Lie cebiri olduğunda $F$ Lie cebirinin Hall bazını, $F$ nin $F_n$, alt merkezi, $F^{(n)}$ türetilmiş ve $F_{n_1,n_2,\mathbf{K},n_k}$ polycentral serilerinin terimleri için serbest üreteç kümelerini ve bazlarını veren bilgisayar programı yapılmıştır. Ayrıca sonlu takdimli bir Lie cebirinin doğurayları belli olan bir $B$ alt cebiri için modülo$B$ bazını hesaplayan program yapılmıştır.

**Anahtar Kelimeler:** Hall bazı, serbest Lie cebirleri, Lyndon kelimeleri, dairesel kelimeler, Gröbner bazı.

**ABSTRACT**

**MSc THESIS**

**COMPUTATION IN FREE LIE ALGEBRAS**

**Ebubekir TOPAK**

**DEPARTMENT OF MATHEMATICS**
**INSTITUTE OF NATURAL AND APPLIED SCIENCES**
**UNIVERSITY OF ÇUKUROVA**

Supervisor : Assist.Prof.Dr.Ahmet TEMİZYÜREK
Year : 2008, Pages: 113
Jury : Assist.Prof.Dr.Ahmet TEMİZYÜREK
Assist.Prof.Dr. Ela AYDIN
Assist.Prof.Dr. Hüseyin BİLGİÇ

In this study we have investigated the universal enveloping algebra U(*L)* of a Lie algebra *L* and the structure of its bases has been given. Furthermore, we have given further computer programs different from former programs for some well known algorithm about Lie algebras. Let $F$ be a Free Lie algebra, we have given a computer program which calculate the Hall bases of a free Lie algebra of $F$, free generators sets and bases for the terms of the lower central series $F_n$, the polycentral series $F_{n_1,n_2,\mathbf{K},n_k}$ and derived series $F^{(n)}$ of $F$. .Let $L$ be a finitely presented Lie algebra and $B$ be a subalgebra of $L$ which its generators are given. Then we give a computer program finding bases in $L$ modulo the subalgebra $B$.

**Key Words:** Hall base, free Lie algebra, Lyndon words, circular words, Gröbner base.

**TEŞEKKÜR**

Bu çalışmanın hazırlanması sırasında bilgi ve tecrübeleriyle beni aydınlatan, her aşamasında yardımlarını esirgemeyen, değerli zamanlarını ayırarak çalışmanın tamamlanmasını sağlayan, bilgisi ve kişiliğiyle örnek aldığım saygıdeğer danışmanım Yrd.Doç.Dr. Ahmet TEMİZYÜREK'e sonsuz teşekkürlerimi sunarım. Ayrıca saygı değer hocam Prof.Dr. Naime EKİCİ'ye, her zaman beni cesaretlendiren sevgili arkadaşım ve hocam Yrd.Doç.Dr. Ela AYDIN'a ve tüm Matematik Bölümü akademik personeline teşekkür ederim. Tezimin yazımında yardımlarını esirgemeyen değerli mesai arkadaşım Ertuğrul UYGUN'a ve ağabeyim Mehmet TOPAK'a teşekkür ederim. Bugüne kadar desteklerini esirgemeyen ve her zaman yanımda olan anneme, babama teşekkür ederim. Özellikle çalışmamın hazırlanması sürecinde desteğini esirgemeyen sevgili eşim Yasemin'e ve dünya tatlısı yavrularıma sonsuz teşekkürlerimi sunarım.

**İÇİNDEKİLER** SAYFA

## SİMGELER VE KISALTMALAR DİZİNİ

| | |
|---|---|
| $[\ ,\ ]$ | : Lie cebirinde braket çarpımı |
| $N < L$ | : $L$ nin bir $N$ ideali |
| $L/N$ | : Bölüm cebiri |
| $L_2 = [L, L]$ | : $L$ nin türetilmiş ( komütatör) alt cebiri |
| $L$ | : Lie cebiri |
| $F$ | : Serbest Lie cebiri |
| $X$ | : Üreteç Kümesi |
| $X^*$ | : $X$ in elemanlarının serbest çarpımı |
| $F\langle X\rangle$ | : $X$ tarafından üretilen serbest Lie cebiri $F(X)$ olarak da gösterilir |
| $K = A$ ? | : $K$ üzerinde $A$ nın tüm serilerinin kümesi |
| $H$ | : Hall Kümesi |
| $l(a)$ | : $a$ kelimesinin uzunluğu |
| $m$ | : Möbius fonksiyonu |
| $F_{(m)}$ | : $F$ nin $m$-inci alt merkezi serisinin terimini |
| $C_m$ | : $F_{(m)}$ için bir serbest üreteç kümesi |
| $ld(f)$ | : $f$ nin en yüksek dereceli (leading) terimi |
| $LM(h)$ | : $h$ nin en büyük derceli monomiali (tek terimlisi) |
| $\overline{x}_i$ | : $\overline{x}$ in $f$ altındaki görüntüsü $f(x_i) = \overline{x}_i$ |
| $U(L)$ | : $L$ nin Universal Enveloping cebiri |
| $j$ | : Euler fi fonksiyonu |

# 1. GİRİŞ

Kompozisyon ve parçalanma metotları, Campbel-Housdorf formülünü içeren metotlar ve son zamanlarda gelişen manifoldlar üzerinde diferansiyel denklemlerin integrallenebilmesi için Lie grup metotları gibi, Lie cebirlerin de hesaplama içeren bir çok algoritma vardır. Ayrıca Serbest Lie cebirlerinde birçok algoritma GAP paket programını kullanarak bilgisayarlar yardımıyla kolayca çalıştırılabilmektedir. A.M. Cohen, W. A. De Graaf nilpotent olmayan bir Lie cebirinde nilpotent olmayan bir elemanı belirleyen algoritma ve Cartan alt cebirlerini belirleyen algoritmayı bilgisayar kullanarak hızlı bir şekilde çalıştığını göstermişlerdir.

Lyndon kelimeleri bir serbest Lie cebirinin n-inci homojen bileşeninin bazını kurmada kullanılmaktadır. J.Sawada ve F. Ruskey (2003) $n$- uzunluklu tüm Lyndon kelimeleri için standart braketleri etkin olarak doğuran bir algoritma geliştirmişlerdir. (Bu braketler serbest Lie cebirinin $n$-inci homojen bileşeninin bazını teşkil eder.) Serbest Lie cebirlerinde elde edilen algoritmaların bilgisayar ortamında çalıştırılabilmeleri en az algoritma geliştirmek kadar önemlidir.

Bu çalışmada Serbest Lie cebirlerinde verilen bazı algoritmaları inceleyerek bu algoritmalar için bilgisayar programı yazılmıştır.

İkinci bölümde tez içinde kullanılacak olan tanım, teorem ve bazı gösterimlerden bahsedilmiştir.

Üçüncü bölümde $L$ bir Lie cebiri olduğunda $L$ nin evrensel enveloping cebirinin yapısı incelenmiş olup Evrensel enveloping cebirin bir bazı için Poincaré – Birkhoff – Witt Teoremi verilmiştir.

Dördüncü bölümde Serbest Lie cebiri ile dairesel kelimeler arasındaki ilişkileri incelenmiştir. Witt formülü ile verilen homojen boyutun ve ilkel kolyelerin sayısının eşitliğini söyleyerek kolyelerin ve ilkel kolyelerin sayısı hesaplanmıştır. Daha sonra Hall kelimeleri ile ilkel kolyeler arasındaki birebir ve örten bağıntı incelenmiştir. Lyndon kelimelerini üreten bir algoritmayı inceleyerek verilen bir kelimenin Lyndon kelimelerindeki parçalanışı hesaplanmıştır. Hall kelimeleri içindeki bir kelimenin parçalanışı, kelimeler ile ilkel kolyelerin multisetleri arasında birebir ve örten bağıntıyı ifade ettiği gösterilmiştir.

Beşinci bölümde Lyndon kelimelerini daha değişik bir yolla algoritmik olarak veren bir teknikten badsedilmiştir. J. M. Champarnaud, G. Hansel ve D. Perin (2003) tarafından verilen "Sabit uzunluklu iptaledilemez kelimeler" ile belirlendiği gösterilmiştir.

Tezin son bölümünde, daha önce yapılan bilgisayar programlarından farklı bir yolla bir serbest Lie cebirinin Hall bazını, alt merkezi serisinin terimleri için üreteç kümeleri ve bazlarını bulan bir bilgisayar programı yapılmıştır. ( En çok yirmi uzunluktaki kelimelere kadar incelenmiştir). Ayrıca Ela AYDIN (1997) tarafından verilen sonlu takdimli bir $L$ Lie cebirinin herhangi bir $B$ alt cebirinin üreteçleri bilindiğinde koboyutu veren algoritmanın bilgisayar programı yazılmıştır.

# 2. TEMEL TANIM VE TEOREMLER

Bu çalışmamız boyunca aksi belirtilmedikçe cebirlerin tamamını bir $k$ cismi üzerinde kabul edeceğiz.

## 2.1. Lie Cebiri

**Tanım 2.1.1:** $L$, $k$ cismi üzerinde herhangi bir vektör uzayı olsun. Her $x, y \in L$ için $(xy) = xy$ $L$ de bilineer çarpma olsun. Bu çarpım her $x, y \in L$ için

1. $(xx) = 0$ $\forall x \in L$ *için*
2. $(x(yz)) + (z(xy)) + (y(zx)) = 0$ $\forall x, y, z \in L$ *için*

Koşullarını sağlıyorsa $L$ ye bu çarpımla birlikte $k$ cismi üzerinde bir Lie cebiri denir. Eğer $L$ vektör uzayı olarak sonlu boyutlu ise Lie cebiri olarak da sonlu boyutludur. Boyutu 1 olan Lie cebirleri abelyendirler. 1-boyutlu bir tek Lie cebiri vardır. 2-boyutlu, izomorfik olmayan sadece iki Lie cebiri vardır. Ayrıca Lie cebirleri Jacobi özdeşliğinden dolayı birleşmeli değildirler.

2. denkleme Jakobi özdeşliği denir. Lie çarpımı $xy = [x, y]$ şeklinde gösterilir.

$$[x + y, x + y] = 0$$

koşulu

$$[x, y] = -[y, x]$$

olmasını gerektirir.

$L$ bir Lie cebiri ve $M \subseteq L$ olsun. $M$, $L$ nin bir alt uzayı ise (Bir vektör uzayı gibi düşüneceğiz), $M$ Lie çarpımı altında kapalı ise $M$ ye $L$ nin bir Lie alt cebiri denir ve $M \leq L$ ile gösterilir.

**Tanım 2.1.2:** Eğer $N$, $L$ nin bir alt cebiri ve $NL \subseteq N$ (veya $xy = -yx$ olduğundan $LN \subseteq N$ ) ise $N$ ye $L$ nin bir ideali denir ve $N < L$ ile gösterilir.

**Tanım 2.1.3:** $L$, bir Lie cebiri ve $M < L$ olsun. $L/M$ bölüm cebiri aşağıdaki şekilde tanımlanır

$$L/M = \{M + x \mid \ x \in L\}$$

Olmak üzere $L/M$ içerisinde çarpma ve toplama işlemini;

Eğer $x+M, y+M \in L/M$ için

$$(x+M)+(y+M)=(x+y)+M$$
$$(x+M)(y+M)=(xy)+M$$

olarak tanımlayalım. $L/M$ bu işlemlerle bir Lie cebiridir. Bu cebire $L$ nin $M$ ile bölüm cebiri denir.

$L$ bir Lie cebiri ve $x, y \in L$ ise $[a,[a,\mathbf{L}\,[a,b]] = a^n b$ ile gösterilir.

**Tanım 2.1.4:** $L$ ve $M$ aynı $k$ cismi üzerinde iki Lie cebiri olsun. Bir $a : L \to M$, bir lineer dönüşümü $x, y \in L$ için;

$$a\,([x,y]) = [a(x), a(y)]$$

oluyorsa $a$ ya bir Lie homomorfizmi denir. Eğer bu dönüşüm birebir ve örten ise izomorfizm denir. Eğer $a$, $L$ üzerinde bir izomorfizm ise o zaman $a$ ya $L$ nin otomorfizmi denir.

Genellikle homomorfizm teoremleri Lie cebiri homomorfizmleri için de doğrudur.

**Teorem 2.1.5 (Stewart, 1970) :** $L$, $M$ bir $k$ cismi üzerinde iki Lie cebiri ve $a : L \to M$ bir homomorfizm olsun. O zaman aşağıdakiler doğrudur.

i) $a(L) \le M$ ; $\quad Kernel(a) < L$ ve $L/Kernel(a) \cong a(L)$

ii) $H, K < L$ ve $K < H$ ise $(L/K)/(H/K) \cong L/H$

iii) $H < K$, $K < L$ ise $H < (H + K)$ ve $(H \cap K) < K$ ve $(H+K)/K \cong K/(H \cap K)$

iv) $\Phi : L \to (L/H)$ doğal homomorfizmdir. $H < L$ ise $L$ nin idealleri ile $L/H$ nin idealleri arasında bire bir ve örten bir ilişki vardır.

**Tanım 2.1.6:** $n = 1,2,\mathbf{K}$ için $L_n$ terimleri tümevarımla aşağıdaki şekilde tanımlanır:

$$L_1 = L, \quad L_{n+1} = [L_n, L]$$

Böylece $L$ nin ideallerinin azalan bir serisi elde edilir.

$$L = L_1 \supseteq L_2 \supseteq \ldots \supseteq L_n \supseteq L_{n+1} \supseteq \ldots$$

Bu seriye $L$ nin alt merkezi serisi denir.

$L$ nin $L_2 = [L, L]$ alt cebirine türetilmiş ( komütatör) alt cebir denir.

Eğer $L_{k-1} \neq \{0\}$ ve $L_k = \{0\}$ olacak şekilde bir $k$ pozitif tam sayısı varsa $L$ ye $k$ – yıncı dereceden nilpotent Lie cebiri denir. Eğer $L$ 2–inci dereceden nilpotent yani $[L,L] = 0$ ise $L$ ye abelyen denir.

**Tanım 2.1.7:** $n = 1,2,\mathbf{K}$ için $L^n$ terimleri tümevarımla aşağıdaki şekilde tanımlanır:

$$L^1 = L, \quad L^{n+1} = [L^n, L^n]$$

Böylece $L$ nin ideallerinin azalan bir serisi elde edilir.

$$L = L^1 \supseteq L^2 \supseteq \ldots \supseteq L^n \supseteq L^{n+1} \supseteq \ldots$$

Bu seriye $L$ nin türetilmiş serisi denir.

Eğer $L^{k-1} \neq \{0\}$ ve $L^k = \{0\}$ olacak şekilde bir $k$ tam sayısı varsa $L$ ye $k$–yıncı dereceden çözülebilir Lie cebiri denir.

**Tanım 2.1.8:** $i = 1,2,\mathbf{K},k,\mathbf{K}$ için $n_i \geq 1$ olmak üzere pozitif tamsayıların bir $\{n_1, n_2, \mathbf{K}, n_k, \mathbf{K}\}$ dizisi için $L$ nin polisentral serisi aşağıdaki şekilde tanımlanır:

$L_{n_1}$ ; $L$ nin alt merkezi serisinin $n_1$ - inci terimi

$L_{n_1,\ldots,n_i,n_{i+1}} = (L_{n_1,\ldots,n_i})_{n_{i+1}}$ , $L_{n_1,\cdots,n_i}$ nin alt merkezi serisinin $n_{i+1}$-inci terimi olsun. Böylece elde edilen

$$L \supseteq L_{n_1} \supseteq L_{n_1,n_2} \supseteq \cdots \supseteq L_{n_1,\ldots,n_i} \supseteq L_{n_1,\ldots,n_i,n_{i+1}} \supseteq \cdots$$

serisine $L$ nin polisentral serisi denir.

Eğer $L_{n_1,\ldots,n_k} = \{0\}$ ve $n_i$ lerin hiçbiri bu eşitlik sağlanacak şekilde daha küçük pozitif sayılarla değiştirilemiyor ise $L$ ye $\{n_1, \ldots, n_k\}$ dizisine göre polinilpotent Lie cebiri denir.

Eğer $n_1 = n_2 = 2$ ve $L_{2,2} = \{0\}$ ise $L$ ye metabelyen Lie cebiri denir.

## 2.2. Serbest Lie Cebirleri

**Tanım 2.2.1:** $X$ herhangi bir küme, $F$ bir Lie cebiri ve $i: X \to F$ bir dönüşüm olsun. Her $B$ Lie cebiri ve her $a : X \to B$ dönüşümü için $a = h\,i$ olacak şekilde bir tek $h : F \to B$ Lie homomorfizmi varsa $(F,i)$ çiftine $X$ üzerinde bir serbest Lie cebiri denir. Bunu aşağıdaki diyagramla ifade ederiz:

$$\begin{array}{ccc} X & \xrightarrow{\ i\ } & F \\ {\scriptstyle a}\downarrow & \swarrow_{\,h\ \text{bir tek}} & \\ B & & \end{array}$$

$X$ herhangi bir küme olduğunda $X$ üzerinde bir serbest Lie cebiri aşağıdaki gibi kurulur.

Her bir pozitif $n$ tamsayısı için $X_n$ bir kümesini aşağıdaki gibi tanımlayalım:

$$X_1 = X$$

$$X_{n-1} = \bigcup_{p=1}^{n-1} (X_p \times X_{n-p}) \qquad (2.2.1)$$

buradaki "$\times$" işlemi kümelerdeki kartezyen çarpımdır. $M(X) = \bigcup_{n=1}^{\infty} X_n$ olsun. Her $a,b \in M(X)$ için $n = p+q$ olduğunda $a \in X_p$ ve $b \in X_q$ ise $(a,b) \in (X_p \times X_q)$ dur.

O zaman $(a,b)\in(X_p\times X_{n-p})$, $X_n$ için $X_{n-1}=\bigcup_{p=1}^{n-1}(X_p\times X_{n-p})$ (2.2.1) deki birleşimin bileşenlerinden biridir. $(a,b)$ nin $(X_p\times X_{n-p})\to X_n$ e kanonik injeksiyon altındaki görüntüsünü $(ab)$ olarak gösterelim. O zaman $\forall a,b\in M(X)$ için $(a.b)$ çarpımını tanımlayabiliriz. $M(X)$ in $a$ elemanının uzunluğu $p$ tamsayısıdır öyle ki $a\in X_p$ dir ve $l(a)$ ile gösterilir. Bu tanıma göre uzunluğu 1 olan elemanlar $X$ in elemanlarıdır. $a,b\in M(X)$ için;

$$l(ab)=l(a)+l(b)$$

Uzunluğu $\geq 2$ olan elemanlar, $a$ ve $b$ nin uzunluğu $c$ nin uzunluğundan daha küçük olmak üzere,

$$c=(ab)$$

şeklindedir.

$k$ herhangi bir cisim olmak üzere, $k$ üzerinde bazı $M(X)$ olan bir vektör uzayı, $M(X)$ deki elemanların $k$ – lineer kombinasyonlarından oluşur. $M(X)$ deki çarpma, bu vektör uzayının tamamına genişletilebilir. Böylece $k$ üzerinde sonlu boyutlu olmayan ve birleşmeli olmayan serbest bir cebir elde edilmiş olur. Bu cebire $N(X)$ diyelim. $N(X)$ içinde

$$Q(a) = aa$$

$$J(a,b,c) = a(bc) + b(ca) + c(ab)$$

şeklindeki elemanlarla doğrulan ideal $A$ olsun. O zaman $F(X) = N(X)/A$ bölüm cebiri $X$ üzerinde bir serbest Lie cebiridir. $X$ kümesine $F(X)$ için bir serbest üreteç kümesi denir. Bundan böyle $F(X)$ i kısaca $F$ ile göstereceğiz.

**Tanım 2.2.2 :** $F$ bir serbest cebir olsun. $F$ de her bir terimi aynı dereceden olan elemana homojen eleman denir. Örneğin: $X=\{a,b,c\}$ ve $F(X)$ serbest cebir olsun. $u=(ab)+(bc)$ elemanı ikinci dereceden homojen elemandır.

**Teorem 2.2.3 (Bourbaki, 1972):** $h : N(X) \to F(X)$ kanonik dönüşüm ve $f$ de $h$ nin $X$ e kısıtlaması olsun. $L$ herhangi bir Lie cebiri olmak üzere her $f : X \to L$ dönüşümü $f = gf$ olacak şekilde bir tek $g : F(X) \to L$ homomorfizmi vardır.

$$\begin{array}{ccc} X & \xrightarrow{f} & L \\ & & \uparrow g \\ N(X) & \xrightarrow{h} & F(X) \end{array}$$

**Tanım 2.2.4:** $L$ bir Lie cebiri ve $A = \{a_i \mid i \in I\}$ $L$ nin elemanlarının bir ailesi olsun. $F(I)$, $I$ üzerinde serbest bir Lie cebiri olsun. Eğer

$$s : i \to a_i$$

olacak şekilde $F(I)$ dan $L$ ye içine bir $s$ homomorfizmi varsa ve

i) $s$ örtense $A$ ya $L$ nin üreteç kümesi;

ii) $s$ birebir – örten ise $L$ ye serbest Lie cebiri denir.

$L$ serbest ise $A$ ya $L$ nin serbest üreteç kümesi denir. $A$ sonlu ise $L$ ye sonlu üreteçli serbest Lie cebiri ya da sonlu üretilmiş serbest Lie cebiri denir.

Tanıma göre $F$ bir serbest Lie cebiri içindeki herhangi iki serbest üreteçli iki küme, aynı kardinaliteye sahiptir. $F$ nin serbest üreteçli bir kümesinin kardinalitesine $F$ nin rankı diyeceğiz.

**Tanım 2.2.5:** Eğer bir $u \in F$ elemanı $F$ nin bir serbest üreteç kümesine aitse $u$ ya primitif eleman denir.

**Teorem 2.2.6 (Shirsov, 1958) :** Bir serbest Lie cebirinin her alt cebiri de serbesttir.

## 2.3. Serbest Lie Cebirinin Hall Bazları

Daha önce tanımladığımız $M(X)$ kümesi, $k$ üzerinde bir vektör uzayı gibi düşünülen asosyatif olmayan $N(X)$ serbest cebiri için bir bazdır. $M(X)$, $N(X)$ içinde lineer bağımsız olmasına rağmen, $F(X)$ içinde lineer bağımsız değildir.

Örneğin $a,b\in M(X)$ için $ab$ ve $ba$ formundaki elemanlar $N(X)$ de lineer bağımsız olmakla beraber $F(X)$ de lineer bağımlıdırlar. $(ab)=-(ba)$ dır. Bu nedenle $F(X)$ için $M(X)$ bir baz olamaz.

Bu bölümde $F(X)$ i bir vektör uzayı olarak düşündüğümüzde $F(X)$ serbest Lie cebiri için bir baz kümesi kuracağız.

$M^n(X)$ ile $M(X)$ içindeki uzunluğu $n$ olan elemanları gösterelim. Açıkça görülür ki

$$M^1(X)=X$$

dir.

**Tanım 2.3.1:** Bir Hall, $H\subset M(X)$ alt kümesi aşağıdaki gibi tanımlanır.

i) $X\subseteq H$ ve $X$ e tam sıralama verilmiş olsun.

ii) $H\cap M^2(X)$ kümesi $x,y\in X$ iken $x<y$ olmak üzere $(xy)$ formundaki elemanlardan meydana gelir.

iii) $H\cap M^m(X)$ $m=1,2,3,...,n-1$ için tanımlanmış ve uzunluğu koruyan bir sıralama verilmiş olsun. Yani $u,v\in M(X)$ ve $l(u)<l(v)$ ise $u<v$ yazalım. Aynı uzunluktaki elemanları keyfi olarak yazalım. O zaman

$$H\cap M^n(X)=\left\{a(bc)\middle| a,b,c,(bc)\in \bigcup_{k=1}^{n-1}\left(H\cap M^k(X)\right), i\le k\le n-1,\ a\ge b<c,\ a<bc\right\}$$

ve $H=\bigcup_{n=1}^{\infty}\left(H\cap M^n(X)\right)$ olsun

$$H_n=H\cap M^n(X)$$

dersek o zaman $H=\bigcup_{n=1}^{\infty}H_n$ kümesi $F(X)$ in bir bazıdır. Buna Hall bazı denir ve $H_1,H_2,...,H_n$ kümelerine Hall kümeleri denir.Verilen bir $X$ kümesi üzerinde farklı Hall kümeleri tanımlanabilir.

Kısaca $X\neq\varnothing$ olmak üzere Hall bazı şöyle ifade edilir.

$H_1=X$ , $X$ e tam sıralama verilir.

$$H_2 = \{(xy) | x, y \in X,\ x < y\}$$

$$H_n = \{a(bc) |\ a,b,c,bc \in (H_1 \cup H_2 \cup ... \cup H_{n-1}),\ a \geq b < c,\ a < bc\}$$

**Teorem 2.3.2 (Bourbaki, 1972):** $F$ , $X$ kümesi üzerinde serbest bir Lie cebiri olsun. $X$ kümesi kümesi üzerindeki bir Hall kümesi $F$ nin vektör uzayı olarak bir bazıdır.

$F$ serbest Lie cebirinin bir elemanı, baz elemanlarının bir lineer kombinasyonu olarak yazılabilir.

$F$ , $X$ üzerinde bir serbest Lie cebiri olsun. $F$ nin Hall bazını $H^X$ ile göstereceğiz. Hall kümesindeki herhangi bir kelimenin uzunluğu aksi belirtilmedikçe kelimenin kompozisyonundaki harflerin sayısıdır. Bunu $u \in H$ için $X - l(u)$. $H$ kümesinin sırasını aşağıdaki gibi belirleyeceğiz.

i) $X$ keyfi seçilmiş bir sıradadır

ii) $H$ nin iki elemanı $u = u_1 u_2$ ve $v = v_1 v_2$ olsun. Eğer $l(u) < l(v)$ ise $u < v$ yazılır.

**Örnek 2.3.3:** $X = \{a,b,c\}$ olsun.

$H_1 = X$ ve kabul edelim ki $a < b < c$ olsun.

$H_2 = \{ab, ac, bc\}$ ve kabul edelim ki $ab < ac < bc$ olsun.

$H_3 = \{a(ab), b(ab), c(ab), a(ac), b(ac), c(ac), b(bc), c(bc)\}$ ve kabul edelim ki yazılış sırası gerçek sırası olsun.

$$\begin{aligned} H_4 = \{ & a(a(ab)), b(a(ab)), c(a(ab)), b(b(ab)), c(b(ab)), c(c(ab)), \\ & a(a(ac)), b(a(ac)), c(a(ac)), b(b(ac)), c(b(ac)), c(c(ac)), \\ & b(b(bc)), c(b(bc)), c(c(bc)), (ab)(ac), (ab)(bc), (ac)(bc)\} \end{aligned}$$

Eğer $X$ sonlu bir küme ise $H_n$ in içindeki elemanların sayısını belirlemenin bir metodu vardır. Önce aşağıdakine ihtiyacımız var.

**Tanım 2.3.4:** $N$ pozitif tamsayıları gösterir. Möbius **fonksiyonu** $m : N \to \{1, 0, -1\}$ şöyle tanımlanır:

$$m(n) = \begin{cases} m(n) = 0, & \text{Eğer } n \text{ bir asal sayının karesi ile bölünebiliyorsa} \\ m(1) = 1, & \\ m(n) = (-1)^k, & \text{Eğer } n = p_1.p_2...p_k \quad (p_i \text{ ler asal sayı}) \end{cases}$$

**Örnek 2.3.5:**

$m(1) = 1$

$m(2) = (-1)^1 = -1,$ $\quad 2 = 2,$ $\quad k = 1$

$m(3) = (-1)^1 = -1$ $\quad 3 = 3$ $\quad k = 1$

$m(4) = 0$ $\quad 4 = 2^2$

$m(5) = (-1)^1 = -1$ $\quad 5 = 5$ $\quad k = 1$

$m(6) = (-1)^2 = 1$ $\quad 6 = 2.3$ $\quad k = 2$

M

$m(12) = 0$ $\quad 12 = 2^2.3$

M

**Teorem 2.3.6 (Bourbaki, 1972):** $X$ bir küme ve $|X| = r$, $X$ in elemanlarının sayısını göstersin. O zaman $H$, $X$ üzerinde bir Hall küme ise $H_n$ içindeki elemanların sayısı $n \geq 1$ için

$$|H_n| = \frac{1}{n} \sum_{\substack{d \\ d|n}} m(d).r^{\frac{n}{d}}$$

dir.

**Örnek 2.3.7:** Kabul edelim ki $|X| = 3$ olsun. O zaman;

$|H_1| = 3$ $\quad n = 1$ $\quad |H_1| = \frac{1}{1} m(1).3^{\frac{1}{1}} = 3$

$$|H_2|=3 \qquad n=2 \qquad |H_2|=\frac{1}{2}\left( m(1).3^{\frac{2}{1}}+m(2).3^{\frac{2}{2}} \right)=3$$

$$|H_3|=8 \qquad n=3 \qquad |H_3|=\frac{1}{3}\left( m(1).3^{\frac{3}{1}}+m(3).3^{\frac{3}{3}} \right)=8$$

$$|H_4|=18 \qquad n=4 \qquad |H_4|=\frac{1}{4}\left( m(1).3^{\frac{4}{1}}+m(2).3^{\frac{4}{2}}+m(4).3^{\frac{4}{4}} \right)=18$$

$$|H_5|=48 \qquad n=5 \qquad |H_5|=\frac{1}{5}\left( m(1).3^{\frac{5}{1}}+m(5).3^{\frac{5}{5}} \right)=48$$

ve genel olarak $H_n$ in eleman sayısı

$$\frac{1}{n}\sum_{\substack{d \\ d|n}} m(d).3^{\frac{n}{d}}$$

şeklindedir. Ayrıntılı bilgi için Warm (1964) , Shirsov (1962) ve Bourbaki (1972) ye bakabilirsiniz.

**Not:** Tek üreteçli serbest Lie cebirleri için bir istisnai durum vardır. $(xx)=0$ olmasından ve bir vektör uzayı olmasından dolayı bir boyutlu olduğu görülür.

## 2.4. Bir Serbest Lie Cebirinin Alt Merkezi Serisinin Terimleri İçin Serbest Üreteçler

$k$ üzerindeki bir serbest Lie cebirinin $m \geq 2$ için $F_{(m)}$ alt merkezi serisinin terimleri $F$ nin bir alt cebiri gibi sonlu üretilmiş değildir. Bir eleman tarafından üretilmiş serbest Lie cebirleri serbest abelyen olduğundan bu durum istisnadır. Alt merkezi seriler için serbest üreteç kümesini bulma problemi ilk kez Grünberg (1957) tarafından grup teorisinde ele alınmıştır. Bir sonuç olarak Witt (1956) tarafından serbest Lie halkaları için uygulanmıştır. $F$, bir serbest Lie cebiri olsun, $F_{(m)}$ için serbest üreteç kümesi Smel'kin (1963) tarafından verilmiştir. Gruplar ve serbest Lie halkaları için detaylı bilgiyi Warm (1972) da bulabilirsiniz.

$F$, $k$ cismi üzerinde serbest bir Lie cebiri ve $X$ serbest üreteç kümesi olsun. $H$, $X$ üzerinde yapılanmış $F$ için bir Hall kümesi olsun. $H_{(n)}$, uzunluğu $n$ olan $H$

deki elemanların kümesini göstersin. $F_{(m)}$ ile $F$ nin $m$-inci alt merkezi serisinin terimini göstereceğiz.

**Teorem 2.4.1 (Smel'kin, 1963):** $C_m$ yi şöyle tanımlayalım.

$$C_m = \{x = a_1 a_2 \mid a_1, a_2 \in H,\ l(x) \geq m,\ x \in H,\ l(a_1) < m\}$$

$C_m$, $F_{(m)}$ için bir serbest üreteç kümesidir.

$H$ nin tanımı kullanılarak $C_m$ nin açıklaması aşağıda daha iyi yapılmıştır.

$$\begin{aligned} C_m = \{\ x = x_1(x_2(\ldots(x_{r-1}x_r))\ldots) \mid l(x) \geq m,\quad & x_i \in (H_1 \cup \ldots \cup H_{m-1}), \\ & x_1 \geq x_2 \geq \ldots \geq x_{r-1} < x_r,\quad r \geq 2, \\ & \text{Eğer } x_r = b_1 b_2 \text{ ise } x_{r-1} \geq b_1\ \} \end{aligned}$$

$C_m$ in elemanlarının ürünlerini oluşturarak $F_{(m)}$ Lie cebiri için $C_m$ serbest üreteçleri üzerinde $H^{C_m}$ bir Hall bazı yapısını kurarız. $h$, $C_m$ formundaki elemanların bir kelime ise $h$ nin *X-uzunluğu*nu $X - l(h)$ ve $C_m$-uzunluğunu da $C_m - l(h)$ ile göstereceğiz.

$C_m$, $H$ nin bir alt kümesi olduğundan $C_m$ deki sıra $H$ deki ile aynı sırada alınabilir.

$$H_1^{C_m} = C_m$$

$$H_2^{C_m} = \{a_1 a_2 \mid a_1, a_2 \in C_m,\ a_1 < a_2\}$$

Şimdi $H_2^{C_m}$ aşağıdaki gibi sıradadır. $h, g \in H_2^{C_m}$, $h = h_1 h_2$ ve $g = g_1 g_2$, $h_1, h_2, g_1, g_2 \in C_m$. Eğer $X - l(h) < X - l(g)$ ise $h < g$ yazılır. Kabul edelim ki $h$ ve $g$ aynı uzunlukta olsun. O zaman $h < g \Leftrightarrow h_2 < g_2$ veya $h_2 = g_2$ ise $h_1 < g_1$ dir.

Kabul edelim ki $H_1^{C_m}, \ldots, H_{n-1}^{C_m}$ tanımlı ve sıralı olsun. Şunu yazarız;

$$\begin{aligned} H_n^{C_m} = \{\ x = a_1(a_2 a_3) \mid C_m - l(x) = n,\ \ & a_1 < a_2 a_3,\ a_1 \geq a_2 < a_3, \\ & a_1, a_2, a_3, a_2 a_3 \in (H_1^{C_m} \cup \ldots \cup H_{n-1}^{C_m})\ \} \end{aligned}$$

eşitsizliğin işareti (yönü) $(H_1^{C_m} \cup \ldots \cup H_{n-1}^{C_m})$ in içindeki sırasına göredir. Şimdi

$$H_n^{C_m},\ H_2^{C_m}$$

için tanımlanmış benzer bir sırada verelim. Artık bu sırayı

$$H^{C_m} = \bigcup_{j=1}^{\infty} H_j{}^{C_m}$$

gibi tanımlar ve $H^{C_m}$ i genişletmiş olabiliriz.

**Not:** $H^{C_m}$ in sırası $H$ deki ile bir arada olmasına gerek olmayabilir. Şu aşikârdır ki $H_2{}^{C_m}$ in içindeki elemanlar öyle elemanlardır ki $H_1{}^{C_m} = C_m$ in bir elemanından $X-l$ den küçük.

Eğer $L$ bir nilpotent Lie cebir ise $L$ nin her alt cebiri aynı zamanda nilpotent olacak ve asıl cebirin olası bir alt merkezi nilpotenttir.

Bir önceki bölümde şunu ifade etmiştik; verilen bir $X$ kümesi, $X$ üzerindeki $H$ Hall kümesi, $X$ üzerindeki $F$ serbest Lie cebiri için toplamsal bir bazdır. $\left(L_{(m)}L_{(n)}\right) \subseteq L_{(m+n)}$ den dolayı uzunluğu $\geq n$ olan $F$ nin her elemanı $F_{(n)}$ in içindedir. Bu nedenle $H_n \subseteq F_{(n)}$ dir.

**Tanım 2.4.2:** $L$, bir $X$ üzerinde serbest bir Lie cebiri olsun. Eğer bir $n$ pozitif tamsayısı için

$$L = F \big/ F_{(n)}$$

olacak şekilde bir $X$ tarafından üretilen $F$ serbest Lie cebiri varsa $L$ ye $n$-inci dereceden serbest nilpotent Lie cebiri denir.

**Teorem 2.4.3:** $F$, bir $X$ üzerinde serbest bir Lie cebiri olsun. $H$, $F$ için bir Hall bazı olsun. O zaman $F_{(n)} \big/ F_{(n+1)}$ serbest Lie cebirinin bazı da $H$ dir ve uzunluğu $n$ olan elemanlardan oluşur. Yani $H_n$ dir.

**Teorem 2.4.4:** $L = F \big/ F_{(n)}$ bir serbest nilpotent ve $F$ için bir Hall bazı $H$ olsun. uzunluğu $\leq n$ olan $H$ nin elemanları ise $\left(H_1 \cup H_2 \cup ... \cup H_{n-1}\right)$, $L$ için bir toplamsal baz formundadır.

## 2.5. Policentral Seriler ve Polinilpotent Lie Cebirleri

**Tanım 2.5.1:** $L$, serbest bir Lie cebiri olsun ve $L$ nin alt merkezi serisi $n \in N$ için $\{L_{(n)}\}$ olsun. Kabul edelim ki $\{n_1,...,n_i,...\}$, $n_i \geq 1$ olmak üzere tamsayıların herhangi bir dizisi olsun. Policentral seriyi şöyle tanımlarız.

$$L \supseteq L_{(n_1)} \supseteq ... \supseteq L_{(n_1),...,(n_i)} \supseteq L_{(n_1),...,(n_{i+1})}$$

aşağıdaki gibi $L$ nin $n_1$-inci alt merkezi serisinin elemanıdır. $L_{(n_1),...,(n_{i+1})} = \left(L_{(n_1),...,(n_i)}\right)_{(n_{i+1})}$ de $L_{(n_1),...,(n_i)}$ nin $n_{i+1}$-inci alt merkezi serinin terimidir. Eğer

$$L_{(n_1),...,(n_k)} = \{0\} \qquad (2.5.1)$$

ise $\{n_1,...,n_k\}$ dizisine göre $L$ ye polinilpotent Lie cebiri denir. $n_i$ tamsayılarının hiçbiri, $i = 1,2,...,k$ daha küçük bir tamsayı ile yer değiştirebilir ve (2.5.1) formunun bir denklemi hala doğru olur.

$L$ nin policentral serisinin terimleri $L$ de bir idealler zinciri şeklindedir.

$$L_{(n_1),...,(n_{i+1})} < L_{(n_1),...,(n_i)} < ... < L_{(n_1)} < L$$

Buradan, policentral bölüm cebirleri yapısı kurulabilir.

$$L_{(n_1),...,(n_i)} \Big/ L_{(n_1),...,(n_{i+1})}$$

Eğer bazı tamsayılar dizisine göre, $L$ bir polinilpotent Lie cebiri ise $L$ nin herhangi bir alt cebiri aynı zamanda polinilpotent olacaktır. Fakat bu alt cebir aynı diziye göre polinilpotent olmak zorunda değildir

**Tanım 2.5.2:** *L* bir *X* kümesi tarafından üretilen bir serbest Lie cebiri olsun.

**a)** Eğer $L \cong F \Big/ F_n$ olacak şekilde X üzerinde serbest bir $F$ Lie cebiri varsa $L$ ye $n$ – yinci dereceden serbest nilpotent Lie cebiri denir. Eğer $L \cong F \Big/ F_2$ ise $L$ ye serbest abelyen Lie cebiri denir.

**b)** Eğer $L \cong F/F^m$ olacak şekilde X üzerinde serbest bir $F$ Lie cebiri varsa $L$ ye $m$ – yinci dereceden serbest çözülebilir Lie cebiri denir.

**c)** Eğer $L = F/F_{n_1,\ldots,n_k}$ olacak şekilde X üzerinde serbest bir $F$ Lie cebiri varsa $L$ ye $\{n_1, \ldots, n_k\}$ dizisine göre serbest polinilpotent Lie cebiri denir. Eğer $L = F/F_{2,2}$ olacak şekilde X üzerinde serbest bir $F$ Lie cebiri varsa $L$ ye serbest metabelyen Lie cebiri denir.

Dikkat edilecek olursa $F_{\underbrace{2,2,\ldots,2}_{k}} = F^{2^k}$ olup $\{2,2,\ldots,2\}$ dizisine göre serbest polinilpotent bir Lie cebiri, $k$ – yıncı dereceden serbest çözülebilir Lie cebiridir.

## 2.6. Bir Serbest Lie Cebirinin Policentral Serilerinin Terimleri için Bazlar ve Serbest Üreteçler

Bir $\{n_1,\ldots,n_k\}$ pozitif tamsayı dizisine göre $F_{(n_i),\ldots,(n_k)}$ teriminin serbest üreteç kümesi ve Hall bazını oluşturalım.

Daha önce $F_{(n_1)}$ için $C_{n_1}$ serbest üreteç kümesini şöyle tanımlamıştık;

$$C_{n_1} = \{x = a_1 a_2 \mid x \in H \text{ ve } l(x) \geq n_1;\ l(a_1) < n_1\}$$

$F_{(n_1)}$ için $H^{C_{n_1}}$ Hall bazıdır. Bunu daha önce vermiştik. Şimdi $F_{(n_1),(n_2)}$ için aynı işlemleri $C_{n_1,n_2}$ serbest üreteç kümesi için yapalım.

$$C_{n_1,n_2} = \{x = a_1 a_2 \mid x \in H^{C_{n_1}};\ C_{n_1} - l(x) \geq n_2; C_{n_1} - l(a_1) < n_2\}$$

Bu küme üzerindeki sıralama verilen elemanların $C_{n_1}$ - uzunluğuna ve *X*- uzunluğuna bakılarak sıralanır. Bu şekilde tanımlı $C_{n_1,n_2}$ kümesi $F_{n_1,n_2}$ nin serbest üreteç kümesidir.

Şimdi $F_{(n_1),(n_2)}$ için $H^{C_{n_1,n_2}}$ Hall bazını oluşturalım.

$$H_1^{C_{n_1,n_2}} = C_{n_1,n_2}$$

$$H_2^{C_{n_1,n_2}} = \{a_1 a_2 \mid a_1, a_2 \in C_{n_1,n_2};\ a_1 < a_2\}$$

Bu kümeyi daha önceki gibi sıralayalım. Şimdi böyle $H_1^{C_{n_1,n_2}},...,H_{m-1}^{C_{n_1,n_2}}$ de tanımlanmış ve sıralanmış olsun. O zaman

$$H_m^{C_{n_1,n_2}} = \{\ x = a_1(a_2a_3) \mid C_{n_1,n_2} - l(x) = m,\ a_1 < a_2.a_3,\ a_1 \geq a_2 < a_3$$
$$a_1, a_2, a_3, a_2a_3 \in (H_1^{C_{n_1,n_2}} \cup ... \cup H_{m-1}^{C_{n_1,n_2}})\ \}$$

$H^{C_{n_1,n_2}} = \bigcup_{m=1}^{\infty} H_m^{C_{n_1,n_2}}$ kümesi $F_{(n_1),(n_2)}$ nin Hall bazıdır.

$F_{(n_1)} \supseteq F_{(n_1),(n_2)} \supseteq F_{(n_1),(n_2),...,(n_{k-1})} \supseteq F_{(n_1),(n_2),...,(n_k)}$ kapsamını göz önünde bulundurursak bu cebirler için serbest üreteç kümesi ve Hall bazları tanımlanmış olsun. Şimdi, $F_{(n_1),(n_2),...,(n_{k-1})}$ in üreteç kümesi $C_{n_1,...,n_{k-1}}$, ve bazı $H_1^{C_{n_1,...,n_{k-1}}}$ dir.

$$C_{n_1,...,n_k} = \{x = a_1a_2 \mid x \in H_1^{C_{n_1,...,n_{k-1}}},\ C_{n_1,...,n_{k-1}} - l(x) \geq n_k,\ C_{n_1,...,n_{k-1}} - l(x) < n_k\ \}$$

Şeklindedir. Bu küme aşağıdaki sıraya göre uzunluklar göz önünde bulundurularak sıralanır.

$$C_{n_1,...,n_{k-1}} - lengths$$
$$C_{n_1,...,n_{k-2}} - lengths$$
$$\vdots$$
$$C_{n_1} - lengths$$
$$x - lengths$$

Bu küme $F_{(n_1),(n_2),...,(n_k)}$ nın serbest üreteç kümesidir. Kabul edelim ki $H_1^{C_{n_1,...,n_k}},...,H_{m-1}^{C_{n_1,...,n_k}}$ tanımlı ve sıralı olsun. O zaman

$$H_m^{C_{n_1,...,n_k}} = x = \{\ a_1(a_2a_3) \mid C_{n_1,...,n_k} - l(x) \geq m,\ a_1 < a_2.a_3,\ a_1 \geq a_2 < a_3,$$
$$a_1, a_2, a_3, a_2a_3 \in (H_1^{C_{n_1,...,n_k}} \cup ... \cup H_{m-1}^{C_{n_1,...,n_k}})\ \}$$

Olarak tanımlarsak $H = \bigcup_{m=1}^{\infty} H_m^{C_{n_1},...,n_k}$ kümesi $F_{(n_1),(n_2),...,(n_k)}$ nın bazıdır.

## 2.7. Serbest Polinilpotent Lie Cebirleri için Bazlar

Bu kısımda $L = F/F_{(n_1),\ldots,(n_k)}$ serbest polinilpotent Lie cebirinin bir bazını kurmak istiyoruz. Bunun için aşağıdaki $B_n$ kümesini tanımlayalım.

$$B_1 = H_1 \cup H_2 \cup \ldots \cup H_{n_1-1}$$
$$B_2 = H_1^{C_{n_1}} \cup H_2^{C_{n_2}} \cup \ldots \cup H_{n_2-1}^{C_{n_1}}$$
$$\vdots$$
$$B_k = H_1^{C_{n_1,\ldots n_{k-1}}} \cup \ldots \cup H_{n_k-1}^{C_{n_1,\ldots n_{k-1}}}$$

buradan

$$B = \bigcup_{i=1}^{k} B_i$$

Olsun. Bu şekilde tanımlanan $B$ kümesi $L = F/F_{(n_1),\ldots,(n_k)}$ serbest polinilpotent Lie cebirinin bir bazıdır [ Smel'kin ]. $B_i$ kümeleri parçalarına ayırmalıdır. Şöyle ki $B_i \cap B_j = \varnothing \;\; i \neq j \;\; i, j \in \{1, 2, \ldots, k\}$.

$$B_1 = X$$
$$B_2 = C_2$$
$$B_3 = C_{2,2}$$
$$\vdots$$
$$B_k = C_{\underbrace{2,2,\ldots,2}_{(k-1)-\text{tane}}}$$

şeklindedir.

$B = \bigcup_{i=1}^{k} B_i$, kümesi $L = F/F^{(k)}$ nın bir bazıdır.

Özel olarak $n_1 = n_2 = \ldots = n_k = 2$ alınırsa $F^{n_1 n_2 \ldots n_k} = F^{2,2,\ldots,2} = F^{(k)}$ olup $F/F^{2,2,\ldots,2}$ Lie cebirine serbest metabelyen Lie cebiri denir.

## 2.8. Dairesel kelimeler ve modülo *B* bazı

**Tanım 2.8.1:** Değişkenleri $x_1, x_2, \ldots, x_n$ olan bir polinomun değişkenlerinin herhangi bir permütasyonu alındığında değişmeyen polinomlara simetrik polinomlar denir. Bir

başka deyişle simetrik polinom aşağıdaki formda olan polinomlardır. $1, 2, \mathbf{L}, n$ indislerinin herhangi bir permütasyonunu $p(i)$ olsun $y_i = x_{p(i)}$ olmak üzere

$$P(y_1, y_2, \mathbf{L}, y_n) = P(x_1, x_2, \mathbf{L}, x_n)$$

dir. Örneğin $\sqrt{x_1^2 + x_2^2 + \mathbf{L} + x_n^2}$ veya $x_1^2 + x_2^2 + \mathbf{L} + x_n^2 - 2x_1x_2\,\mathbf{L}\,x_n$ polinomları simetrik polinomlardır.

**Tanım 2.8.2:** $A$ bir alfabe ve $K$ bir halka olsun. $w = a_1a_2\,\mathbf{L}\,a_n$ $(a_i \in A)$ olmak üzere $K$ üzerinde tanımlı bir polinom

$$P = \sum_{w \in A^*} (P, w)w$$

şeklindedir. Tüm polinomlar kümesini $K\langle A\rangle$ ile gösterelim. $K\langle A\rangle$, $A$ tarafından üretilen serbest asosyatif $K$-cebirdir. $K$ üzerinde $A$ nın formel serisi (veya serisi) aşağıdaki şekilde sonlu formel lineer kombinasyonlardır.

$$S = \sum_{w \in A^*} (S, w)w$$

Tüm serilerin kümesi $K = A$ **?** ile gösterilir.

**Tanım 2.8.3:** $a$ ve $b$ iki harf olmak üzere $log(e^a e^b)$ serisi bir Lie serisidir. Bu seriye Hausdorff serisi denir. Bu formüle Campbell-Baker-Hausdorff (CBH) formülü denir.

**Tanım 2.8.4.:** $F$ , $X$ üzerinde serbest Lie cebiri olsun. $X$ kümesine lineer sıralama verilmiş olalım. $X$ in elemanlarını uzunluğu 1 olan **regüler kelimeler** olarak isimlendirelim. Uzunluğu $n$'den küçük olan kelimeleri sıralamış olduğumuzukabul edelim. O zaman $X$-uzunluğu $n$ olan bir $w=uv$ kelimesine, aşağıdaki koşulları sağlıyorsa bir **regüler kelime** denir.

*i)* $u$ ve $v$ regüler kelimelerdir. ,

*ii)* $u>v$,

*iii)* $u = u_1u_2$ ise $u_2 \le v$ dir.

$X$ kümesindeki elemanların Lie çarpımlarından oluşan bir kelimeye bir **monomial** denir. Monomiallerin herhangi bir lineer kombinasyonuna da bir Lie **Polinomu** diyeceğiz. O halde bir regüler kelime bir monomialdir.

**Tanım 2.8.5.:** $L$ bir Lie cebiri ve $M \neq \emptyset$, $L$ nin bir alt kümesi olsun. Eğer her $f \in M$ için $f$ nin leading terimi olan $ld(f)$ elemanı , $\{ld(g) | g \in M - \{f\}\}$ kümesinin doğurduğu alt uzaya ait değilse $M$ ye **indirgenmiş küme** diyeceğiz. Boş kümenin indirgenmiş olduğunu varsayacağız.

**Teorem 2.8.7**. (Kukin, 1972) : Bir serbest Lie cebirinin indirgenmiş bir $M$ alt kümesi bağımsızdır.

**Tanım 2.8.8**.: $L$ bir Lie cebiri ve $B$, $L$ nin bir alt cebiri olsun. $L$ de modülo$B$ maksimal lineer bağımsız kümeye ***L* nin modülo*B* bazı** denir. $L$ nin modülo$B$ bazının eleman sayısına $B$ nin $L$ deki **koboyutu** diyeceğiz ve bunu $(L:B)$ ile göstereceğiz.

# 3. UNIVERSAL ENVELOPING ALGEBRA VE POİNCARÉ – BİRKHOFF – WİTT TEOREMİ

Evrensel enveloping cebir Lie cebirlerinin temsil teorisinin bir temel yapısıdır. $L$ , bir bazı $\{x_1, x_2, \mathbf{L}, x_n\}$ olan bir Lie cebiri ve $r : L \to gl(V)$ ye bir temsil olsun ($V$ herhangi bir vektör uzayı). $x, y \in L$ için $r(x)r(y)$ çarpımı genelde $r(L)$ ye ait olmaz. Bununla birlikte bir çok yerde bu çarpım fonksiyonların bileşkesi olarak alınır. Yapının temeli birim eleman ve $x \in L$ olmak üzere $r(x)$ ler tarafından üretilen asosyatif cebirdir. Bu cebire L nin enveloping cebiri denir ve $r(L)^*$ ile gösterilir. Bundan başka doğuraylar $r(x_i)r(x_j) = r(x_j)r(x_i) + r([x_i, x_j])$ bağıntılarını sağlarlar. Bu bağıntılar özel bir $r$ temsiline bağlı değildir.

$L$ nin enveloping cebiri $L$ ve $x_1, x_2, \mathbf{L}, x_n$ gibi $n$ tane bilinmeyen tarafından üretilen ve $x_i x_j - x_j x_i - [x_i, x_j]$ bağıntılarını sağlayan asosyatif cebirdir. Böylece her enveloping cebir evrensel enveloping cebirin bir bölüm cebiridir. Bu anlamda evrensel enveloping cebir $L$ nin tüm enveloping cebirlerini kapsar.

Biz bu kısımda $L$ bir Lie cebiri olduğunda $L$ nin evrensel enveloping cebirini vererek bir bazının hangi formda olduğunu inceleyeceğiz. Ayrıca bir serbest asosyatif cebirin her bir alt cebirinin Gröbner bazını inceleyeceğiz.

## 3.1. Serbest asosyatif cebirlerde idealler.

$X = \{x_1, x_2, \mathbf{L}\}$ bir küme olsun. $X^*$ ile $X$ in elemanları üzerinde tanımlı $w = x_{i_1} x_{i_2} \mathbf{L}\, x_{i_k}$ kelimelerinin tamamının kümesini gösterelim. $X^*$ aynı zamanda 1 ile gösterilen boş kelimeyi de içerir. $X^*$ kümesi $X$ üzerinde serbest monoiddir. $X^*$ üzerindeki ikili işlemi aşağıdaki gibi tanımlarız;

$$\cdot : X^* \times X^* \to X^* \qquad u \cdot v = uv \text{ (yan yana yazma)} \qquad (3.1.1)$$

Eğer $w = x_{i_1} x_{i_2} \mathbf{L}\, x_{i_k} \in X^*$ ise $w$ nunderecesi $\deg(w) = k$ dır.

Eğer $u, v \in X^*$ olmak üzere $v = w_1 u w_2$ olacak şekilde $w_1, w_2 \in X^*$ elemanları var ise $u$ ya $v$ nin bir çarpanı (faktörü) deriz. Örneğin $u$ , $v$ nin alt kelimesi ise $v$ nin

bir çarpanıdır. Eğer $v = uw_2$ ise $u$ , $v$ nin sol çarpanı, $v = w_1u$ ise $u$ , $v$ nin sağ çarpanı denir.

Bu bölüm boyunca $X^*$, çarpımsal ve azalan zincir kuralını sağlayan < şeklinde tam sıralı olarak alacağız. Yani $u < v$ ise $w \in X^*$ için $wu < wv$ ve $uw < vw$ dir. Eğer $w_1 \geq w_2 \geq \mathbf{L}$ kelimelerin azalan bir zinciri verildiğinde $w_k = w_{k+1} = \mathbf{L}$ olacak şekilde bir $k > 0$ vardır. Örnek olarak deglex sıralamasını $<_{dlex}$ olarak alalım. Sıralama tanımı şöyledir;

$\deg(u) < \deg(v)$ ise $u <_{dlex} v$ Eğer $\deg(u) = \deg(v)$ iken $i<j$ ve $w, u', v' \in X^*$ olmak üzere $u = wx_iu'$, $v = wx_jv'$

Şimdi $F$ bir cisim ve $F(X)$ , $X^*$ tarafından gerilen vektör uzayı olsun. 3.1.1 de verilen "$\cdot$" işlemini $F(X)$ e bilineer olarak genişletelim. Bu durumda $F(X)$, birim elemanlı bir asosyatif cebir olur. Bu cebire $F$ üzerinde birim elemanı 1 olan bir serbest asosyatif cebir denir. $X^*$ ın elemanlarına monomialler denir. $f \in F(X)$ nin boş olmayan tüm monomiallerinin kümesine $f$ nin dayanağı (support) denir. < sıralaması $f \in F(X)$ elemanının leading monomiali fikrini belirler. $f \in F(X)$ leading monomiali $f$ nin dayanak elemanlarının en büyük dereceli (< sıralamasına göre) monomialidir ve LM($f$) ile gösterilir.

$S \subset F(X)$ için $LM(S) = \{LM(f) \mid f \in S\}$

olarak tanımlanır.

$I$ , $F\langle X\rangle$ in bir ideali olsun. $F\langle X\rangle / I$ yı hesaplayalım. $F\langle X\rangle / I$ nın bir bazını bulmak istiyoruz. Yani $F\langle X\rangle$ in bütün modülo $I$ temsilcilerinin bir kümesini bulmak istiyoruz. Bunun anlamı $F\langle X\rangle$ deki $I$ ya göre tümleyen olan ve $B$ tarafından gerilen $B \subset F\langle X\rangle$ kümesini bulmaktır. Bundan başka baz elemanlarının çarpımını modülo $I$ baz elemanlarının lineer kombinasyonu olarak göstermemiz gerekmektedir. Bu problem $F\langle X\rangle$ in *modülo* $I$ normal kelimelerinin kümesi ile çözülür. Yani bu küme

$$N(I) = \{u \in X^* \mid u \notin LM(I)\}.$$

dir. $C(I)$ ile $N(I)$ tarafından gerilen vektör uzayını gösterelim.

**Önerme 3.1.1:** $I \subset F\langle X\rangle$ bir ideal ise $F\langle X\rangle = C(I) \oplus I$ dır.

**İspat:**

$$N(I) = \{u \in X^* \mid u \notin LM(I)\}$$

$$C(I) = \text{Span}\, N(I)$$

Olsun.

$$C(I) \cap I = 0$$

olduğu açıktır.

$f \in F\langle X\rangle$ olsun $f = v + p$ olacak şekilde $v \in C(I)$ ve $p \in I$ olduğunu göstereceğiz. Tümevarımla;

$$LM(h) < LM(f)$$

olacak şekildeki her $h \in F\langle X\rangle$ için doğru olsun. (Azalan zincir kuralına göre $LM(f)$ den küçük monomiallerin sayısı sonludur). $LM(h) < LM(f)$ ise $l \in F$ için

$$f = l\,LM(f) + h$$

yazılır. Tümevarımdan

$$h = v + c,\ v \in C(I) \text{ ve } p \in I$$

dir.

Eğer $LM(f) \in N(I)$ ise ispat biter. Bunun yanlış olduğu yani $LM(f) \notin N(I)$ olduğu durumda bir $g \in I$ için $LM(f) < LM(g)$ olur ve

$$g = mLM(g) + h'$$

ve tümevarımdan $u \in C(I)$ ve $q \in I$ için,

$$f - \frac{l}{m} g\ ,\ u + q$$

ya eşit olur. Sonunda

$$f = u + q + \frac{l}{m} g$$

olur ve ispat biter.

Şimdi $f \in F\langle X\rangle$ ise Önerme 3.2.1 e göre $f$ nin $f = v + p$ olacak şekilde ($v \in C(I)$, $p \in I$) bir tek yazılışa sahiptir. $v \in C(I)$ elemanına $f$ nin $I$ modülüne göre normal formu denir. $Nf_I(f)$ veya $Nf(f)$ ile gösterilir.

$$f = v + p$$
$$\downarrow$$
$$Nf(f) = v$$

$u, v \in C(I)$ ve $u * v = Nf(uv)$ olsun. O halde * işlemi ile $C(I)$ bir cebirdir. Bu cebir $F\langle X\rangle / I$ ya izomorfiktir.

Eğer biz, bir normal formu hesaplayan bir algoritma geliştirebilirsek o zaman $F\langle X\rangle / I$ cebirini kolayca hesaplayabiliriz. $G \subseteq F\langle X\rangle$, $F\langle X\rangle$ in bir idealinin doğuray kümesi olsun. Herhangi bir $f \in F\langle X\rangle$ elemanı olmak üzere, her $g \in G$ için LM($g$), $f$ nin ifadesindeki hiçbir monomialin bir faktörü değilse $f$ elemanına $G$ ye göre normal formdadır denir.

Aşağıdaki algoritma Modülo$G$ normal form hesaplama algoritmasıdır.

**Normal Form Algoritması**

**İnput:** $I \subset F\langle X\rangle$ in bir üreteç kümesi $G$ ve $f \in F\langle X\rangle$ olsun.

**Output:** $f \in F\langle X\rangle$ elemanı modülo $G$ normal formdadır öyle ki $f = f \bmod I$

**Adım 1:** $f = 0$ ve $h = f$ olsun.

**Adım 2:** Eğer $h = 0$ ise $f$ yi getir. Değil ise $u = LM(h)$ yap ve $l$ yı $u$ nun katsayısı olarak al, $h$ yi belirle.

**Adım 3:** $u$ nun bir çarpanı $LM(g)$ olacak şekilde $g \in G$ olsun. Eğer böyle bir $g$ yoksa

$h = h - lu$ ve $f = f + lu$

yap ve Adım 2 ye dön.

**Adım 4:** $g$ deki $LM(g)$ nin katsayısı $m$ ve $v, w \in X^*$ olsun. Öyle ki $vLM(g)w = u$.

$$h = h - \frac{l}{m} vgw$$

alıp Adım 2 ye dön.

**Açıklama:** Her adımda *LM(h)* nin azaldığı görülür. Böylece < azalan zincir koşulunu sağladığından bu işlemler belli bir adım sonunda sona erer. Sadece herhangi bir *LM(g)* yi çarpan olarak içermeyen monomialler $f$ ye eklenir. Açık olarak, sonlu adım sonunda elde edilen $f$ G ye göre normal formdadır. Bundan başka herhengi bir adımda elde edilen $f = h + f \bmod I$ olup $f = f \bmod I$ dır.

Normal form algoritmasını aşağıdaki biçimde tekrar formalize edebiliriz.

$h \in F\langle X \rangle$ ve $w \in X^*$ da $h$ nin supportunda bulunan ve bir $g \in G$ için $LM(g)$ yi bir alt kelime olarak içersin. $u, v \in X^*$ olmak üzere

$$u\,LM(g)\,v = w$$

Olsun. Bu durumda $h$ elemanı $h'$ ye indirgenir denir. Öyle ki

$$h' = h - \frac{l}{m} ugv \qquad \text{dir.}$$

Burada $l$ ve $m$ sırasıyla $h$ ve $g$ nin leading monomiallerinin katsayılarıdır.

Daha genel olarak, $h_1, \mathbf{L}, h_k$ lar $h_i$ ler $h_{i+1}$ e indirgenmiş olsun $1 \le i \le k-1$. Bu durumda $h_1$, $h_k$ ya indirgenmiştir. Ve $h_1 \to h_k$ ile gösterilir. Aslında normal form algoritması indirgeme adımları serisidir. Eğer başka mümkün indirgeme yoksa istenen sonuç elde edilmiştir.

**Örnek 3.1.2:** $X = \{x, y\}$, $G = \left\{ \underset{g_1}{xy - x^2}, \underset{g_2}{x^3 - yx}, \underset{g_3}{y^3} \right\}$ ve $<_{dlex}$ olsun. $x <_{dlex} y$ olsun.

$$LM(G) = \{LM(g_1), LM(g_2), LM(g_3)\} = \{xy, x^3, y^3\} \text{ dir.}$$

$f = x^2y^2 \in F\langle X \rangle$ elemanının normal formunu bulmak istiyoruz.

$f = x^2y^2$ ise $f = x(xy)y$ şeklinde yazılabilir. Şunu görürüz $f \to x^3y$. Şöyle devam eder ve

$$x^3y \to yxy \to yx^2$$

olur. Bu ise en sonunda $G$ ye göre normal formdur. Sonuç olarak;

$$-(xy-x^2)y^2+x(y^3)=x^2y$$

$$x^3y=x^2(xy)\to x^4=x(x^3)\to xyx\to x^3\to yx$$

çıktı $yx$ .dir.

Görüldüğü gibi üreteç kümesi $G$ olan $I$ idealinin bir $f$ elemanını modülo $I$ normal formu tek bir tane değildir. Acaba Normalformun tek bir tane olduğu durum var mıdır? Bunun için Gröbner bazına ihtiyaç duyacağız.

**Tanım 3.1.3 (Gröbner Bazı):** $I$ , $F\langle X\rangle$ in bir ideali olsun. $G\subset I$ için eğer her $f\in I$ için $LM(g)$, $LM(f)$ nin bir alt kelimesi olacak şekilde bir $g\in G$ varsa $G$ ye $I$ nın bir **Gröbner bazı** denir.

**Örnek 3.1.4** : $F=F\langle X\rangle=\langle x,y,z\rangle$ olsun. $G=\{x,y\}$ kümesi $I=\langle x,y\rangle$ idealinin Gröbner bazıdır.

**Teorem 3.1.5:** $G\subset F\langle X\rangle$, $I$ idealinin bir Gröbner bazı olsun. $\forall g\in G$ için $LM(g)$, $w$ nun bir çarpanı olmayacak şekildeki her $w\in X^*$ elemanlarının kümesi $N(I)$ dır. Dahası $NormalForm(G,f)$, her $f\in F\langle X\rangle$ için $Nf_I(f)$ yı verir.

**Tanım 3.1.6:** $G\subset F\langle X\rangle$ olsun. $G$ ye self-reduced denir. Eğer; birinci olarak $g\neq h\in G$ için $LM(g)$, $LM(h)$ nin bir çarpanı değildir. İkinci olarak her $g\in G$ için $LM(g)$ nin katsayısı 1 dir.

$I\subset F\langle X\rangle$ bir ideal ve $G=\{g_1,g_2,...\}$ tarafından üretilsin. $w\in X^*$ için

$$I_{<w}=\left\{\sum_i l_iu_ig_{k_i}v_i \mid u_i,v_i\in X^* \text{ öyle ki } u_iLM(g_{k_i})v_i<w\right\},$$

örneğin; $u,v\in X^*$ için $ugv$ formundaki tüm elemanların gerdiği vektör uzayı olsun ve $g\in G$ öyle ki $LM(ugv)<w$ dur.

**Teorem 3.1.7:** $I$ , $F\langle X\rangle$ in self-reduced $G$ kümesi tarafından üretilen ideali olsun. $G$ , $I$ nın bir Gröbner bazıdır ancak ve ancak her $g_1, g_2 \in G$ ve $u, v \in X^*$ için

$$LM(g_1)u = vLM(g_2)$$

buradan $g_1u - vg_2 \in I_{<t}$ ($t = LM(g_2)u$ için) dir.

**Tanım 3.1.8:** $g_1, g_2 \in F\langle X\rangle$ ve $w_i = LM(g_i)$ $i = 1,2,...$ için $g_1$ ve $g_2$ içindeki $w_1$ ve $w_2$ nin katsayılarının 1 olduğunu kabul edelim. $w_1$, $w_2$ nin bir faktörü değil ve $w_2$ de $w_1$ in bir katı (faktörü) olmasın. O zaman $u, v \in X^*$ için $w_1u = vw_2$ dir ve $u$ , $w_2$ için bir öz sağ çarpan ve $v$, $w_1$ için bir öz sol çarpandır. Öyleyse $g_1u - vg_2$, $g_1$ ve $g_2$ nin bir kompozisyonu (bileşeni) dir denir.

**Sonuç 3.1.9:** $I$ , $F\langle X\rangle$ in self-reduced bir $G$ kümesi tarafından üretilen ideali olsun. $G$ , $I$ nın bir Gröbner bazıdır ancak ve ancak $G$ nin $f$ elemanlarının her kompozisyonu sıfırdır. Yani NormalForm($G$,$f$)=0 dır.

### 3.2. Universal Enveloping Algebra

$L$, bir $F$ cismi üzerinde bazı $B = \{x_1, x_2, ...x_n\}$ olan bir Lie cebiri olsun. ($L$ sonlu boyutlu olmak zorunda değil ) $X$ de $B$ ile aralarında bijeksiyon olacak şekilde bir küme, $f : B \to X$ bir bijeksiyon olsun ve $f : L \to F\langle X\rangle$ bilineer olarak genişletelim. $F\langle X\rangle$, $X^*$ ın elemanları tarafından gerilen vektör uzayı olsun. $X^*$ üzerindeki çarpmayı $F\langle X\rangle$ e genellersek, o zaman $F\langle X\rangle$ asasyatif cebir olur. $x_i \in B$ nin $f$ altındaki görüntüsünü $f(x_i) = \overline{x}_i$ ile gösterelim.

$f(B) = \{f(x_1), f(x_2), \mathbf{L}, f(x_n)\} = X$

$f(B) \times f(B) \to f(B)$ kartezyen çarpımı ile bir serbest monoiddir.

$F = span\, X^*$, bu çarpımı bilineer olarak genişletelim.

$I$ , $F\langle X\rangle$ in aşağıdaki elemanlar tarafından üretilen bir ideali olsun.

$$g_{ij} = \overline{x}_j \overline{x}_i - \overline{x}_i \overline{x}_j - f\left(\left[x_j, x_i\right]\right) \qquad 1 \le i < j \text{ için}$$

O zaman $F\langle X\rangle \big/ I$ bölüm cebirine $L$ nin $U(L)$ Universal Enveloping cebiri denir. $p : F\langle X\rangle \to U(L)$ bir projeksiyon dönüşümü olsun.

$$i : L \xrightarrow{f} F\langle X\rangle \xrightarrow{p} U(L) \cong F\langle X\rangle \big/ I$$

dönüşümü $L$ yi $U(L)$ ye dönüştürür.

$$v \in L \Rightarrow v = l_1 x_1 + l_2 x_2 + \mathbf{L} + l_n x_n$$

$$f(v) = l_1 f(x_1) + l_2 f(x_2) + \mathbf{L} + l_n f(x_n)$$

$$f\left(\left[x_1, x_2\right]\right) = \left[\overline{x}_1, \overline{x}_2\right] = \overline{x}_1 \overline{x}_2 - \overline{x}_2 \overline{x}_1$$

Bir Lie cebirinin evrensel enveloping cebirinin bazını belirleyen aşağıdaki teorem Poincaré – Birkhoff – Witt Teoremi olarak bilinir. Bu teoremin ispatı bir çok kaynakta bulabilirsiniz (Graaf, 2000)

**Teorem 3.2.1 (Poincaré – Birkhoff – Witt):** $U(L)$ nin bir bazı

$$\overline{x}_{i_1}^{m_1} \overline{x}_{i_2}^{m_2} \ldots \overline{x}_{i_k}^{m_k} \quad , \; k \ge 0 \; ve \; i_1 < i_2 < \ldots < i_k$$

formundaki monomiallerden meydana gelir.( $k = 0$ ise monomial 1 alınır.)

**İspat:** $G = \left\{ g_{ij} \middle| 1 \le i < j \right\} I$ idealinin doğuray kümesi olsun. $X^*$ üzerindeki sıralamayı (term order) $<_{dlex}$ alalım ve $i < j$ ise $\overline{x}_i <_{dlex} \overline{x}_j$ olsun. O zaman $LM(g_{ij}) = \overline{x}_j \overline{x}_i$ dir. $G$ nin self-reduced olduğunu kabul edelim ( Yani her bir $g_{ij}$, $g_{kl}$, $LM(g_{ij})$, $LM(g_{kl})$ nin bir parçası değil ve katsayıları 1 olsun ).

$G$ nin $g_{ij}$ ve $g_{kl}$ elemanlarının tüm mümkün olan kompozisyonlarını ele alalım. Burada kısaca görüleceği gibi $\overline{x}_i = \overline{x}_l$ olması durumunda kompozisyon oluşur aksi halde kompozisyon meydana gelmez.

$$g_{ij} = \overline{x}_j\overline{x}_i - \overline{x}_i\overline{x}_j - f\left(\left[x_j, x_i\right]\right)$$
$$g_{kl} = \overline{x}_k\overline{x}_l - \overline{x}_k\overline{x}_l - f\left(\left[x_l, x_k\right]\right)$$

$$\left.\begin{array}{l} LM(g_{ij}) = \overline{x}_j\overline{x}_i \\ LM(g_{kl}) = \overline{x}_l\overline{x}_k \end{array}\right\} LM(g_{ij})u = vLM(g_{kl})$$

olacak şekilde $u, v \in X^*$ bulunabiliyorsa kompozisyon oluşur. O zaman

$\overline{x}_j\overline{x}_i u = v\overline{x}_l\overline{x}_k$, $u$, $\overline{x}_l\overline{x}_k$ nin özsağ kısmı, $v$, $\overline{x}_j\overline{x}_i$ nin özsol parçası

O halde $u = x_k$, $v = x_j$ olmalıdır.

$$\overline{x}_j\overline{x}_i\overline{x}_k = \overline{x}_j\overline{x}_l\overline{x}_k$$

$\overline{x}_i = \overline{x}_l$ olması halinde kompozisyon oluşur. Aksi halde kompozisyon oluşmaz.

$$g_{ij}u - vg_{ki} = \left(\overline{x}_j\overline{x}_i - \overline{x}_i\overline{x}_j - f\left(\left[x_j, x_i\right]\right)\right)u - v\left(\overline{x}_i\overline{x}_k - \overline{x}_k\overline{x}_i - f\left(\left[x_i, x_k\right]\right)\right)$$
$$= \cancel{\overline{x}_j\overline{x}_i\overline{x}_k} - \overline{x}_i\overline{x}_j\overline{x}_k - f\left(\left[x_j, x_i\right]\right)\overline{x}_k - \cancel{\overline{x}_j\overline{x}_i\overline{x}_k} + \overline{x}_j\overline{x}_k\overline{x}_i + \overline{x}_j f\left(\left[x_i, x_k\right]\right)$$

O halde $u = x_k$, $v = x_j$ olmalıdır.

$$g_{ij}u - vg_{ki} = \overline{x}_j\overline{x}_k\overline{x}_i - \overline{x}_i\overline{x}_j\overline{x}_k - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_i, x_k\right]\right) \ldots\ldots\ldots(1)$$

dir.

Şimdi bu kompozisyonun modülo $G$ normal formunu bulalım. (Eğer bu kompozisyonun modülo G normal formunun sıfıra indirgendiğini gösterirsek o zaman problem çözülmüş olur.)

İki adım sonunda $\overline{x}_j\overline{x}_k\overline{x}_i$

$$\overline{x}_k\overline{x}_i\overline{x}_j + \overline{x}_k f\left(\left[x_j, x_i\right]\right) + f\left(\left[x_j, x_i\right]\right)\overline{x}_i \quad k < i < j$$

elemanına indirgenir. Görelim;

$$f = \overline{x}_j\overline{x}_k\overline{x}_i - \overline{x}_i\overline{x}_k\overline{x}_j - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_i, x_k\right]\right)$$
$LM(f) = \overline{x}_j\overline{x}_k\overline{x}_i$, $g_{kj} = \overline{x}_j\overline{x}_k - \overline{x}_k\overline{x}_j - f\left(\left[x_j, x_k\right]\right)$ için
$LM(g_{ij}) = \overline{x}_j\overline{x}_k$

olup $LM(f)$ nin bir alt kelimesidir.

$$h_1 = f - 1.g_{kj}\overline{x}_i$$
$$= f - \left(\overline{x}_j\overline{x}_k - \overline{x}_k\overline{x}_j - f\left(\left[x_j, x_k\right]\right)\right)\overline{x}_i$$
$$= \cancel{\overline{x}_j\overline{x}_k\overline{x}_i} - \overline{x}_i\overline{x}_j\overline{x}_k - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_k, x_i\right]\right) - \cancel{\overline{x}_j\overline{x}_k\overline{x}_i} + \overline{x}_k\overline{x}_j\overline{x}_i - f\left(\left[x_j, x_k\right]\right)\overline{x}_i$$
$$= \overline{x}_k\overline{x}_j\overline{x}_i - \overline{x}_i\overline{x}_j\overline{x}_k - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_k, x_i\right]\right) - f\left(\left[x_j, x_k\right]\right)\overline{x}_i$$

$LM(h_1) = -\overline{x}_i\overline{x}_j\overline{x}, \quad g_{kj} \in G$ için $LM(g_{kj})$, $LM(h)$ nin alt kelimesidir.

$$h_2 = h_1 + \overline{x}_i g_{kj}$$
$$= \overline{x}_k\overline{x}_j\overline{x}_i - \cancel{\overline{x}_i\overline{x}_j\overline{x}_k} - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_k, x_i\right]\right) - f\left(\left[x_j, x_k\right]\right)\overline{x}_i$$
$$+ \overline{x}_i\left(\overline{x}_j\overline{x}_k - \overline{x}_k\overline{x}_j - f\left(\left[x_j, x_k\right]\right)\right)$$
$$\cancel{\overline{x}_i\overline{x}_j\overline{x}_k} - \overline{x}_i\overline{x}_k\overline{x}_j - \overline{x}_i f\left(\left[x_j, x_k\right]\right)$$
$$= \overline{x}_k\overline{x}_j\overline{x}_i - \overline{x}_i\overline{x}_k\overline{x}_j$$

Not: Bu işlem terim terimde olabilir. Bu şekilde alınırsa işlemlerin daha kısa olması bakımından kolaylık sağlar.

$$g_{ij}u - vg_{ki} = \overline{x}_j\overline{x}_k\overline{x}_i - \overline{x}_i\overline{x}_j\overline{x}_k - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_i f\left(\left[x_k, x_i\right]\right)$$

$\overline{x}_j\overline{x}_k\overline{x}_i$ de $\overline{x}_j\overline{x}_k$ alt kelimesi $LM(g_{kj})$ olup

$$h_1 = \overline{x}_j\overline{x}_k\overline{x}_i - (g_{kj})\overline{x}_i$$
$$= \overline{x}_j\overline{x}_k\overline{x}_i - \left(\overline{x}_j\overline{x}_k - \overline{x}_k\overline{x}_j - f([x_j, x_k])\right)\overline{x}_i$$
$$= \cancel{\overline{x}_j\overline{x}_k\overline{x}_i} - \cancel{\overline{x}_j\overline{x}_k\overline{x}_i} + \overline{x}_k\overline{x}_j\overline{x}_i + f([x_j, x_k])\overline{x}_i$$
$$= \overline{x}_k\overline{x}_j\overline{x}_i + f([x_j, x_k])\overline{x}_i$$

$h_1$ deki birinci terim olan $\overline{x}_k\overline{x}_j\overline{x}_i$ yi tekrar indirgersek

$$h_2 = h_1 - \overline{x}_k(g_{ij})$$
$$= \overline{x}_k\overline{x}_j\overline{x}_i - \overline{x}_k\left(\overline{x}_j\overline{x}_k - \overline{x}_i\overline{x}_j - f([x_j, x_i])\right)$$
$$= \cancel{\overline{x}_k\overline{x}_j\overline{x}_i} + f([x_j, x_k])\overline{x}_i - \cancel{\overline{x}_k\overline{x}_j\overline{x}_i} + \overline{x}_k\overline{x}_i\overline{x}_j + \overline{x}_k f([x_j, x_i])$$
$$= \overline{x}_k\overline{x}_i\overline{x}_j + f([x_j, x_k])\overline{x}_i + \overline{x}_k f([x_j, x_i]) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

$LM(h_2) = \overline{x}_k\overline{x}_i\overline{x}_j$ olup $LM(g)$ alt kelime olacak şekilde $g \in G$ bulunmaz.

O halde indirgeme işlemi biter.

$(g_{ij}, g_{kj})$ kompozisyonundaki diğer terime bakarsak, $\overline{x}_i\overline{x}_j\overline{x}_k$ terimini *ModülоG* ye göre indirgeyelim. O zaman

$$\begin{aligned} h_1 &= \overline{x}_j\overline{x}_k\overline{x}_i - \overline{x}_i(g_{kj}) \\ &= \overline{x}_i\overline{x}_j\overline{x}_k - \overline{x}_i\left(\overline{x}_j\overline{x}_k - \overline{x}_k\overline{x}_j - f([x_j, x_k])\right) \\ &= \cancel{\overline{x}_i\overline{x}_j\overline{x}_k} - \cancel{\overline{x}_i\overline{x}_j\overline{x}_k} + \overline{x}_i\overline{x}_j\overline{x}_k + \overline{x}_i f([x_j, x_k]) \\ &= \overline{x}_i\overline{x}_k\overline{x}_j + \overline{x}_i f([x_j, x_k]) \end{aligned}$$

$LM(h_1) = \overline{x}_i\overline{x}_k\overline{x}_j$ olup $\overline{x}_i\overline{x}_k$ alt kelime ve

$$\begin{aligned} h_2 &= -h_1 - g_{ki}\overline{x}_j \quad (LM(h_1) \text{ in katsayısı -1 old.}) \\ &= \overline{x}_i\overline{x}_k\overline{x}_j - \overline{x}_i f([x_j, x_k]) - \left(\overline{x}_i\overline{x}_k - \overline{x}_k\overline{x}_i - f([x_i, x_k])\overline{x}_j\right) \\ &= \cancel{\overline{x}_i\overline{x}_k\overline{x}_j} + \overline{x}_i f([x_j, x_k]) - \cancel{\overline{x}_i\overline{x}_k\overline{x}_j} + \overline{x}_k\overline{x}_i\overline{x}_j + f([x_i, x_k])\overline{x}_j \\ &= \overline{x}_k\overline{x}_i\overline{x}_j + \overline{x}_i f([x_j, x_k]) + f([x_i, x_k])\overline{x}_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3) \end{aligned}$$

(2),(3) de elde ettiğimiz sonuçları (1) de yerine yazarsak ve terimler yerine ModuloG indirgenmiş formları yazarsak,

$$\begin{aligned} g_{ij}u - vg_{ki} &= (2) - (3) - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left([x_i, x_k]\right) \\ &= \overline{x}_k\overline{x}_i\overline{x}_j + f\left(\left[x_j, x_k\right]\right)\overline{x}_i + \overline{x}_k f\left(\left[x_j, x_i\right]\right) \\ &\quad -\left(\overline{x}_k\overline{x}_i\overline{x}_j + \overline{x}_i f\left(\left[x_j, x_k\right]\right) + f\left([x_i, x_k]\right)\overline{x}_j\right) \\ &\quad - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left([x_i, x_k]\right) \\ &= \cancel{\overline{x}_k\overline{x}_i\overline{x}_j} + f\left(\left[x_j, x_k\right]\right)\overline{x}_i + \overline{x}_k f\left(\left[x_j, x_i\right]\right) \\ &\quad - \cancel{\overline{x}_k\overline{x}_i\overline{x}_j} - \overline{x}_i f\left(\left[x_j, x_k\right]\right) - f\left([x_i, x_k]\right)\overline{x}_j \\ &\quad - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left([x_i, x_k]\right) \\ &= f\left(\left[x_j, x_k\right]\right)\overline{x}_i + \overline{x}_k f\left(\left[x_j, x_i\right]\right) - \overline{x}_i f\left(\left[x_j, x_k\right]\right) \\ &\quad - f\left([x_i, x_k]\right)\overline{x}_j - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left([x_i, x_k]\right) \quad \ldots\ldots\ldots\ldots\ldots..(4) \end{aligned}$$

Hatırlatma: $V, F\langle X\rangle$ içerisinde $X$ kümesi tarafından gerilen vektör uzayını göstersin. $v \in V$ alalım. $\overline{x}_r v - v\overline{x}_r$ ($x_r$ kendi kendisi ile kompozisyon oluşturur.) elemanını ModüloG indirgersek $f\left(\left[x_r, f^{-1}(v)\right]\right)$ elemanına indirgenir.

$$v \in V \Rightarrow v = c_1\overline{x}_1 + c_2\overline{x}_2 + \mathbf{L} + c_n\overline{x}_n + \mathbf{L}$$

$$\begin{aligned}\overline{x}_r v - v\overline{x}_r &= c_1\overline{x}_r\overline{x}_1 + c_2\overline{x}_r\overline{x}_2 + \mathbf{L} + c_n\overline{x}_r\overline{x}_n + \mathbf{L} - c_1\overline{x}_1\overline{x}_r - c_2\overline{x}_2\overline{x}_r - \mathbf{L} - c_n\overline{x}_n\overline{x}_r - \mathbf{L} \\ &= c_1(\overline{x}_r\overline{x}_1 - \overline{x}_1\overline{x}_r) + c_2(\overline{x}_r\overline{x}_2 - \overline{x}_2\overline{x}_r) + \mathbf{L} + c_n(\overline{x}_r\overline{x}_n - \overline{x}_n\overline{x}_r) + \mathbf{L}\end{aligned}$$

$g_{ij} = \overline{x}_j\overline{x}_i - \overline{x}_i\overline{x}_j - f\left(\left[x_j, x_i\right]\right), \quad g_{ij} \in I$ olup indirgenmiş olan kısım,

$$\begin{aligned}&= c_1 f([x_r, x_1]) + c_2 f([x_r, x_2]) + \mathbf{L} + c_n f([x_r, x_n]) + \mathbf{L} \\ &= f([x_r, c_1x_1 + c_2x_2 + \mathbf{L} + c_nx_n + \mathbf{L}]) \\ &= f([x_r, f^{-1}(v)])\end{aligned}$$

olduğunu buluruz. O halde 4 ile verilen eşitliğe bakarsak, bu eleman

$$f\left(\left[x_j, x_k\right]\right)\overline{x}_i + \overline{x}_k f\left(\left[x_j, x_i\right]\right) - \overline{x}_i f\left(\left[x_j, x_k\right]\right)$$
$$-f\left(\left[x_i, x_k\right]\right)\overline{x}_j - f\left(\left[x_j, x_i\right]\right)\overline{x}_k + \overline{x}_j f\left(\left[x_i, x_k\right]\right)$$

ModüloG indirgenmiş normal formu,

$$-\left(\overline{x}_i f\left(\left[x_j, x_k\right]\right) - f\left(\left[x_j, x_k\right]\right)\overline{x}_i\right)$$
$$+\overline{x}_k f\left(\left[x_j, x_i\right]\right) - f\left(\left[x_j, x_i\right]\right)\overline{x}_k$$
$$+\overline{x}_j f\left(\left[x_i, x_k\right]\right) - f\left(\left[x_i, x_k\right]\right)\overline{x}_j$$
$$-f\left(\left[x_i, f^{-1}\left(\left[x_j, x_k\right]\right)\right]\right) + f\left(\left[x_k, f^{-1}\left(\left[x_j, x_i\right]\right)\right]\right) + f\left(\left[x_j, f^{-1}\left(\left[x_i, x_k\right]\right)\right]\right)$$
$$f\left(\left[\left[x_j, x_k\right], x_i\right]\right) + f\left(\left[x_k, \left[x_j, x_i\right]\right]\right) + f\left(\left[x_j, \left[x_i, x_k\right]\right]\right)$$
$$f\left(\underbrace{\left[\left[x_j, x_k\right], x_i\right]}_{-\left[\left[x_i, x_j\right], x_k\right] - \left[\left[x_k, x_i\right], x_j\right]} + \underbrace{\left[x_k, \left[x_j, x_i\right]\right]}_{-\left[\left[x_k, x_j\right], x_i\right]} + \underbrace{\left[x_j, \left[x_i, x_k\right]\right]}_{-\left[\left[x_k, x_i\right], x_j\right]}\right)$$
$$\left[\left[x_j, x_i\right], x_k\right] + \left[x_k, \left[x_j, x_i\right]\right] + \left[x_i, \left[x_k, x_i\right]\right] = 0$$

Jakobi özdeşliğinden

$$f\left(\left[\left[x_j, x_k\right], x_i\right] + \left[x_k, \left[x_j, x_i\right]\right] + \left[x_j, \left[x_i, x_k\right]\right]\right) = 0$$

O halde *G, I* idealinin Gröbner bazıdır. Böylece $F\langle X\rangle$ in modülo I normal formundaki hiçbir monomial $LM(g_{ij})$ yi alt kelime olarak içermez. O halde teoremin ispatı $F\langle X\rangle = C(I) + I$ olduğundan ispat biter.

$\Rightarrow {F\langle X\rangle}/{I}$ nın bazı $i > j$ olmak üzere $x_i x_j$ formunda eleman içermez. □

**Sonuç 3.2.2:** $i : L \to U(L)$ dönüşümü injektiftir.

$$i : L \xrightarrow{f} F\langle X\rangle \xrightarrow{p} {F\langle X\rangle}/{I} = U(L) \quad , \qquad i = p \circ f$$

**İspat:** Yukarıdaki teoremin ispatından, $\overline{x}_i$, $i \geq 0$ elemanları ModüloI lineer bağımsızdır. Dolayısı ile

$\forall v \in V,\ v = \sum l_i x_i \Rightarrow f(v) = \sum l_i f(x_i) = \sum l_i \overline{x}_i$ olup $i(v) \in U(L)$ dir

$\Rightarrow i$ injektiftir. □

Son sonuçta *L* yi *U(L)* içindeki görüntüsü ile özdeşleştirelim. Yani *L* yi *U(L)* nin bir alt uzayı gibi düşünebiliriz. Böylece $\overline{x}_i \in X$ yerine $x_i$ alabiliriz. *U(L)* *X* tarafından üretildiğinden, aynı zamanda *L* tarafından üretilen bir cebirdir. (Böyle düşünebiliriz)

**Tanım 3.2.3:** *L* , bir bazı $\{x_1, x_2, \mathbf{L}\}$ olan bir Lie cebiri olsun.

$$x_{i_1}^{k_1} x_{i_2}^{k_2} \ \mathbf{L}\ x_{i_n}^{k_n} \in U(L)$$

elemanına **Standart monomial** denir.

$r : L \longrightarrow gl(V),\ gl(V) = \{ f_k : V \longrightarrow V \mid f \ lineer \}$, *L* nin bir representasyonu olsun.

$r(l)$ **a** $f_l : V \longrightarrow V$

$x, y \in L \Rightarrow r(x)r(y),\ r(L)$ de kapsanmayabilir. Bu çarpmayı bileşke olarak alırsak o zaman $r(x)r(y) \in r(L)$ olur. Bu dönüşümlerin kümesi bir cebir meydana getirir. Bu cebiri $r(L)^*$ ile gösteririz. Buna **Enveloping cebir** denir. (L nin enveloping cebiri)

$r : L \longrightarrow gl(V)$ bir temsil ve $A = r(L)^*$ bu temsile karşılık gelen enveloping cebir olsun. $r$ dönüşümünü $r : U(L) \to A$ dönüşümüne genişletelim

$$r\left(x_{i_1}^{m_1} x_{i_2}^{m_2} \mathbf{L}\, x_{i_k}^{m_k}\right) = r\left(x_{i_1}\right)^{m_1} r\left(x_{i_2}\right)^{m_2} \mathbf{L}\, r\left(x_{i_k}\right)^{m_k}$$

ile genişletmiş olalım. O zaman $r$ örten bir cebir homomorfizmi olur.

$$r(L)^* = A \cong {U(L)}\big/{J} \quad , \; J = \ker r$$

Buradan $L$ nin her enveloping cebiri Evrensel enveloping cebirin (universal enveloping cebir) bölüm cebirine izomorfiktir. Aynı zamanda $r, V$ yi bir $U(L) - Modül$ yapar.

$$av = r(a)v \quad \forall a \in U(L), v \in V$$
$$U(L) \times V \to V$$
$$(a, v) \;\; \mathbf{a} \;\; r(a)v$$

Ayrıca, $U(L) \to A$ olan her homomorfizmin $L$ ye kısıtlaması $r$ ya eşit olmak zorundadır.(çünkü *L, U(L)* yi doğurur.)

Bütün bunlardan, herhangi bir temsilin $L$ den *U(L)* ye genişletmenin bir tek yolu vardır. Bundan başka, $r : U(L) \to gl(V)$ bir representasyon ise $r$ nun $L$ ye kısıtlanışı da aynı zamanda $L$ nin bir temsilidir (representasyonudur). Böylece $L$ nin representasyonu ile *U(L)* nin representasyonu arasında birebir bir ilişki vardır.

# 4. DAİRESEL KELİMELER (CIRCULAR WORDS)

Serbest Lie cebiri ile dairesel kelimeler arasında birçok ilişki vardır. Witt formülü ile verilen homojen boyutun ve ilkel kolyelerin (Primitive neclaces) sayısının eşitliğini hemen örnek olarak verebiliriz. Bu bölümde kendi kendine sonlanan dairesel kelimeleri inceleyeceğiz.

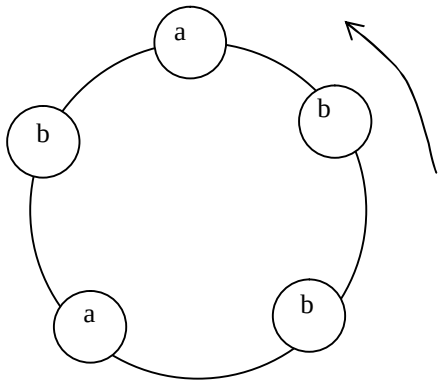
Önce kolyelerin (neclaces) ve ilkel kolyelerin (primitive neclaces) sayısını hesaplamakla işe başlayacağız. Sonra Hall kelimeleri ile ilkel kolyeler arasındaki birebir ve örten bağıntıyı tanımlayacağız. Daha sonra Lyndon kelimelerini üreten yeterli bir algoritma vereceğiz ve verilen bir kelimenin Lyndon kelimelerindeki parçalanışı hesaplayacağız. Hall kelimeleri içindeki bir kelimenin parçalanışı, kelimeler ile ilkel kolyelerin multisetleri arasında birebir ve örten bağıntıyı ifade eder. En son olarak, permütasyonları sabit (invariant) bırakan ve serbest Lie cebiriyle bağlantılı çeşitli simetrik fonksiyonların konusu içinde uygulamaları olan diğer birebir ve örten fonksiyonlar veriyoruz. Bu bölüm ile ilgili temel kavramlar C. Reutenauer (1993) de bulabilirsiniz.

## 4.1. İlkel kolyelerin (primitive neclaces) sayısı

Eğer; bazı $x, y$ kelimeleri için $u = xy$ ve $v = yx$ eşitlikleri sağlanıyorsa ($A^*$ $A$ kümesi üzerindeki serbest monoid olmak üzere ) $A^*$ ın iki kelimesi olan $u, v$ elemanına konjugedirler (conjugate) deriz. '$u$ ve $v$ conjuge' bağıntısı bir denklik bağıntısıdır, buna konjugasyon (conjugation) deriz. Bir denklik sınıfına bir konjuge (conjugacy) sınıfı deriz. Bir konjuge sınıfı ya bir dairesel kelime veya bir kolyedir. Geometrik olarak ifade edecek olursak, bir kolye (neclace) köşeleri $A$ nın elemanları olan düzgün $n-$gen dir. Eğer bir döndürme, bir dönüşüm veya bir homoteti uygulandığında çakışık olan *n*-genler aynır (identical) olarak alınacaktır.

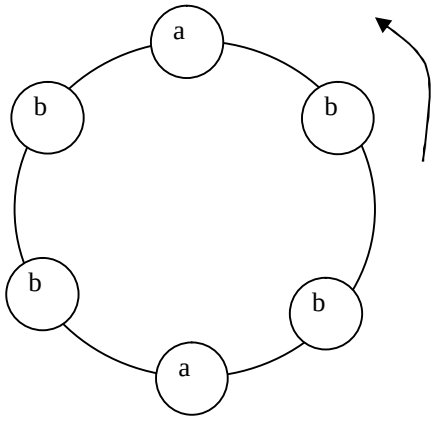
Eğer bir kolyenin kendisini sabit bırakan aşikâr olmayan döndürme yoksa bu kolyeye ilkel kolye (primitive neclace) denir. Bir kelimenin konjuge sınıfı ilkel kolye ise bu kelimeye ilkeldir (primitive) denir. Daha genel olarak; bir kolye, daima onun

uzunluğunu bölen bir en küçük $d$ periyoduna sahiptir ve buna göre konjuge sınıfında $d$ tane eleman vardır ve bu elemanlar, $C = \left\{u_1^{n/d}, u_2^{n/d}, \mathbf{L}, u_d^{n/d}\right\}$ öyle ki $\{u_1, u_2, \mathbf{L}, u_d\}$ bir ilkel konjuge sınıfıdır. Artık $C$ içindeki her bir $w$ kelimesi $d$ periyoduna ve $n/d$ üssüne sahip olduğunu ifade edebiliriz.




*ababb*
*babba*
*abbab*
*bbaba*
*babab*

**Şekil 4-1** *ababb* nin konjuge sınıfı




$abbabb = (abb)^2$
$bbabba = (bba)^2$
$babbab = (bab)^2$

**Şekil 4-2** *abbabb* nin konjuge sınıfı (periyot 3, üs 2)

**Teorem 4.1.1:** $A=\{a_1,a_2,\mathbf{L},a_q\}$, $q$ elemanlı bir alfabe olsun. $n$ uzunluklu ilkel kolyelerin sayısı

$$\frac{1}{n}\sum_{d|n}m(d)q^{n/d} \qquad (4.1.1)$$

dir.

$a_i$ harfinin $n_i$ olaylı ilkel kolyelerin sayısı ($i=1,\mathbf{L},q$ olmak üzere)

$$\frac{1}{n}\sum_{d|n_i}m(d)\binom{n/d}{n_1/d,n_2/d,\mathbf{L},n_q/d}, \qquad (4.1.2)$$

$n=n_1+\mathbf{L}+n_q$

dır. (Burada; "$m$": Möbius fonksiyonu ve "$\binom{\ }{\ }$" : kombinezonu gösterir.)

Teorem 4.1.1 i direkt olarak ispat edebiliriz, fakat üreteç fonksiyonlarının özelliklerini kullanmayı tercih ederiz. Çünkü bu aynı zamanda sonuç için de kullanışlı olacaktır. $x_1,\mathbf{L},x_q$ değişmeli değişkenler olsun. Bir $u$ kolyesinin evaluasyonu (değeri) $Q[x_1,\mathbf{K},x_q]$ polinomu içindeki $x_1^{n_1}\mathbf{L}\,x_q^{n_{q1}}$ monomialidir, burada $i=1,2,\mathbf{L},q$ olmak üzere $u$ kolyesinde $a_i$ den $n_i$ tane vardır. Bir kelimenin evaluasyonu benzer şekilde tanımlanır. Eğer $L$ kolyelerin (veya kelimelerin) bir kümesi ise $L$ nin üreteç fonksiyonu $L$ nin elemanlarının evaluasyonlarının bir toplamıdır. Örneğin; 2 uzunluklu ilkel (primitif) kolyelerin kümesinin üreteç fonksiyonu aşağıdaki gibidir,

$$\sum_{i<j}x_ix_j$$

Çünkü her bir ilkel kolyeler, $i<j$ olmak üzere $a_ia_j$ şeklinde bir tek temsile sahiptir. Şimdi $n$-yinci kuvvetler toplamı simetrik polinom olan $p_n$ i tanımlayalım

$$p_n(x_1,\mathbf{K},x_q)=x_1^n+\mathbf{L}+x_q^n$$

**Teorem 4.1.2:** $n$ uzunluklu ilkel kolyelerin üreteç fonksiyonu

$$\frac{1}{n}\sum_{d|n}m(d)\,p_d^{n/d} \qquad (4.1.3)$$

dir.

**İspat:** $C_e(n)$ ile $n$ uzunluklu ve $e$ periyotlu elemanların kümesini gösterelim. $n$ uzunluklu kelimelerin kümesi olan $A^n$ nin bir parçalanışı

$$A^n = \bigcup_{e|n} C_e(n)$$

şeklindedir.

Uzunluğu $e$ olan ilkel kelimelerin kümesini $P(e)$ ile gösterelim.

$$u \ \mathbf{a} \ u^{n/e}, \quad P(e) \to C_e(n)$$

dönüşümü birebir ve örtendir. Şuna dikkat edelim $P(e)$ içindeki her kelimenin her biri aynı eveluasyonlu $e$ tane konjugesi vardır. Evaluasyonu $ev$ ile gösterirsek

$$ev(u^{n/u}) = ev(u)\Big|_{x_i \to x_i^{n/i}}$$

olur.

Daha genel olarak ifade edecek olursak $A^n$ ın üreteç fonksiyonu

$$(x_1 + \mathbf{L} + x_q)^n = p_1^n(x_1, \mathbf{K}, x_q)$$

dur. Böylece uzunluğu $e$ olan ilkel kolyelerin üreteç fonksiyonunu $l_e(x_1 + \mathbf{L} + x_q)$ ile gösterirsek

$$p_1^n(x_1, \mathbf{K}, x_q) = \sum_{e|n} el_e(x_1^{n/e} + \mathbf{L} + x_q^{n/e}) \qquad 4.1.4$$

elde ederiz. $p_d(x_1, \mathbf{K}, x_q) = p_1(x_1^d, \mathbf{K}, x_q^d)$ olduğunu biliyoruz. Bu nedenle.

$$\frac{1}{n}\sum_{d|n} m(d)\, p_d^{n/d}(x_1, \mathbf{K}, x_q) = \frac{1}{n}\sum_{d|n} m(d)\, p_1^{n/d}(x_1^d, \mathbf{K}, x_q^d)$$
$$= \frac{1}{n}\sum_{d|n} m(d) \sum_{e|n/d} el_e(x_1^{n/e}, \mathbf{K}, x_q^{n/e})$$

olduğunu yazabiliriz. (4.1.4 den $n$ yerine $n/d$ ve $x_i$ yerine $x_i^d$ yazdık.)

$d|n$ ve $e|n/d$ den, $e|n$ ve $d|n/e$ olmasına denktir. Bu da

$$\frac{1}{n}\sum_{e|n} el_e(x_1^{n/e}, \mathbf{K}, x_q^{n/e}) m(d) \sum_{d|n/e} m(d)$$

dir.

$n = e$ olmadıkça ikinci toplam 1 dir ve böylece ifade $l_n(x_1, \mathbf{K}, x_q)$ ya eşit olur. □

**Teorem 4.1.1 in ispatı:** (4.1.3) formülünde $, x_1, \mathbf{K}, x_q = 1$ koyarsak bu (4.1.1) formülünü verir. Dahası $a_i$, $(i = 1, \mathbf{K}, q)$ harflerinden $n_i$ tane bulunan ilkel kolyelerin sayısı (4.1.3) deki $x_1^{n_1} \mathbf{L}\, x_q^{n_q}$ nin katsayısıdır. Bu katsayı (4.1.2) dir . Çünkü

$$p_d^{n/d}(x_1, \mathbf{K}, x_q) \sum_{r_1 + \mathbf{L} + r_q = n/d} \binom{n/d}{r_1, \mathbf{L}, r_q} x_1^{dr_1} \mathbf{L}\, x_q^{dr_q}$$

dur.

**Sonuç 4.1.3:** $n$ uzunluklu kolyeleri üreten fonksiyonu

$$\frac{1}{n} \sum_{d|n} j(d)\, p_d^{n/d}$$

dur. Burada $j$ Euler fonksiyonudur. $q$ harften oluşan bir alfabede, $n$ uzunluklu kolyeleri sayısı

$$\frac{1}{n} \sum_{d|n} j(d) q^{n/d}$$

dir.

İçinde $a_i$ $(i = 1, \mathbf{K}, q)$ harfinden $n_i$ tane bulunan kolyelerin sayısı

$$\frac{1}{n} \sum_{d|n} j(d) \binom{n/d}{n_1/d, \mathbf{L}, n_q/d}$$

dir.

**İspat:** $n$ uzunluklu kelimelerin her bir $C$ konjuge sınıfı için, uzunluğu $n$ yi bölen ilkel kelimelerin bir tek $C'$ konjuge sınıfı vardır. $n/e$ dersek $C = \{u^e \,|\, u \in C'\}$ olduğunu yazabiliriz. Bu durum, kolyelerin (veya prmimitif kolyelerin) üreteç fonksiyonu olan $k_n$ (veya $l_n$) ile aralarındaki

$$k_n(x_1, \mathbf{K}, x_q) = \sum_{e|n} l_{n/e}(x_1^e, \mathbf{K}, x_q^e)$$

özelliği verir.

Teorem 4.1.2 den

$$k_n(x_1,\mathbf{K},x_q)=\sum_{e|n}\frac{1}{n/e}\sum_{f|n/e}m(f)p_f^{n/ef}(x_1^e,\mathbf{K},x_q^e)$$
$$=\sum_{e|n}\frac{e}{n}\sum_{ef|n}m(f)p_{ef}^{n/ef}(x_1,\mathbf{K},x_q)$$
$$=\frac{1}{n}\sum_{e|n}p_d^{n/d}(x_1,\mathbf{K},x_q)\sum_{e|d}em\left(\frac{d}{e}\right)$$
$$=\frac{1}{n}\sum_{d|n}j(d)p_d^{n/d}(x_1,\mathbf{K},x_q),$$

elde edilir. Çünkü $p_f(x_1^e,\mathbf{K},x_q^e)=p_{ef}(x_1,\mathbf{K},x_q)$ ve $j(d)=\sum_{e|d}em(d/e)$ dir.

Diğer iki formulü elde etmede gene Teorem 4.1.1 den görülür. □

## 4.2. Hall Kelimeleri ve İlkel Kolyeler (Primitive Neclaces)

Kelimeler üzerindeki konjuge olma bağıntısı bir denklik bağıntısıdır. Bu yüzden ilkelliği (primitive) ve periyodikliği korur. Şuna dikkat edelim $H$ ilkel konjuge sınıfların temsili kümesi ise bu küme

$$\{h^n \,|\, h\in H, n\geq 1\}$$

dır. Uzunluğu pozitif olan tüm konjuge sınıfların temsilinin bir kümesidir. Aşağıdaki sonuç Schützenberger (1965) in teoreminin özel bir durumudur.

**Teorem 4.2.1 (**Reutenauer C. 1993)**:** $H$, $A^*$ bir alt kümesi olsun. Bir $\geq$ sıralaması alınsın. Kabuledelim ki $A^*$ ın içindeki her bir elemanın bir tek parçalanışı olsun. Yani

$$h_1\,\mathbf{L}\,h_n, n\geq 0, h_i\in H, h_1\geq \mathbf{L}\geq h_n$$

O zaman $H$, ilkel konjuge sınıfların temsili bir kümesidir.

**Sonuç 4.2.2:** $H$, $A^*$ da bir Hall kümesi olsun. O zaman $H$ deki her kelime ilkeldir ve boş olmayan her kelime, $h^n, h\in H, n\geq 1$ formundaki bir tek kelimenin konjugesidir.

**Örnek 4.2.3:** Daha önceki Örnekteki Hall kümesini ele alalım; Biz $abaababa$ kelimesine konjuge olan Hall kelimesini bulmak istiyoruz. $aabab, aab, a, ab, b$ lerin Hall kelimeleri olduğu biliniyor ve artan sıra içinde yazıyoruz. Algoritma şunu verir;

$$(a,b,a,a,b,a,b,a) \to (ab,a,a,b,a,b,a) \to (ab,a,ab,a,b,a)$$
$$\to (ab,a,ab,ab,a) \to (aab,a,ab,ab)$$
$$\to (aab,aab,ab) \to (aab,aabab)$$
$$\to (aababaab)$$

Son kelime istenilen Hall kelimesidir.

$A$ tam sıralamalı olan bir alfabe olsun ve $A^*$ alfabetik sıralı olsun. Bir Lyndon kelimesi, trivial sağ çarpanı olmayan kelimelerin tümünden daha küçük olan bir kelimedir.

**Sonuç 4.2.4:** Bir $w$ kelimesinin Lyndon kelimesi olabilmesi için gerek ve yeter koşul primitif ve konjuge sınıfındaki en küçük kelime olmasıdır.

**İspat:** Lyndon kelimelerinin kümesi olan $L$, özel bir Hall kümesidir. O halde, Sonuç 4.2.2 den , $L$, ilkel kelimelerin konjuge sınıflarının temsilinin bir kümesidir. Şimdi, eğer $w$ bir Lyndon kelimesi $w = uv,\ u,v \neq 1$ ile $w$ nun bir konjugesi $vu$ olsun. O zaman Lyndon kelimesinin tanımından $w < v$ olduğunu biliyoruz. $v$, $w$ dan daha kısa olduğundan $w$, $v$ nin bir ön eki olamaz bu yüzden $w < vu$ olduğunu anlarız. Bu nedenle her Lyndon kelimesi onun konjuge sınıfındaki elemanların en küçüğüdür, bu da ispatı sonlandırır.

### 4.3. Lyndon kelimelerinin üretilmesi

$A$, tam sıralı sonlu bir alfabe olsun. $A^*$, alfabetik sıralı olsun ve $L$ ile Lyndon kelimelerinin kümesini gösterelim. Sabit bir $N \geq 1$ tamsayısı için uzunlukları $\leq N$ olan Lyndon kelimelerinin kümesinin $L_N$ ile gösterelim. Şimdi bir fonksiyon tanımlayalım:

$$l_N : L_N \setminus \{z\} \to L_N$$

(Burada $z$, $A$ içindeki en büyük harftir) Şart şu: $l_N(u)$, $\{x \in L_N \,|\, x > u\}$ kümesi içinde alfabetik sıraya göre en küçük kelimedir.

Şunu gözlemleyelim; $l_N$ basit olarak, sonlu tam sıralı $L_N$ kümesi içindeki "sonraki eleman fonksiyonudur" (en büyük eleman $z$ iken). Bu nedenle, $l_N$ nin bilgisi uzunluğu $\leq N$ olan tüm kelimelerin üretilmesi için basit bir algoritma verir : (en küçük harf olan ) $a$ dan başlanır, $u = l_N(a)$ hesaplanır, sonra $l_N(u)$ ve sonunda $z$ elde edilene kadar böyle devam edilir.

Bundan sonraki sonucu, tam olarak Duval (1988), $l_N$ fonksiyonunun çok hızlı ve verimli bir şekilde hesaplanabilir olduğunu göstermiştir. Bunu için iki tanıma ihtiyacımız var. Verilen bir $u$ kelimesi için , $D_N(u) = u^k p$ kelimesine $u$ nun $N$ uzunluklu periyodik genişlemesi denir. ( Burada $p$ , $u$ nun boş olmayan bir öneki $k$ ve $p$, $k \in \mathbf{N}$ , $|u^k p| = N$ olmak üzere bir tek şekilde tanımlanmışlardır). Diğer taraftan, $D_N(u)$, $uuu...u...$ "sonlu kelimeler" in $N$ uzunluklu bir tek önekidir. Not: Bu önekler, $u$ nun boş olmayan bir $p$ ön eki olmak üzere genellikle $u^k p$ gibi yazılabilen $N \geq 1$ uzunluklu olan önekleridir.

Şimdi, şöyle bir fonksiyon tanımlayalım: $S : A^* \backslash z^* \to A^*$. Burada $z^*$, $z$ nin kuvvet kümesini göstersin,

$$S(w) = \min\left\{x \in A^* \,|\, x > w, w|x| \leq |w|\right\}$$

$S(w)$ yi hesaplayan basit bir algoritma var. $z$ den farklı bir $a$ harfi olmak üzere, $w = vaz^i$ olsun. Sonlu tam sıralı $A$ içinde $a$ dan sonraki harf $b$ olsun. O zaman,

$$S(w) = vb \qquad (4.3.1)$$

dir.

**Teorem 4.3.1:** $l_N$ fonksiyonu, $S \circ D_N$ bileşkesine eşittir.

Önce birkaç lemma ispatlayalım.

**Lemma 4.3.2:** Eğer $u < v$ olacak şekilde $u, v$ iki Lyndon kelimesi ve $n, p \geq 1$ olan iki tamsayı ise $u^n v^p$ bir Lyndon kelimesidir.

**İspat :** $n = p = 1$ olması durumunda ispat açıktır. Genel durumda tümevarım ile izlenir, bir kere şuna dikkat edelim $u < u^n v^p < v$ (son eşitsizliktir çünkü $u^n v^p$ bir Lyndon kelimesidir).

Sonraki lemma gösterir ki bir Lyndon kelimesi ile onun periyodik genişlemesi arasında herhangi bir sırada Lyndon kelimesi yoktur. Bu, Teorem 4.3.1 in ispatı için ilk adımdır.

**Lemma 4.3.3:** $u, v, |u| \leq N$ ve $u \leq w \leq D_N(u)$ olacak şekilde iki Lyndon kelimesi olsun o zaman $w = u$ dir.

**İspat:** Farz edelim ki $w$ , $u \leq w \leq D_N(u)$ durumunu sağlasın. Gösteririz ki $w$ bir aşikâr olmayan $s$ son ekine sahiptir ve $u$, bir $p$ ön ekine sahiptir öyle ki $s \leq p$ dir. Bu nedenle $s \leq p \leq u < w$ dir ve $w$ bir Lyndon kelimesi değildir.

Önce farz edelim ki $w, D_N(u)$ nin bir ön eki olsun. O halde $D_N(u)$, $u$ nun bazı kuvvetleri olan $u^q$ nun bir önekidir, aynı zamanda $w$ nun da böyle olduğuna dikkat edelim. Bu nedenle bazı $k \geq 0$ tam sayı olmak üzere, $u$ nun bazı $p$ önleri için $w = u^k p$ dir. Buradan $w \neq 1$ olduğunu görürüz ve trivial olmayan $p$ seçebiliriz (Eğer zorunlu ise $p = u$ ile $p = 1$ ve $k - 1$ ile $k$ yer değiştirir). Bu durumda $p = s$, $w = u^k p$ nin aşikâr olmayan bir son ekidir.

Şimdi farz edelim ki $w$, $D_N(u)$ nin bir ön eki olmasın. Öncelikle alfabetik sırlamanın tanımından $w \leq D_N(u)$ olduğunu ve bazı $a < b$ harfleri ve $x, w', t$ kelimeleri için $w = xaw'$, $D_N(u) = xbt$ olduğunu biliyoruz. $D_N$ nin tanımından, $k \geq 0$ için $x = u^k u'$ dir ve $u$ nun ön eki $p = u'b$ dir. $s = u'aw'$ olsun o zaman $w = u^k u'aw' = u^k s$ ve $s$, $w$ nun aşikar olmayan bir son ekidir. Daha da ileri gidersek

$$s = u'aw' < u'b = p$$ olur.

**Lemma 4.3.4**: $s \neq 1$ olmak üzere $u = ps$ bir Lyndon kelimesi olsun. $c$, $s < c$ olacak şekilde bir harf olsun. O zaman $pc$ bir Lyndon kelimesidir.

**İspat:** $t$, $pc$ nin boş olmayan bir öz (proper) son eki olsun. $pc < t$ olduğunu gösteririz ki bu $pc$ nin bir Lyndon kelimesi olduğunu ispatlayacaktır. $t'$, $p$ nin bir öz (proper) son eki olduğunda $t = t'c$ olduğunu elde ederiz. O zaman $t = t's$, $ps$ nin boş olmayan bir öz son ekidir. Ve bu nedenle $ps < t's$ dir çünkü $ps = u$ bir Lyndon kelimesidir. Devam edersek $s < c$ olduğundan $t's < t'c$ dir ve $p$, $ps$ nin bir öz ön ekidir. Bu nedenle $p < ps$. Böylece $p < ps < t's < t'c = t$ dir ve $p < t$ dir. Farz edelim ki $p$, $t$ nin bir ön eki olsun, $t$, $pc$ nin bir öz son eki olmasından şunu elde ederiz; $|p|+1 = |pc| > |t| \Rightarrow |p| \geq |t|$ dir. Ve $p = t$ yi elde etmiş oluruz. Bu ise doğru değildir. Çünkü bir önceki eşitsizlikte $p < t$ idi. Bu nedenle dikkat edersek $pc < t$ dir.

**Sonuç 4.3.5:** $u$, alfabenin en büyük harfi olmayan bir Lyndon kelimesi, $p$, $u$ nun boş olmayan bir öneki ve $k$ bir tam sayı olsun. O zaman $S(u^k p)$ bir Lyndon kelimedir.

**İspat:** $z, A$ daki en büyük harf olsun. O zaman $p$ kesinlikle $z$ ile başlamaz. Gerçektende, aksi takdirde $u = z'u$ olur ve $u \geq z \geq u$ nun son harfi olur; öncelikle $u$ Lyndon kelimesidir bu şunu gerektirir $u$, onun son harfine eşittir. Bu nedenle $u = z$ dir, bu ise ilk kabullenmemize karşıdır yani çelişkidir. Özellikle $p$, $z$ nin bir kuvveti değildir. Bu nedenle (4.3.1) den dolayı $a \in A \backslash z$ \için $p = p_1 a z^i$ yi elde ederiz. $A$ içinde $a$ sonra gelen harf $b$ ise $S(p) = p_1 b$ dir. $a$ ile başlayan bazı $s_1$ kelimeleri için $u = p_1 s_1$ olduğunu elde ederiz. Bu nedenle $s_1 < b$ dir ve Lemma 5.3.4 den $p_1 b$ nin bir Lyndon kelimesi olduğunu anlarız. Şimdi (4.3.1) e geri dönersek $S(u^k p) = S(u^k p_1 a z^i) = u^k p_1 b$ yi elde ederiz. $p_1 a$, $u$ nun bir ön eki olduğundan $u < p_1 b$ yi elde ederiz ve Lemma 5.3.2 i kullanarak ispatı halletmiş oluruz.

**Teorem 4.3.1 in İspatı:** $u$ uzunluğu $\le N, u \ne z$ olan bir Lyndon kelimesi ve $w = l_N(u)$ olsun. O zaman tanımdan, $w$, $\{x \in A^* \mid x > u, |x| \le N\}$ kümesi içindeki en küçük Lyndon kelimesidir.

Lemma 5.3.3 dan $w > D_N(u)$ olduğunu biliyoruz. Bu nedenle $w$,

$$\{x \in A^* \mid x > D_N(u), |x| \le N\}$$

kümesindeki en küçük Lyndon kelimesidir. Fakat $S(D_N(u))$ bu küme içindeki en küçük kelimedir. Bazı $k$ tam sayıları ve $u$ nun bazı boş olmayan $p$ öneki için $D_N(u) = u^k p$ Olduğu biliniyor. Biz Sonuç 4.3.5 den $S(D_N(u))$ nin bir Lyndon kelimesi olduğunu biliyoruz. Bu nedenle $S(D_N(u)) = w = l_N(u)$ dur.

**Örnek: 4.3.6:** $N = 9$, $A = \{a < b\}$ olsun. O zaman $u = aabbb$ bir Lyndon kelimesidir. O halde şunu söyleriz;

$$l_9(u) = S \circ D_9(u) = S(aabbbaabb) = aabbbab$$

dir.

## 4.4. Lyndon Kelimelerine Parçalanış

$A$ tam sıralı sonlu alfabe olmak üzere $L$, $A^*$ üzerinde alfabetik sırlamaya göre Lyndon kelimelerinin kümesi olarak bilinir. $L$ , Özel bir Hall kümesidir. Bu nedenle $A^*$ daki her $w$ kelimesi tek türlü azalan parçalanışa sahiptir.

$$w = l_1 ... l_n, \quad l_i \in L, l_i \ge \mathbf{L} \ge l_n \qquad (4.4.1)$$

(4.4.1) deki parçalanışın varlığı ve tekliği direkt olarak ispatlanabilir. Gerçekten her kelime, Lyndon kelimeleri içinde bir parçalanışa sahiptir (örneğin $w$ , içindeki harflerin çarpımıdır) ve çarpanlarının sayısı en azdır. Şimdi Çarpanları en az olan bir parçalanış alalım. Lemma 5.3.2 dan, $k, l$ Lyndon kelimesi olup $k < l$ iken $kl \in L$ olur. Bu minimal parçalanış azalan olmak zorundadır. Bu durum (4.4.1) deki varlığını ispatlar. Teklik ise aşağıdaki sonucun bir neticesidir.

**Lemma 4.4.1:** $w$ nun (4.4.1) seklindeki parçalanışı için aşağıdaki özellikler sağlanır:

*(ii)* $l_n$, $w$ nun aşikâr olmayan en küçük son ekidir.

*(iii)* $l_n$, $w$ nun en büyük son eki olan bir Lyndon kelimesidir.

*(iv)* $l_1$, $w$ nun en büyük ön eki olan bir Lyndon kelimesidir.

**İspat:** $s$, $w$ nun aşikâr olmayan bir son eki olsun. Bu durumda $l'$, $l_i$ nin $1 \le i \le n$ olacak şekildeki boş olmayan bir son eki olmak üzere $s = l_i' l_{i+1} \mathbf{K} l_n$ dir.

*(i)* $l_i$ bir Lyndon kelimesidir. Böylece $l_i \le l_i' \le l_i' l_{i+1} \mathbf{K} l_n = s$ dir ve $l_n \le l_i$ olduğundan $l_n$, $w$ ni aşikâr olmayan en küçük son ekidir

*(ii)* Farz edelim ki $s$, $l_n$ den daha büyük olsun. O zaman, $i < n$ olduğundan $l_n' \le s$ olmasını gerektirir. (i) deki sonucu kullanırsak, $l_n < s$ olduğunu anlarız. Bu sonuç $s$ nin kendisinden daha küçük aşikâr olmayan bir son eke sahip olduğunu söyler, bu nedenle s bir Lyndon kelimesi olamaz.

*(iii)* p, w nun , $l_1$ den kesinlikle daha uzun olan bir öneki olsun. O zaman $2 \le k \le n$ ve $l_j$ nin boş olmayan bir öneki $l_j'$ olmak üzere $p = l_1 \mathbf{K} l_{j-1} l_j'$ şeklindedir. (4.4.1) i kullandığımızda $l_j' \le l_j \le l_1 < l_1 \mathbf{K} l_{j-1} l_j' = p$ sonucuna varırız bu ise p nin bir Lyndon kelimesi olmadığını gösterir. □

Duval (1978, 1983) tarafından verilen çok daha etkin bir algoritma vardır. Şimdi bunu tanımlayacağız. Bu amaçla burada $p$, $u$ nun bir öneki ve $k$ tam sayı olmak üzere, $u^k p$ şeklindeki kelimedeki, $u$ kelimesinin sesquipowerını tanımlayacağız. Genelliği kaybetmeksizin, $p \ne u$ ($p$, $u$ dan farklıdır) olarak alabiliriz. Eğer $k \ge 1$ ise bir sesquipowere nontrivial denir. Lyndon kelimelerinin nontrivial sesquipower'lerinin kümesini $S$ ile gösterelim. S nin bir $u$ elemanı daima

$$u = l^k p \qquad (4.4.2)$$

şeklinde bir temsile sahiptir. Burada $l \in L$, $p$, $l$ nin öz öneki ve $k \geq 1$ dir. $S$ nin verilen bir u elemanı için temsil (4.4.2) tektir: Gerçekten $p$ nin Lyndon kelimeleri içerisindeki azalan parçalanışı $p = h_1 \mathbf{K} h_q$ olsun o zaman $h_1 \leq p < l$ dir. Bu yüzden $u$ nun Lyndon kelimeleri içindeki azalan parçalanışı $l^k h_1 \mathbf{K} h_q$ dır. Buradan parçalanışın bir tek olması demektir.

Bu bölümün sonunda, $S$ nin elemanlarının (4.4.2) deki tek türlü temsile sahip olduğunu anlayacağız.

$S$ nin (4.4.2) deki temsile sahip bir $u$ kelimesi için, $s$ kelimesi ve bir $a$ harfi olmak üzere $l = pas$ yazabiliriz. $a = e(u)$ şeklinde yazalım. Şimdi $S \times A^*$ kümesi üzerinde $\rightarrow$ ile göstereceğimiz bir ikili bağıntıyı, $u \in S$, $b \in A$, $v \in A^*$ olmak üzere eğer $e(u) \leq b$ ise

$$(u, bv) \rightarrow (ub, v)$$

olarak tanımlayalım.

$e(u) \leq b$ olması $ub \in S$ olmasını gerektirdiğinden bu bağıntı iyi tanımlanmıştır. Gerçekten $u = l^k p$ ise (4.4.2) deki gibi ve $l = pas$ ise bu durumda $a = e(u)$ dur. Bu yüzden, ya $a = b$ dir bu durumda $ub = (pas)^k pa$ nin S de olduğu açıktır ya da $a < b$ ve $as < b$ dir. Bu yüzden Lemma 5.3.4 den $pb$ , $l < pb$ olacak şekilde bir Lyndon kelimesidir öyle ki Lemma 4.3.2 den $ub = l^k pb$ bir Lyndon kelimesidir bu nedenle $S$ içindedir.

$\rightarrow$ un geçişken bir kapanışını $\xrightarrow{*}$ ile gösterelim. Bu durumda $S \times A^*$ üzerinde kısmi bir sıralama olduğu açıktır. Ve eğer $x \rightarrow y$ olacak şekilde $S \times A^*$ de hiç bir $y$ elemanı yoksa $x \in S \times A^*$ ye maksimal denir. $S \times A^*$ deki her $x$ için $S \times A^*$ da $x \xrightarrow{*} y$ olacak şekilde bir tek maksimal $y$ elemanı olduğu açıktır.

Parçalanış algoritmasını bir sonraki teoremde tanımladık.

Not: bir kelimenin Lyndon kelimelerine azalan parçalanışı şöyle yazılabilir;

$$w = l_1^{k_1} \mathbf{K} l_p^{k_p}, \quad l_1 > \mathbf{L} > l_p, \quad k_1, \mathbf{K}, k_p \geq 1 \qquad (4.4.3)$$

**Teorem 4.4.2:** $w$ (4.4.3) de olduğu gibi parçalanışa sahip olan bir kelime, $c$ de $w = cw'$ , olacak şekilde $w$ nun ilk harfi, $(u,v)$, $S \times A^*$ daki tek maksimal eleman öyle ki $(c,w') \xrightarrow{*} (u,v)$ olsun. (Burada $u$ (4.4.2) deki gibidir). O zaman

$$l_1 = l \;\; ve \;\; k_1 = k$$

dır.

Diğer bir ifade ile $\rightarrow$ rewriting sistemi (yeniden yazma sistemi), $w$ nun parçalanıştaki ilk Lyndon kelimelerinin kuvvetini hesaplamamıza yarar. Bu yüzden sonuna kadar $w$ nun yerine $pv$ yazılarak devam edilip gidilir.

**Örnek 4.4.3:** $w = abbabbababb$, $a < b$. $S$ indeki her $u$ (4.4.2) deki gibi yazılmış olsun. $e(u)$ harflerini koyu şekilde yazalım. Şunu elde ederiz;

$$\begin{aligned}(\text{a,bbabbababb}) &\rightarrow (\text{ab,babbababb}) \rightarrow (\text{abb,abbababb}) \\ &\rightarrow ((\text{abb})\text{a,bbababb}) \\ &\rightarrow ((\text{abb})\text{ab,bababb})((\text{abb})^2\text{,ababb}) \\ &\rightarrow ((\text{abb})^2\text{a,babb}) \\ &\rightarrow ((\text{abb})^2\text{ab,abb})\end{aligned}$$

Sonuncu maksimaldir çünkü $b > a$ dır. Bu yüzden $w = (abb)^2 s$ dir ve $s$ ile devam edersek : $(a,\ babb) \rightarrow (ab, abb) \rightarrow ((ab)a, bb) \rightarrow ((ab)^2, b) \rightarrow (ababb, 1)$. Bu yüzden $s$ bir Lyndon kelimesidir ve $w$ nun parçalanışı $(abb)^2(ababb)$ dir.

**İspat:** $(c,w') \xrightarrow{*} (u,v)$ olduğundan $w = cw' = uv = l^k pv$ elde ederiz. $p \in A$, $l = pas$ olsun. $pv$ nin Lyndon kelimelerine azalan parçalanışı $pv = h_1 \mathbf{K} h_q$ olsun. Ya; $h_1$ , $p$ nin bir önekidir. Bu nedenle $h_1 \le p < l$ olur ya da; $p$ $h_1$ in bir öz öneki dir: $pbh'$ dir, burada $b$ , $v$ nin ilk harfidir; sonra $(u,v)$ maksimal olduğundan $a > b$ almalıyız, bu yüzden $h_1 = pbh' < pas = l$ dir. Her iki durumda da $h_1 < l$ dir. Bu da şunu gösterir; $w$ nun Lyndon kelimelerindeki azalan parçalanışı $l^k h_1 \mathbf{K} h_q$ dur ve böyle devam edersek

$$l^k = l_1^{k_1}, \quad h_1 \mathbf{K} h_q = l_2^{k_2} \mathbf{K} l_p^{k_p}$$

olur.

Bu teoremin ispatı algoritmanın lineer olduğunu gösterir. Daha özel olarak ifade edersek $w$ yu parçalamak için $A$ nın harfleri arasından en çok $2|w|$ defa karşılaştırmaya ihtiyaç vardır.

## 4.5. İlkel Kolyelerin Multikümeleri ve Kelimeleri

Bir $E$ kümesi için yeni bir tanım yapacağız. $M : E \to N$ bir dönüşüme $E$ nin elemanlarının multikümesi (katlı kümesi) denir. $E$ deki $e$ için, $M(e)$ ye $M$ deki $e$ nin multiplicity (katlılığı) denir. Eğer bunun, kardinalitesi (kümenin eleman sayısı) sonlu ise $\sum_{e \in E} M(e) < \infty$ sonludur.

Teorem 4.2.1 ün hipotezine göre $A^*$ bir H alt kümesi verilsin. 4.2.1 de verilene göre $A^*$ nın kelimeleri ile $H$ elemanlarının multikümeleri arasında aşikâr bir bijeksiyon vardır. Bu durum $H$ Hall kümesi için bu doğrudur veya Lyndon kelimelerinin kümesi olan $L = H$ içinde doğrudur. Üstelik ileri gidersek Sonuç 4.2.2, $H$ ile ilkel kolyelerin kümesi arasında bir bijeksiyonun olduğunu söyler. Bu yüzden aşağıdaki sonuca ulaşırız. Multikümelerin bileşenlerindeki evaluasyonu (veya sırasıyla uzunluğu ) evaluasyonların çarpımıdır (veya uzunluların toplamıdır).

**Teorem 4.5.1:** $A^*$ ın bir Hall kümesi verilsin (özellikle Lyndon kelimelerinin kümesi), o zaman aşağıdaki üç küme arasında evaluasyonu koruyan kanonik bir bijeksiyon vardır;

*(i)* $n$ uzunluklu kelimelerin kümesi

*(ii)* $n$ uzunluklu Hall kelimelerinin multikümelerinin kümesi

*(iii)* $n$ uzunluklu ilkel kolyelerin multikümelerinin kümesi

Şimdi, bir öncekinden daha değişmez özellikleri sahip ilkel kolyelerin multikümeleri ve kelimeleri arasında bir başka bijeksiyon tanımlayacağız. Bu bijeksiyon Serbest Lie cebirleri ile bağlantılı olan çeşitli simetrik fonksiyonlar ile çalışmada kullanışlı olacaktır.

$A$ tamsıralı bir alfabe olmak üzere $a_i \in A$ için $A^*$ ın bir elemanı $w = a_1 \mathbf{K} a_n$ olsun. $[n] = \{1, \mathbf{K}, n\}$ olmak üzere $d_w : [n]$ **a** $A \times [n]$, $d(i) = (a_n, i)$ şeklinde bir fonksiyon tanımlayalım. $d_w$ nun injective (bire çok) olduğu açıktır. $A \times [n]$ sözlük sıralamasına göre sıralıdır. $d_w(i) < d_w(j)$ şartından, üzerinde bir tam sıralama tanımlar. Bu şart aşağıdaki özelliklerle birlikte bir denkliktir,

$$(a_i < a_j) \text{ veya } (a_i = a_j \quad ve \quad i < j). \qquad (4.5.1)$$

$[n]$ üzerindeki bu tam sıralamaya $w$ nun standart numaralanması denir öyle ki: $w$ daki en küçük harften başlayarak, $w$ nun harflerinin pozisyonlarının, soldan sağa doğru numara sırasına göre belirlenir. Daha sonra ikinci en küçük kelimeden devam edilerek numaralandırma sürdürülür. (Örnek 4.5.1 bak)

$w$ nun standart permütasyonu, $s(i) < s(j)$ , $d_w(i) < d_w(j)$ de denklik olmak üzere $[n]$ nin $st(w) = s$ olan bir tek permütasyondur. Eğer yukarıdaki gibi $w$ daki harflerin pozisyonlarının bir numarası olarak alırsak, $[n]$ üzerinde bir kelime gibi görüp, $s$ direkt olarak elde edilebilir.

Aşağıdaki bilginin doğrulanmasını kolayca görülebilir.

$$s(i) = \left| \{ j \mid 1 \le j \le n,\ a_j < a_i \} \right| + \left| \{ j \mid 1 \le j \le i,\ a_i = a_j \} \right|.$$

**Örnek: 4.5.2:**

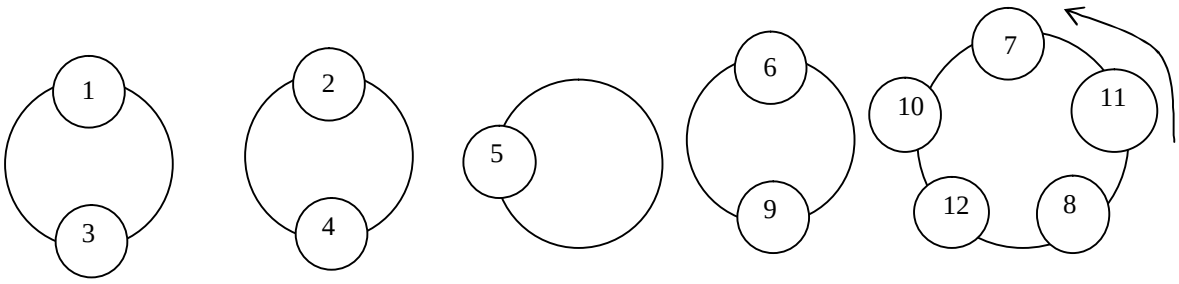
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $w$ = | b | b | a | a | b | d | d | d | b | d | b | c |
| $s$ = | 3 | 4 | 1 | 2 | 5 | 9 | 10 | 11 | 6 | 12 | 7 | 8 |

Daha önce olduğu gibi $s$ ve $w$ lar için , $s$ nun bir $c = (i_1, \mathbf{N}, i_k)$ devrini düşünelim. $a_i a_{i_1} \mathbf{K} a_{i_k}$ Kelimelerinin denklik sınıfları olan kolyeleri $v_c(w)$ ile gösterelim.

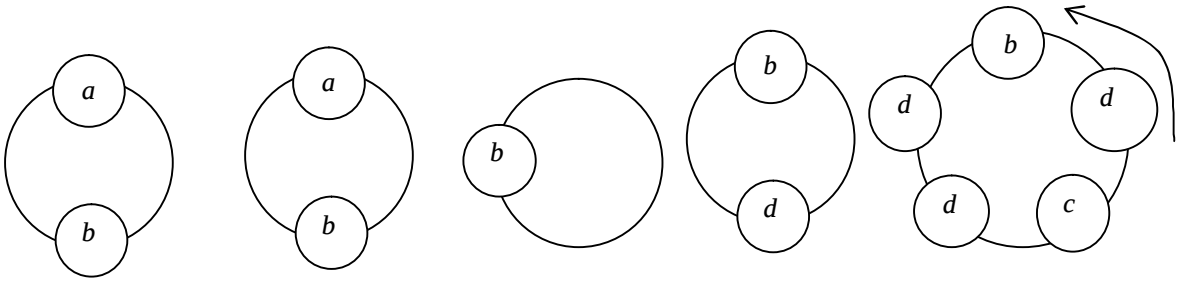
O zaman, $M(w)$ kolyesinin bir multiset tanımı $\{ v_c(w) \mid c,\ st(w) \text{ nin devridir} \}$ multiseti olur. Daha da formülleştirirsek; her $v$ kolyesi için $v_c(w) = v$ olacak şekilde $M(w)(v) = st(w)$ nin c devrinin sayısı tarafından $M(w)$ tanımlanır.

Dikkat edersek, $w$ nun evaluasyonu (değeri) $M(w)$ evaluasyonuna eşittir.

**Örnek 4.5.3:** $s$ ve $w$ lar Örnek 4.5.2 olduğu gibi olsun. $s$ nun devirleri şekil 4.3 te gösterilmiştir. $M(w)$ multiseti, $a_{s^{-1}(i)}$ ye göre her $i$ rakamının yerinin değiştirilmesi ile elde edilir ( $a_i -$ yerine bu kolyeler değişmez). Şekil 4.4 e bak.




**Şekil 4-3**




**Şekil 4-4**

**Teorem 4.5.4**: $w$ **a** $M(w)$ fonksiyonu kelimelerin kümesinden ilkel kolyelerin sonlu multisetlerinin kümesine evaluasyonu koruyan bir bijeksiyondur.

İspatta $W$ ters bijeksiyonu anlatılmıştır.

**İspat:** (a) Aşağıdaki gibi kurulan, evaluasyonu koruyan (evaluation-preserving) $W$ fonksiyonu, ilkel kolyelerin sonlu multisetlerinin kümesinden $A^*$ a şöyle tanımlansın: $M \circ W = id$. O zaman $W$ injectif (bire çok ) olacaktır. Bu nedenle Teorem 4.5.1 e göre surjectif (örten) olur. Çünkü verilen bir evaluasyonlu kelimelerin kümesi sonludur. Bu nedenle $M$ , $A^*$ dan ilkel kolyelerin multisetlerinin kümesi üzerine bir bijeksiyon dur.

(b) $P$, $A^{¥}$ de periyodik dizilerin kümesi olsun.

$p : A^{+} \to P$, $(b_i \in A)$ için

$p(b_1 \mathbf{K} b_k) = (b_1, \mathbf{K}, b_k, b_1, \mathbf{K}, b_k, \mathbf{K})$

olmak üzere $P$ deki her eleman $p(u)$ şeklindedir. Şuna dikkat edelim; $p$, ilkel kelimeler ile periyodik diziler arasında bir bijeksiyon tanımlar.

$z : A^{¥} \to A^{¥}$

$z(a_0, a_1, a_2, \mathbf{K}) = (a_1, a_2, a_3, \mathbf{K})$

Şeklinde tanımlanan bir değiştirme fonksiyonu (shift mapping) olsun. Bunun $P$ ye kısıtlaması $P$ nin bir permütasyonudur. Şunu elde ederiz

$$z(p(b_1 \mathbf{K} b_k)) = p(b_2 \mathbf{K} b_k b_1) \qquad (4.5.2)$$

ve

$$z^{-1}(p(b_1 \mathbf{K} b_k)) = p(b_k b_1 \mathbf{K} b_{k-1}) \qquad (4.5.3)$$

Bu şunu gösterir, $u$ ilkel, $p(v)$ dizisine göre $v$ ile $u$ konjuge olur ve $p(u)$ nun $z^{-1}$ altındaki orbiti (yörüngesi) tam $|u|$ elemanlıdır yani $u$ nun konjugelerinin sayısı kadardır.

$A^{¥}$ üzerinde sözlük sıralaması koyalım ve aynısını $A^{¥} \times ¥$ içinde yapalım. $f : A^{¥} \to A$ fonksiyonu, dizinin ilk elemanının üzerine göndersin. $z$ ve $f$, $P \times ¥$ de $z(s,l) = (z(s), l)$ ve $f(s,l) = f(s)$ ile bir genişlemedir (extend).

(c) Bir u kelimesinin konjugasyonunu $(u)$ ile onun ters dönüşümünü de $\overline{u}$ ile gösterelim ( yani şöyle tanımlanmış; eğer $u = a_1 \mathbf{K} a_k$, $a_i \in A$ ise $\overline{u} = a_k \mathbf{K} a_1$). $N$, $n$ uzunluklu ilkel kolyelerin sonlu bir multiseti olsun. $N$ ile $P \times ¥$ nin $E$ alt kümesini birleştirirsek, $[0] = \varnothing$ olmak üzere

$$E = \bigcup_{u \in A^{*}} \left\{ p\left(\overline{u}\right) \right\} \times [N((u))] \qquad (4.5.4)$$

**Not:** E nin kardinalitesi n dir çünkü ilkel olan bir u kelimesinin konjuge sınıfında $|u|$ eleman vardır. Not: aynı zamanda E, $A^{¥} \times ¥$ nin < tam sıralamasını miras olarak alır.

Şöyle yazarız $E=\{e_1<e_2<\mathbf{L}<e_n\}$. Eğer u,v konjuge kelimeler ise o zaman N((u))=N((v)) dir. Ve $\tilde{u},\tilde{v}$ konjugedirler. O halde 4.5.2 ve 4.5.3 den E nin z ye kısıtlaması $E\to E$ ye bir permütasyondur. $s$ , $[n]$ nin $s(i)=e_i$ olmak üzere $s=d^{-1}\mathbf{o}z^{-1}\mathbf{o}d$ şeklinde tanımlanan bir permütasyonu olsun. $d,d^{-1}$ lerin artan fonksiyonlar olduklarına dikkat edelim. $a_i=f\mathbf{o}z^{-1}(e_i)$ için $w=a_1,\mathbf{K}\,a_n$ olsun. M(w)=N olduğunu görürüz ki bu ispatın bittiğini gösterir: ve bu w=W(N) şeklinde kendini ortaya çıkarır.

d) Biz $s$ nun w nun standart permütasyonu olduğunu iddia ediyoruz. Gerçekten, $s(i)<s(j)$ , $z^{-1}(e_i)<z^{-1}(e_j)$ ye denktir (çünkü $d,d^{-1}$ artandırlar). Alfabetik sıralamanın özeliğinden

$$\{f(z^{-1}(e_i))<f(z^{-1}(e_j))\}\ veya\ \{f(z^{-1}(e_i))=f(z^{-1}(e_j))\ ve\ e_i<e_j\}$$

eşitliğine dönüşür bu ise $(a_i<a_j)\ veya\ (a_i=a_j\ \ ve\ i<j)$ olur ve bu (4.5.4) demektir. Böylece iddiamız ispatlanmış olur.

(4.5.3) ve (4.5.4) ten $z^{-1}$ in bir devri şu şekildedir; l , $1\le l\le M((u))$ ve $u=b_1\mathbf{K}\,b_k$, olmak üzere

$(p(b_1\mathbf{K}\,b_k),l),(p(b_1b_k\mathbf{K}\,b_2),l),\mathbf{K},(p(b_{k-1}\mathbf{K}\,b_1b_k),l)$.

**Not:** Kelimelerin konjugeleri aynı devri tanımlarlar. $s=d^{-1}\mathbf{o}z^{-1}\mathbf{o}d$ olduğundan, $s$ nun devri $c=(i_1,\mathbf{K},i_k)$ şeklindedir.

$$i_1=d^{-1}((p(b_k\mathbf{K}\,b_1),l),$$
$$i_2=d^{-1}((p(b_1b_k\mathbf{K}\,b_2),l),$$
$$\mathbb{M}$$
$$i_k=d^{-1}((p(b_{k-1}\mathbf{K}\,b_k),l)$$

Not: burada $a_i=f\mathbf{o}z^{-1}(e_i)=f\mathbf{o}z^{-1}\mathbf{o}d(i)$ olduğundan 4.5.3 den şunu yazarız;

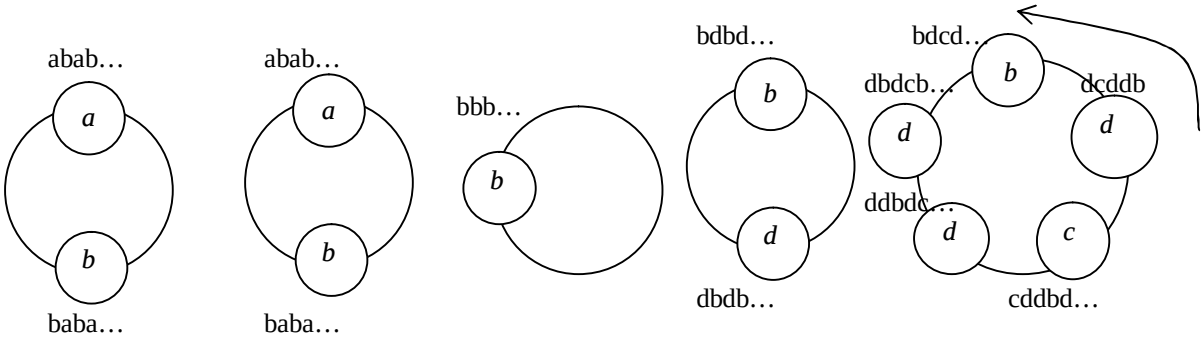
$$a_{i_1} = f \circ z^{-1}((p(b_k \mathbf{K} b_1), l)) = f((p(b_1 \mathbf{K}), l)) = b_1,$$
$$a_{i_2} = f \circ z^{-1}((p(b_1 \mathbf{K} b_2), l)) = f((p(b_2 \mathbf{K}), l)) = b_2,$$
$$\mathbb{M}$$
$$a_{i_1} = f \circ z^{-1}((p(b_{k-1} \mathbf{K} b_k), l)) = f((p(b_k \mathbf{K}), l)) = b_k.$$

Bu şunun olduğunu gösterir; $v_c(w) = (a_{i_1}, \mathbf{K}\, a_{i_k})$ kolyesi (u) ya denktir (eşittir) ve biz *M(w)=M* eşitliğinin olduğu sonucuna varırırz.

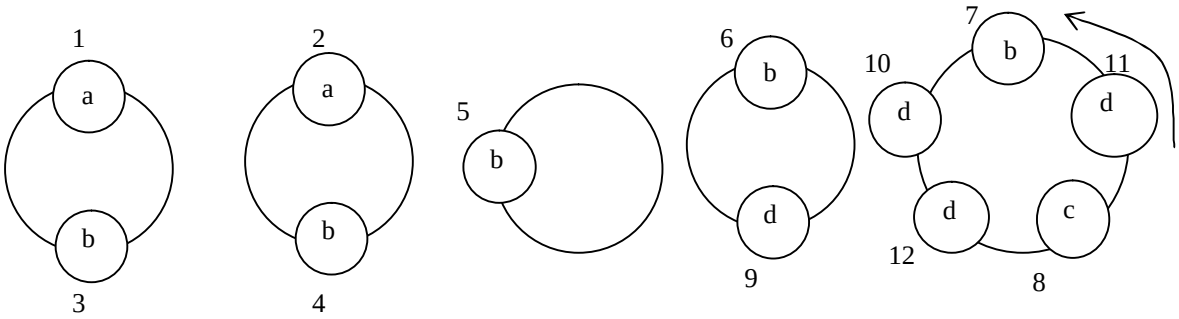
**Örnek 4.5.5**: M nin W ters fonksiyonunu, şekil 4.4 te gösterildiği gibi ilkel kolyelerin multisetleri üzerinde tanımlanmış olsun. Kolyeleri zıt yönde, her kolyenin köşesini, metin tarafından elde edilen peiyodik dizi ile etiketleyelim (şekil 4.5 e bak). Şimdi bu dizi sözlük sıralamasında olsun. (ilk kolye çeşitli sıralamalara sahiptir, çarpımsal yerleştirildiği zaman şekil 4.6 ya bak) Devir şekli içinde bir *s* permütasyonu elde edilir: onu bir kelime gibi yaz ve köşelere göre etiketler ile her bir basmağını yerleştir:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *s* = | 3 | 4 | 1 | 2 | 5 | 9 | 10 | 11 | 6 | 12 | 7 | 8 |
| w = | b | b | a | a | b | d | d | d | b | d | b | c |

örnek 4.1.8 in kelimelerinden elde edilenlerle, olması gereken




**Şekil 4-5**




**Şekil 4-6**

# 5. SABİT UZUNLUKLU İPTAL EDİLEMEZ KÜMELER

Bu kısımda Lyndon kelimelerini daha değişik bir yolla algoritmik olarak veren bir teknikten söz edeceğiz. Bir önceki bölümde Lyndon kelimeleri ayrıntılı olarak incelenmişti. Şimd bu kısımda Lyndon kelimelerini algoritmik olarak veren bir teknikten bahsedeceğiz. Bu da J. M. Champarnaud, G. Hansel ve D. Perin (2003) tarafından verilen sabit uzunluklu kısaltılamaz kelimeler ile belirlenecektir.

## 5.1. Ön bilgiler

$A, <$ ile lineer sıralı bir sonlu küme olsun. $A$ alfabesinde bir kelime, $A$ nın elemanlarının sonlu bir dizisidir. Kapsamadan dolayı boş diziye boş kelime denir. $A$ alfabesi üzerindeki tüm kelimelerin kümesini $A^*$ ile gösterelim. $w \in A^*$ kelimesinin uzunluğunu $|w|$ ile gösterelim. Eğer bir w kelimesi için $w = pu$ olacak şekilde bir $u$ kelimesi varsa $p$ ye $w$ nun ön eki denir. Eğer $p \neq w$ ise $p$ ye öz önek denir. Benzer şekilde simetrik olarak bir son ek tanımlanır. Bir $w$ kelimesi için $w = pxq$ olacak şekilde $p$ ve $q$ kelimeleri varsa $x$ e $w$ nun bir faktörü (çarpanı, parçası) denir. Bunları aynı zamanda sonsuz kelimeler içinde kullanacağız. $A$ üzerinde iki yanı sonsuz kelime, $(a_n)_{n\in\mathbf{Z}}$ dizisidir. $k = |x|$ olmak üzere eğer $x = a_n a_{n+1} \mathbf{L}\, a_{n+k-1}$ olacak şekilde bir $n \in \mathbf{Z}$ indisi var ise $x$ e iki tarafı sonsuz bir kelimenin bir faktörü denir. Eğer her $n \in \mathbf{Z}$ için $a_{n+p} = a_n$ olacak şekilde bir $p \geq 1$ sayısı varsa her iki tarafı sonsuz olan $(a_n)_{n\in\mathbf{Z}}$ dizisine periyodik dizi denir. Bu nedenle, eğer iki tarafı sonsuz olan kelime sonlu bir $u$ kelimesinin tekrarlanmasından elde edilmişse bu kelimeye periyodik kelime denir. Bu şekildeki periyodik kelimelerin kümesini $u^z$ ile göstereceğiz.

Benzer kavramlar tek tarafı sonsuz kelimeler içinde geçerlidir. Bunlar için diziler $(a_n)_{n\in\mathbf{N}}$ dir. $x, y \in A^*$ ve $y$ boş olmayan kelime olmak üzere $xyyy...$ sonsuz kelimeyi $xyw$ ile gösteririz $xyw = xyyy...$ dir.

Bir $w$ kelimesinin bir hem öneki hem son eki olan kelimeye ve boş olmayan kelimeye $w$ nun sınırı (border) denir. Sınırı sadece kendisi olan bir kelimeye sınırsız (unbordered) denir.

$A$ alfabesi üzerindeki tüm kelimelerin kümesi $A^*$ üzerinde $<$ sıralaması uygulanarak alfabetik düzen ile lineer olarak sıralıdır. Tanımdan dolayı $a,b\in A$ ve $a<b$, $u,v,w\in A^*$ olmak üzere eğer; $x$, $y$ nin bir öz öneki (yani $y=xs$, $s\in A^*$) veya eğer $x=uav$, $y=ubw$ ise her iki durumda da $x<y$ dir. Alfabetik sıranın basit bir özelliği olarak şunu söyleyebiliriz; $x<y$ ise ve eğer $x$, $y$ nin bir öneki değilse her $u$, $v$ kelimesi için $xu<yv$ dir. (yani $ali<ayşe$ ise $alibey<ayşehanım$ sözlük sıralıdır)

Eğer $x=uv$ ve $y=vu$ olacak şekilde $u,v$ kelimeleri varsa $x$ ile $y$ kelimeleri konjugedirler denir( örnek: *araba* ile *abara* konjugedirler). Konjugelik $A^*$ da bir denklik bağıntısıdır. Eğer bir kelime bir öz kuvveti yok ise primitiftir (ilkeldir) denir.(örnek: "*bekir*" primitiftir ama "*baba*" primitif değildir çünkü $baba=(ba)2$ dir) Yani $r\in A^*$ için $n>1$ olmak üzere $r^n$ şeklinde olmayan kelimeler primitiftir. $q$ sembollü, $k$ uzunluklu primitif kelimelerin konjuge sınıflarının $p(k,q)$ sayısı witt'in ünlü formülü ile verilmiştir.

$$p(k,q)=\frac{1}{k}\sum_{d|k} m\left(k/d\right)q^d$$

primitif kelimelerin konjuge sınıflarının sayısını gösterir. Burada $m$, Möbius fonksiyonudur.

$c(k,q)$: q harfli k uzunluklu kelimelerin konjuge sınıflarının sayısı olmak üzere;

$$c(k,q)=\frac{1}{k}\sum_{d|k} j\left(k/d\right)q^d$$

konjuge sınıflarının sayısı. Burada $j$ , Eüler fonksiyonudur.

$k\le 13$ ve $q=2$ için $p(k,q)$ ve $c(k,q)$ değerleri alttaki tabloda verilmiştir.

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p(k,2)$ | 2 | 1 | 2 | 3 | 6 | 9 | 18 | 30 | 56 | 99 | 186 | 335 | 630 |
| $c(k,2)$ | 2 | 3 | 4 | 6 | 8 | 14 | 20 | 36 | 60 | 108 | 188 | 352 | 632 |

Gerçekten de: $\{a,b\}$ için $a<b$ olsun. 5 uzunluklu kelimeler aşağıda verilmiştir.

| $k$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $p(k,2)$ | $a,b$ | $ab$ | $aab,abb$ | $aaab,aabb,$ $abbb$ | $aaaab,aaabb,aabab,$ $aabbb,abaab,abbbb$ |
| $c(k,2)$ | $a,b$ | $aa,ab,bb$ | $aaa,aab,$ $abb,bbb$ | $aaaa,aaab,$ $aabb,abab,$ $abbb,bbbb$ | $a^5,aaaab,aaabb,aabab$ $abaab,ababb,abbbb,b^5$ |

Eğer bir kelime kendisinin konjuge sınıfındaki en küçük kelimesi ise bu kelimeye minimaldir denir. Minimal kelimelerin kümesini $M$ ile gösterelim. Minimal kelimelerin öneklerinin kümesini $P$ ile gösterelim. Bir Lyndon kelimesi hem primitif hem minimal olan kelimedir. Lyndon kelimelerinin kümesini $L$ ile gösterelim.

Aşağıdaki önermede, sonuçta kullanılan, Lyndon kelimelerinin basit ve ünlü özellikleri verilmiştir. Tamamlama adına bir de ispat ekliyoruz.

Önce Lyndon kelimelerinin tanımına eşdeğer olanı veriyoruz. Bu özellikle Lyndon kelimelerinin unbordered (sınırsız, kenarsız) olduğunu gösterir.

**Önerme 5.1.1:** Boş olmayan her $w$ kelimesi için aşağıdaki şartlar denktir.

*(i)* $w$ bir Lyndon kelimesidir.

*(ii)* Boş olmayan herhangi $u,v$ kelimeleri için

$w=uv$ den $w<vu$ elde ederiz.

*(iii)* $w$ kendisinin boş olmayan bir öz son ekinden kesinlikle daha küçüktür.

**İspat:** $u, v$ boş değilse ve $uv$ primitifse $uv \neq vu$ olduğu iyi bilinir. O halde $(i) \Rightarrow (ii)$ dir.

$(ii) \Rightarrow (iii)$. $v$, ve $q$ her ikisi de boş değilse $w = vq$ olsun $q$, $w$ nun bir öneki olması mümkün değildir. Gerçekten de diğer durumda boş olmayan bir $u$ için $w = qu$ olurdu. $vq = qu < uq$ olduğundan $vu$ elde ederiz böylece $qv < qu = vq$ olur. Bu ise bir çelişkidir.

$q$, $w$ nun bir öneki değilse buradan $w < qv$ den $w < q$ dur.

$(iii) \Rightarrow (i)$. $w$ nın primitif olduğu açıktır. O halde $w$ bir Lyndon kelimedir. □

Bir sonraki sonuç açıktır. Çünkü her kelimenin, bir primitif kelimenin kuvveti olarak yazmanın bir tek yolu vardır.

**Önerme 5.1.2:** Bir kelime minimaldir ancak ve ancak bu kelime bir Lyndon kelimesinin bir kuvvetidir. Bu, Lyndon kelimesi tek türlü belirlidir.

Bir sonraki sonuç bunun doğruluğunu ispatlar. Onun zıttı da doğrudur.

**Önerme 5.1.3:** $w$, bir minimal kelimenin öneki olsun. O zaman $w$ nun her son eki ya $w$ nun bir ön ekidir ya da $w$ dan daha büyüktür. Buna denk olarak, $w$ nun her öneki, $w$ nun aynı uzunluklu son ekinden küçük ya da eşittir.

$n \geq 1$ ve $p$, $x$ in bir öz öneki olmak üzere bir $w \in A^*$ kelimesi $w = xnp$ şeklinde ise $w$ ya $x$ in bir sesquipoweri denir. (örnek: $ahmetahmetah = (ahmet)^2 ah$ den dolayı "$ahmetahmetah$" kelimesi "$ahmet$" kelimesinin sesquipoweridir, fakat "$ahmetahmetak$" kelimesi sesquipower değildir.) Özellikle Lyndon kelimelerinin sesguipowerları ilginç olacaktır.

Örneğin: $a < b$ olmak üzere $w = aabbaabbaa$, $l = aabb$ Lyndon kelimesinin bir sesquipoweri dir.

Aşağıdaki özellik, Fredericksen ve Maiorana' (1978) de ve to Duval (1983) de verilenlerden tam bağımsız olarak uygun Lyndon kelimeleri üreten bir metoda bağlar. Gerçekten de $P$ nin elemanlarını alfabetik sıralı olarak üretmek kolaydır. Bu aynı zamanda $L$ nin veya $M$ nin elemanlarını üretmek için bir metot verir. Bu

üretme problemi birkaç konu bağlamında düşünülmüştür. (Knuth D.E. (2002), Moreno E. (2003), Reutenauer C. (1993) ).

**Önerme 5.1.4:** Her $w \in A^*$ kelimesi içi aşağıdaki şartlar bir birine denktir.

*(i)* $w$, bir minimal kelimenin boş olmayan bir önekidir.

*(ii)* $w$, bir Lyndon kelimesinin $b, r$ sesquipoweridir

**İspat:** (iii) $\Rightarrow$ (i) olduğu açıktır. Bunun tersini gösterelim. Hipotez gereği öyle bir $s$ Lyndon kelimesi vardır ki $w$, $sn$ nin bir önekidir. Eğer $|w| \geq |s|$ ise $w$, $s$ nin bir sesquipoweri olduğu açıktır. Bu nedenle $w$ yu $s$ nin bir boş olmayan bir öneki olması durumunu düşünmek yeterli olur.

$w$ nun uzunluğu üzerinde bir tümevarım kullanalım.

Eğer; $|w| = 1$ ise $w$ bir Lyndon kelimesidir. O halde özellik doğru olur.

$|w| > 1$ ise $a \in A$ olmak üzere $w = va$ olduğunu farz edelim. Tümevarım hipotezinden $n \geq 1$ olmak üzere $l$ bir Lyndon kelimesi olduğundan $v = lnp$ parçalanışını elde ederiz ve $p$, l in bir öz önekidir.

$b \in A$ ve $x \in A^*$ olmak üzere $l = pbx$ olsun. Şunları göstereceğiz; ya $a = b$ dir ki bu durumda $w = va$, $l$ nin bir sesquipowerıdır ya da onun kendisi bir Lyndon kelimesidir.

Gerçekten de $w \in P$ olduğunda Önerme 5.1.3 ü elde ederiz. $pb \leq pa$ olur ve bu durumda b ≤ a dır. Buradan da b<a ise $w$ bir Lyndon kelimesi olduğunu ispatlarız. Bunun için $s = ta$, $w = lnpa$ nın bir öz öneki olsun. $u = tb$, $z = lnpb$ nin bir öz öneki olan bir kelimedir. Öncelikle $z \in P$ olduğundan $u$ kelimesi $z$ nin ve aynı zamanda $w$ nun da aynı uzunluklu ön ekinden büyük veya eşittir. Fakat $s > u$ olduğundan $s$ kelimesi $w$ dan daha büyüktür. Ve böylece $w$ bir Lyndon kelimesidir. □

**İspat:** $P$ üzerinden yapıldı ki gerçekten $P$ Lyndon kelimelerinin ön eklerinin kümesine eşittir.

Bir $w$ kelimesinin bir bölümü (division) bir *(ln,u)* çiftidir öyle ki $l \in L$, $n \geq 1$ ve $u \in A^*$ için $|u| < |l|$ olmak üzere $w = lnu$ dur.

Önerme 5.1.4. e göre $P$ deki her kelime en az bir bölmeye izin verir. Şunu söyleyebiliriz. Eğer $w$ nun *(ln,u)* şeklinde bir bölümü varsa $w$ kelimesine bir $l$ Lyndon kelimesi karşılık gelir. Her $l \in L$ için $w$ nin birden fazla bölümü olduğu açıktır. $w$ ya karşılık gelen en kısa $l$ Lyndon kelimesi olmak üzere *(ln,u)* ye $w \in P$ nin ana bölümü denir. $ln$ kelimesi $w$ nun ana parçasıdır (principle part) ve p(w) ile gösterilir. $u$ ya $w$ nun dayanağı (rest) denir ve $r(w)$ ile gösterilir.

Örneğin: $a < b$ olmak üzere $aabaabbba$ kelimesi $(aabaabbb, a)$ ve $(aabaabb, ba)$ şeklinde iki türlü bölmeye izin verir. İlkine göre parçalanma bir Lyndon kelimesinin sesquipoweri gibidir. İkincisi ise ana bölmedir. ( Açıklama yaparsak; Neden ilk bölmeyi almadık? dersek çünkü ikinci bölmedeki $l$ Lyndon kelimesi daha küçüktür $(aab < aabaabb)$. Peki niçin $(aabaab, bba) = ((aab)^2, bba)$ şeklinde bölmedik, çünkü ana parçayı oluşturan $l = aab$ Lyndon kelimesinin uzunluğu, dayanak olan $u = bba$ dan büyük olmalı. Bu şekilde bölersek eşit oluyor. Onun için böyle bölemeyiz. )

## 5.2. İptal Edilemez kümeler

$A$ bir alfabe olsun. $I \subset A^*$ kümesi olmak üzere, $A$ alfabesi üzerindeki $(a_n)_{n \in \mathbf{Z}}$ iki tarafı sonsuz kelimelerin en az bir faktörü $I$ kümesi içinde bulunuyorsa $I$ ya $A$ üzerinde iptal edilemez küme denir. Bununla, herhangi bir tarafı sonsuz olan kelimelerin $I$ daki faktörünü sormak da aynı şeydir. Eğer alfabe sonlu olursa iptal edilen kelimeler de sonlu olacaktır. O zaman $I$ da sonlu olur.

**Örnek 5.2.1:** $A = \{a, b\}$ olmak üzere $U = \{a, b^{10}\}$ kümesi, uzunluğu 10 olan her kelime için ya bir harfi $a$ olacaktır, ya da $b10$ olacaktır. Dolayısı ile $U$ kümesi $A$ üzerinde iptal edilemez bir kümedir. Ters bir örnek verirsek; $V = \{aa, b^{10}\}$ kümesi

iptal edilebilir. Çünkü $(ab)w = abababababab...$ sonsuz kelimesinin bir faktörü $V$ kümesinde yoktur.

Bu nedenle uzunluğu $k$ olan kelimelerden elde edilen iptal edilemez kümeler bizim için ilginç olacaktır. Aşağıdaki önermeyi ispatlamak kolaydır.

**Önerme 5.2.2:** $A$ sonlu bir alfabe ve $I$, $A$ üzerindeki k uzunluklu kelimelerin iptal edilemez bir kümesi olsun. $I$ nın kardinalitesi (eleman sayısı), $A$ alfabesi üzerindeki k uzunluklu kelimelerin konjuge sınıflarının sayısından küçük ya da eşittir.

**İspat:** $u \in A^*$, $k$ uzunluklu bir kelime olsun. $uw$ (sonsuz) kelimesinin $k$ uzunluklu faktörleri $u$ nun konjuge sınıfının elemanlarıdır. Bu nedenle $I$ bu sınıftan en az bir eleman içermek zorundadır. □

$k \geq 1$ bir tam sayı ve Mk, k uzunluklu minimal kelimelerin kümesi olsun. Her $m \in Mk$ için $p(m)$ onun (principle part) ana parçası ve $r(m)$ onun dayanağı olsun. $Ik$ kümesi şöyledir;

$$I_k = \{r(m)p(m) \ : \ m \in M_k\}$$

$Ik$ da bulunan ilkel olmayan herhangi bir minimal kelimenin olduğunu söyleriz.

**Örnek 5.2.3:** Tablo 5.3.1 de $M_7$ ve $I_7$ verilmiştir. Aşağıdaki nesneler gösterir ki $I_7$ iptal edilemez bir kümedir. Önerme 5.2.2 den dolayı $I_k$ nın elemanlarının sayısı, $k$ uzunluklu kelimelerin iptal edilemez bir takım elemanlarının mümkün olan en az sayısıdır.

$I_k = \{r(m)p(m) \ : \ m \in M_k\}$

$M_k$ : $k$ uzunluklu minimal kelimelerin kümesi

$p(m)$ : $M_k$ nın $m$ elemanı için ana parça (principle part)

$r(m)$ : $M_k$ nın $m$ elemanı için dayanak (rest)

$I_k$ : $k$ uzunluklu iptal edilemez kelimelerin kümesi

$I_7$ için şunu söyleyebiliriz; her iki yana sonsuz devam eden ve 7 uzunluklu bir kelimenin tekrarı olarak yazılan bir sonsuz kelimenin içinde çarpan (faktör) olarak mutlaka $I_7$ nin bir elemanı vardır.

Burada şuna dikkat edelim: $m=(ln,u)$ deki $ln$ bulurken $l$ yi tespit ederken uzunluğunun $u$ nunkinden daha büyük olmasına dikkat etmeliyiz.

Ayrıca daha önce de bilindiği üzere $c(7,2)=20$ olduğu burada da görülür.

| | $M_7$ nin elemanı $=m$ | $m=(ln,u)$ | $p(m)=ln$ | $r(m)=u$ | $I_7 = r(m)p(m)$ |
|---|---|---|---|---|---|
| 1 | aaaaaaa | $(a^7,1)$ | aaaaaaa | 1 | aaaaaaa |
| 2 | aaaaaab | (aaaaaab,1) | aaaaaab | 1 | aaaaaab |
| 3 | aaaaabb | (aaaaab,b) | aaaaab | b | baaaaab |
| 4 | aaaabab | (aaaab,ab) | aaaab | ab | abaaaab |
| 5 | aaaabbb | (aaaab,bb) | aaaab | bb | bbaaaab |
| 6 | aaabaab | (aaab,aab) | aaab | aab | aabaaab |
| 7 | aaababb | (aaab,abb) | aaab | abb | abbaaab |
| 8 | aaabbab | (aaab,bab) | aaab | bab | babaaab |
| 9 | aaabbbb | (aaab,bbb) | aaab | bbb | bbbaaab |
| 10 | aabaabb | $((aab)^2,b)$ | aabaab | b | baabaab |
| 11 | aababab | (aabab,ab) | aabab | ab | abaabab |
| 12 | aababbb | (aabab,bb) | aabab | bb | bbaabab |
| 13 | aabbabb | (aabb,abb) | aabb | abb | abbaabb |
| 14 | aabbbab | (aabb,bab) | aabb | bab | babaabb |
| 15 | aabbbbb | (aabb,bbb) | aabb | bbb | bbbaabb |
| 16 | abababb | ((ab)3,b) | ababab | b | bababab |
| 17 | ababbbb | (ababb,bb) | ababb | bb | bbababb |
| 18 | abbabbb | ((abb)2,b) | abbabb | b | babbabb |
| 19 | abbbbbb | (abbb,bbb) | abbb | bbb | bbbabbb |
| 20 | bbbbbbb | $(b^7,1)$ | bbbbbbb | 1 | bbbbbbb |

Tablo 5.2.1

## 5.3. Ana sonuç

Aşağıdaki sonucu ispatladığımızda göreceğiz ki k uzunluklu kelimelerin iptal edilemez kümelerinin boyutu (size) üzerinden $c(k,q)$ alt sınırını (lower bound) her $k,q \geq 1$ için uzatırız. Bu zaten J. Mykkelveit (1972) tarafından giriş olarak söylenmişti.

**Teorem 5.3.1:** $k,q \geq 1$ için $q$ sembollü $k$ uzunluklu $c(k,q)$ formunda bir iptal edilemez küme vardır.

Bu teorem aşağıdaki teoremin bir sonucu olacak ve minimal iptal edilemez kümelerin yapısını verir.

**Teorem 5.3.2:** $A$ sonlu bir alfabe olsun ve $k \geq 1$ olsun, $M_k$ $A$ alfabesi üzerindeki $k$ uzunluklu kelimelerin denklik sınıflarının içindeki minimal olanlarının kümesi olsun. Her $m \in Mk$ için $p(m)$ $m$ nin ana parçası ve $r(m)$ onun dayanağı olmak üzere

$$I_k = \{r(m)p(m) \ : \ m \in M_k\}$$

kümesi iptal edilemez kümedir.

Teorem 5.3.2 nin ispatı için bazı ön sonuçlara ihtiyacımız var. Bunların ilki iptal edilemez kümelerin denkliğinin temel tanımıdır.

**Önerme 5.3.3:** $I \subset A^*$ sonlu kelimelerin bir kümesi olsun. Aşağıdaki şartlar birbirine denktir.

*(i)* $I$ kümesi iptal edilemezdir.

*(ii)* İki tarafı sonsuz her periyodik kelimenin, $I$ içinde bulunan en az bir çarpanı vardır.

**İspat:** (ii) $\Rightarrow$ (i) i göstermek yeterlidir. $(a_n)_{n \in \mathbf{Z}}$ iki tarafı sonsuz bir harf dizisi olsun. $u \in A^*$ kelimesi $I$ daki her kelimeden daha büyük olsun ve $(a_n)_{n \in \mathbf{Z}}$ dizisindeki

durumlarının sayısı sonsuzdur. Bu dizi $uvu$ şeklinde en az bir çarpana sahiptir. Hipotez gereği $...uvuvuvuv...$ sonsuz periyodik kelimesi bir $w \in I$ çarpanına sahiptir. $w$ kelimesi $uv$ ve $vu$ kelimelerinden en az birinin bir çarpanıdır. Bu aynı zamanda $(a_n)_{n\in \mathbf{Z}}$ dizisinin bir faktörüdür. Ve bu durumda $I$ iptal edilemezdir. □

**Önerme5.3.4:** $l$ , $l$ nin öneki olacak şekilde iki Lyndon kelimesi olsunlar. $|s| < |l|$ olacak şekilde $l$ nin bir son eki $s \in A^*$ olsun. O zaman her $n > 0$ için $w = l^n s$ kelimesi bir Lyndon kelimesidir.

**İspat:** $w$ nun bir öz soneki $t$ olsun. Üç durumla karşılaşabiliriz.

*1)* $|t| < |s|$ olabilir o zaman, $t$, $l$ Lyndon kelimesinin bir öz sonekidir. Bu durumda $t > 1 > l$ ve $|t| < |l|$ olduğundan $t > l^n s = w$ dir.

*2)* $|t| > |s|$ ve $0 \le i < n$ olmak üzere $t = l^i s$ gibi $t$ yi parçalayabiliriz. $s$ , $l$ nin öz soneki olduğundan $s > 1 \ge l$ yı elde ederiz. Sonuç olarak $t = l^i s > l^i 1$ dir. Çünkü $|s| < |l|$ dir ve buradan $t > l^n s = w$ yu elde ederiz.

*3)* $|t| > |s|$ ve $t$ nin parçalanışı $t = s't'$ , $s'$ , $l$ nın son öz ekidir $l \in L$ olduğundan $s' > l$ dir. Ve $t = s't' > l^n s = w$ olur.

Her durumda da $t > w$ dir. Ve böylece $w$ bir Lyndon kelimesidir.

# 6. LİE CEBİRLERİNİN SONLU KOBOYUTLU ALT CEBİRLERİ VE KOBOYUTU HESAPLAMA ALGORİTMASI

Bu bölümde $L$ Lie cebirinin sonlu koboyutlu alt cebirleri ele alınacak ve koboyutu, verilen bir $c$ tam sayısı olan sonlu sayıda alt cebirin varlığı incelenecektir.

$L$, $k$ cismi üzerinde doğurucular kümesi $X$ olsun. $B$, $L$ nin bir alt cebiri olsun. $L$ yi bir $L^*$ serbest Lie cebirinin bölüm cebiri olarak düşünebiliriz. Buna göre $B^*$, $B$ nin $L^*$ içindeki ters görüntüsü olmak üzere $L^*$ içinde modülo $B^*$ bazının seçimi, $L$ içinde modülo$B$ bazının seçimi ile aynıdır. Bu nedenle, $L^*$ serbest Lie cebirinin bir modülo $B^*$ bazının nasıl bulunacağını Ela AYDIN (1997) göstermiş ve bir de algoritma vermiştir.

Lie cebirinin sonlu bir doğurucu küme ve tanımlayıcı bağıntılarla takdimi, matematiksel ve algoritmik çalışmalarda önemli bir yer tutar. Takdim problemi fizikten matematiğe kadar yayılan geniş bir alanda pratikliği açısından büyük bir önem taşır. Özellikle sonlu takdimler bir çok fiziksel modellerde karşımıza çıkar. Doğurucu elemanların ve bağıntıların sayısının fazla olduğu durumlarda bilgisayar cebirine ihtiyaç duyulur. Sonlu takdimli Lie cebirinin bazının bulunuşu için bir algoritma C dilindeki programı P.Gerdt tarafından yazılmıştır. (Gerdt ve Kornyak;1996).

Bu bölümde sonlu takdimli Lie cebirlerinin sonlu doğrulmuş alt cebirlerine göre modülo alt cebir bazının inşası için verilen yönteme uygun bir algoritma yazılmıştır. Ayrıca sonlu takdimli Lie cebirlerinin bazı sınıfları ve bunların özel alt cebirleri seçilerek bu alt cebire göre modülo baz seçimi yapılmıştır.

$F$, $k$, cismi üzerinde $\{x_1, x_2, \mathbf{L}, x_n\}$ kümesi tarafından doğrulan serbest Lie cebiri olsun. $F$ yi kısaca $F = \langle x_1, x_2, \mathbf{L}, x_n \rangle$ şeklinde belirtelim. Eğer $F$, $k$ üzerinde herhangi bir Lie cebiri ise $I$, $r_1 = (xx)$, $r_2 = (xy)z + (yz)x + (zx)y$ formundaki elemanlar tarafından doğurulan ideal olmak üzere $L$ yi

$$L = F/I = \langle x_1, x_2, \mathbf{L}, x_n \,|\, r_1 = r_2 = 0 \rangle$$

şeklinde ifade edebiliriz. Bu nedenle serbest :Lie cebirleri için söyleyeceğiz.

Şimdi Algoritmayı inceleyelim:

$$L = \langle x_1, x_2, \mathbf{L}, x_k \,|\, r_1 = r_2 = \mathbf{L} = r_m = 0 \rangle$$

Lie cebirini düşünelim. *B, L* nin aşağıdaki takdime sahip bir alt cebiri olsun:

$$B = \langle y_1, y_2, \mathbf{L}, y_t \,|\, s_1 = s_2 = \mathbf{L} = s_n = 0 \rangle$$

Aşağıda vereceğimiz algoritmayı işleterek *L* nin modülo*B* maksimal lineer bağımsız elemanlarının kümesini vereceğiz.

Bu tezin yedinci bölümünde algoritmayı çalıştıran bir bilgisayar programı verilmiştir.

### 6.1. Modülo baz için Algoritma

*L Lie* cebiri ve *B, L* nin alt cebiri olsun. Algoritmayı başlatmak için girilmesi gereken verileri aşağıdaki gibi sıralayabiliriz:

*i)* *L* nin doğurucularının indislere göre sıralanmış $X = \{x_1, x_2, \mathbf{L}\}$ kümesi.

*ii)* *B* nin doğurucularının indislere göre sıralanmış $h = \{y_1, y_2, \mathbf{L}\}$ kümesi.

*iii)* *L* nin tanımlayıcı bağıntılarının $R = \{r_1, r_2, \mathbf{L}\}$ kümesi.

*iv)* *B* nin varsa tanımlayıcı bağıntılarının $S = \{s_1, s_2, \mathbf{L}\}$ kümesi.

*v)* Eğer bağıntılarda yer alıyorsa skaler parametrelerin $P = \{p_1, p_2, \mathbf{L}\}$

Bu veriler girilip az sonra nasıl işletileceğini göstereceğimiz algoritmanın sonucunda *L* nin modülo*B* lineer bağımsız elemanlarının $\{b_1, b_2, \mathbf{L}\}$ kümesini elde edeceğiz.

Şimdi adım adım bu algoritmanın girilen veriler ile nasıl işlediğini görelim:

**Adım1:** $h, B$ nin doğurucularının bir kümesi olmak üzere $h$ nin derecesi *d* olan elemanlarının $h_d$ kümesinin belirlenmesi.

$h$ kümesini elemanlarının derecelerine göre parçalayalım. Bu adımda her bir elemanın üzereinden tek tek gidilerek uzunluklarına göre sınıflandırma işlemi yapılır. $h$ kümesine göre uzunluğu *k* olan tüm elemanların kümesi,

$$h = \left\{ y_i^l \,\middle|\, l(y_i^l) = k, i \in I_k, k = 1,2,3,\mathbf{L} \right\}$$

ile gösterelim. Burada $I_k$ bir indis kümesidir. Bulduğumuz birinci dereceden elemanları yine bulunuş sırasına göre ;

$$y_1^1, y_2^1, \mathbf{L}, y_k^1$$

ile , ikinci dereceden elemanları yine bulunuş sırasına göre

$$y_1^2, y_2^2, \mathbf{L}, y_s^2$$

olarak isimlendirelim. Bu şekilde devam edilirse $h_1, h_2, \mathbf{L}$ kümeleri belirlenmiş olur.

**Adım2:** Eğer $B$ nin doğurucularının $h$ kümesi indirgenmiş değilse, $h$ nın indirgenmesi. Küme indirgenmiş ise 2.Adım sonucunda $h$ da bir değişiklik olmaz.

Bu adımda öncelikle $h_0 = \varnothing$ koyalım ve $h_1, h_2, \mathbf{L}, h_i$ kümelerinin inşa edilmiş olduğunu varsayalım.

$E_i$, $\bigcup_{j=0}^{i} h_j$ kümesi tarafından doğrulan alt cebir;

$T_i$, $B$ alt cebirinin derecesi $d \le i+1$ olan elemanların oluşturduğu vektör uayı;

$T_i'$, derecesi $i+1$ den küçük veya eşit olan elemanların gerdiği $T$ uzayının bir alt uzayı olsun.

$T_i$ vektör uzayında modülo $T_i'$ maksimal lineer bağımsız $h_{i+1}$ kümesini seçebiliriz. Eğer $T_i = T_i'$ ise $h_{i+1} = \varnothing$ dir. Böylece $h = \bigcup_{j=0}^{\infty} h_j$ kümesi indirgenmiş küme olur.

**Adım3:** $U_N$, derecesi $N$ den küçük veya eşit olan elemanların kümesi iken $B$ nin $U_N$ tarafından gerilen $B_N$ alt uzayının belirlenmesi.

1.Adımdaki uzunluklarına göre sıralanmış elemanlardan derecesi $N$ den küçük veya eşit olanların sınıfı alınarak bunların gerdiği uzay bulunur. Bu durumda;

$$B_N = Sp(\{h_0 \cup h_1 \cup \mathbf{L} \cup h_{N-1}\})$$

dir.

**Adım4:** Son adımda $S_0 = \varnothing$ alalım ve $i<k$ için $S_i$ kümelerini inşa etmiş olalım. $S_k$ ile $Sp\left(\left\{B_k \cup \bigcup_{i<k} S_i\right\}\right)$ maksimal lineer bağımsız *k*- yıncı dereceden regüler kelimelerin kümesini gösterelim. Bu durumda $S = \bigcup_{i=0}^{\infty} S_i$ kümesi olarak alıp $L$ nin modülo$B$ bazını belirleyelim. Bu adımda $L$ nin $B_k \cup \bigcup_{i<k} S_i$ kümeleri tarafından gerilen uzayın elemanlarından farklı olan *k* uzunluklu elemanlarını tek tek alarak $S_k$ kümesine koyalım ve bu işleme *k*'yı bir artırarak devam edelim. Eğer $S_i = \varnothing$ oluyorsa, algoritmaya son verelim. Bu durumda modülo baz kümesi sonludur. Eğer hiç bir $S_i = \varnothing$ olmuyorsa baz kümesi sonsuz elemanlıdır.

Sonuç olarak, tüm $S_i$ kümelerindeki $b_i$ elemanları bize $L$ nin modülo$B$ baz kümesini verecektir. Şimdi bu algoritmanın işleyişini özel takdime sahip olan bazı Lie cebirleri üzerinde görelim. Aşağıdaki örnekleri bölüm 7.3.4 de verilen programla çalıştırıp sonuçlarını görebilirsiniz.

Bu çalışmada $h$ nin indirgenmiş olduğu durumlar göz önüne alınmıştır. Ayrıca adım2 deki indirgeme algoritması yapılmamıştır.

Aşağıdaki örnekler Ela AYDIN'nın (1997) Doktora tezinden alınmıştır. Tezden farklı olarak biz burada Braket çarpımının yazılışını sağa yaslı olarak aldık

Tezin son bölümünde bu algoritmanın bilgisayar programı yapılacaktır.

**Örnek 6.1.1**

$L = \langle x, y, z \mid [y,[y,z]] = [x,[x,[x,y]]] = [x,[x,z]] = 0 \rangle$ Lie cebirinin bir alt cebiri $B = \langle y\ ,\ z\ ,\ [x,y]\ ,\ [x,[x,y]]\ ,\ [y,[x,z]]\ ,\ [z,[x,z]] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x,\ [x,z]\}$ ve $B$ nin $L$ içindeki koboyutu 2 dir.

**Örnek 6.1.2**

$L = \langle x, y \mid [x,[x,y]] = [y,[y,[x,y]]] = 0 \rangle$ Lie cebirinin bir alt cebiri B= $\langle y\ ,\ [x,y] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x\}$ ve $B$ nin $L$ içindeki koboyutu 1 dir.

**Örnek 6.1.3**

$L =< x, y \mid [x,[x,[x,[x,y]]]] = [y,[y,[y,[y,[x,y]]]]] = [y,[y,[y,[y,[x,[x,y]]]]]] = [y,[y,[y,[y,[x,[x,[x,y]]]]]]] = 0 >$

Lie cebirinin bir alt cebiri $B = \langle y\ ,\ [x,y] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x\}$ ve $B$ nin $L$ içindeki koboyutu 1 dir.

**Örnek 6.1.4**

$L = \langle x, y \mid [x,[x,[x,y]]] = 0 \rangle$ Lie cebirinin bir alt cebiri $B = \langle y\ ,\ [x,y]\ ,\ [x,[x,y]] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x\}$ ve $B$ nin $L$ içindeki koboyutu 1 dir.>

**Örnek 6.1.5**

$L = \langle x, y \mid [x,[x,[x,y]]] = [y,[y,[y,[x,y]]]] = [y,[y,[y,[x,[x,y]]]]] = 0 \rangle$

Lie cebirinin bir alt cebiri $B = \langle y\ ,\ [x,y],[y,[x,[x,y]]] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x,\ [x,[x,y]]\}$ ve $B$ nin $L$ içindeki koboyutu 2 dir.

**Örnek 6.1.6**

$L(2,3) = \langle x, y \mid [y,[y,[y,[x,[x,y]]]]] = 0 \rangle$ Lie cebirinin bir alt cebiri $B = \langle [y,[x,[x,y]]],\ [y,[y,[x,[x,y]]]] \rangle$ olsun. $L$ nin modülo$B$ bazı $\{x, y,\ [x,y],[x,[x,y]],[x,[x,[x,y]]],[y,[y,[x,y]]], \mathbf{L}\}$ ve $B$ nin $L$ içindeki koboyutu sonsuzdur.

**Örnek 6.1.7**

$L = \langle x, yz, t \mid [x,y] = [x,[y,t]] = 0 \rangle$ Lie cebirinin bir alt cebir $B =< t,[x,t] >$ olsun. $L$ nin modülo$B$ bazı $\{x,\ y,\ z,\ [x,z],[y,z],[y,t],[z,t]\}$ ve $B$ nin $L$ içindeki koboyutu 7 dir.

# 7. PROGRAMLAR

## 7.1. Giriş

Bir serbest Lie cebirinin Hall bazını hesaplayan bir bilgisayar programı Munthe H.-Kaas ve Owren B tarafından verilmiş. Andary P.(1997) Lyndon kelimelerini üreten bil algoritma ile beraber bir C programı yayınlamıştır. Ve daha sonra Sawada, J. (2003) , Andary P.(1997) nin yapmış olduğu algoritmayı geliştirmiş ve hızlandırmış. Son olarak elde edilen Lyndon kelimelerine braket koyma işlemini yapan bir program yayınlamıştır.

Bu çalışmada tekrarlamalı işlemler yardımı ile, Hall kelimelerini bulan bir program yazılmıştır. Hall kelimelerini bulurken bu kelimelerin aynı anda hangi altmerkezi terimin elemanı olduğunu da tespit etmektedir. Program aynı zamanda; 4. bölümde verilen dairesel kelimelerden faydalanarak, elde ettiğimiz Hall kelimesine karşılık gelen Lyndon kelimesini de bulmaktadır. Daha sonra bir serbest $L$ Lie cebirinin üreteçleri belli olan bir $B$ alt cebiri için, $L$ nin Modülo$B$ bazını bulan program verilmiştir.

## 7.2. Hall ve Lyndon Kelimelerini bulan Delpi7 Programı

Aşağıda verdiğimiz Delphi7 programı, rankı en fazla 20 olan bir serbest Lie cebirinin Hall bazının, en fazla 20 uzunluklu kelimelerini bulmaktadır. Bu program daha önce yazılan programlardan farklı olarak alt merkezi serilerin terimlerini ve serbest üreteç kümelerini de bulmaktadır. Ayrıca Hall elemanlarına karşılık gelen Lyndon kelimelerini de bulmaktadır.

Şimdi programın çalışma mantığını anlatalım. Öncelikle bir Hall kelimesini belirlemek için o kelimenin bileşenlerini içeren tüm Hall kelimelerini bulmuş olmalıyız. Bunun için rekürsif işlem yapılır.

**Örnek 7.2.1:** [[y,z],[z,[x,y]]] kelimesi için [y,z] ve [z,[x,y]] kelimeleri bilinmelidir. [y,z] için y ve z bilinmelidir. [z,[x,y]] için z bilinmeli, [x,y] bilinmeli ve [x,y] içinde x ve y bilinmelidir.

Bunun için ilk olarak 1 uzunluklu elemanları $X$ kümesinden alır ve alınış sırasında küçükten büyüğe doğru sıralarız. 2 uzunluklu kelimeleri bulurken 1 uzunluklu elemanların kümesinin kartezyen çarpımını alır, Hall elemanı olup olmadığına bakarız. Eğer Hall elemanı özelliğini taşıyor ise kümeye ekleriz.

**Programda tanımlanmış tipler ve değişkenler 7.2.2.:** Programda tanımladığımız en önemli tip HALL_TYPE dır ve aşağıdaki gibi tanımlanmıştır.

```
HALL_TYPE = record
   index:longword;   // bu elemanın bu uzunluktaki indexi
   I1:longword;      // birinci bileşen indexi
   I2:longword;      // ikinci bileşen indexi
   L1:byte;          // birinci bileşen uzunluğu
   L2:byte;          // ikinci bileşen uzunluğu
end;
```

Bunu şu şekilde açıklayabiliriz; 3 üreteçli bir serbest Lie cebirini en fazla 4 uzunluklu elemanları aşağıdaki gibidir.

| Kelimenin bu uzunluktaki kelimeler arasındaki sırası | Kelimenin birinci bileşeninin o uzunluktaki kelimeler arasındaki sırası | Kelimenin ikinci bileşeninin o uzunluktaki kelimeler arasındaki sırası | Birinci bileşeninin uzunluğu | İkinci bileşeninin uzunluğu | Kelimenin kendisi |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | x |
| 2 | 2 | 0 | 1 | 0 | y |
| 3 | 3 | 0 | 1 | 0 | z |
| 1 | 1 | 2 | 1 | 1 | [x,y] |
| 2 | 1 | 3 | 1 | 1 | [x,z] |
| 3 | 2 | 3 | 1 | 1 | [y,z] |
| 1 | 1 | 1 | 1 | 2 | [x,[x,y]] |

| | | | | | |
|---|---|---|---|---|---|
| 2 | 1 | 2 | 1 | 2 | [x,[x,z]] |
| 3 | 2 | 1 | 1 | 2 | [y,[x,y]] |
| 4 | 2 | 2 | 1 | 2 | [y,[x,z]] |
| 5 | 2 | 3 | 1 | 2 | [y,[y,z]] |
| 6 | 3 | 1 | 1 | 2 | [z,[x,y]] |
| 7 | 3 | 2 | 1 | 2 | [z,[x,z]] |
| 8 | 3 | 3 | 1 | 2 | [z,[y,z]] |
| 1 | 1 | 1 | 1 | 3 | [x,[x,[x,y]]] |
| 2 | 1 | 2 | 1 | 3 | [x,[x,[x,z]]] |
| 3 | 1 | 1 | 1 | 3 | [y,[x,[x,y]]] |
| 4 | 2 | 2 | 1 | 3 | [y,[x,[x,z]]] |
| 5 | 2 | 3 | 1 | 3 | [y,[y,[x,y]]] |
| 6 | 2 | 4 | 1 | 3 | [y,[y,[x,z]]] |
| 7 | 2 | 5 | 1 | 3 | [y,[y,[y,z]]] |
| 8 | 3 | 1 | 1 | 3 | [z,[x,[x,y]]] |
| 9 | 3 | 2 | 1 | 3 | [z,[x,[x,z]]] |
| 10 | 3 | 3 | 1 | 3 | [z,[y,[x,y]]] |
| 11 | 3 | 4 | 1 | 3 | [z,[y,[x,z]]] |
| 12 | 3 | 5 | 1 | 3 | [z,[y,[y,z]]] |
| 13 | 3 | 6 | 1 | 3 | [z,[z,[x,y]]] |
| 14 | 3 | 7 | 1 | 3 | [z,[z,[x,z]]] |
| 15 | 3 | 8 | 1 | 3 | [z,[z,[y,z]]] |
| 16 | 1 | 2 | 2 | 2 | [[x,y],[x,z]] |
| 17 | 1 | 3 | 2 | 2 | [[x,y],[y,z]] |
| 18 | 2 | 3 | 2 | 2 | [[x,z],[y,z]] |

Tablo 7.2.1 Üç üreteçli serbest Lie cebirinin en fazla 4 uzunluğa kadar Hall elemanlarının listesi.

Programdaki en önemli değişkenimiz All_Hall aşağıdaki gibi tanımlanmıştır.

All_Hall: array [1..20,0..1000000] of HALL_TYPE;

Bu değişken içerisinde bir Hall kelimesini barındırır. Birinci bileşen kelimenin uzunluğu , ikinci bileşen ise bu uzunluktaki elemanların içindeki sıra numarasını gösterir. Örnek olarak

All_Hall[4,13]= [z,[z,[x,y]]] dir.

Yani 4 uzunluklu kelimelerin 13. kelimesi demektir.

Özel olarak n uzunluklu kaç kelimenin olduğunu kolay olarak bulmak için dizinin [n,0] terimi kullanılır. Örnek olarak Tablo 7.1.1 de verilen 4 uzunluklu kelimeden 18 tane var. Bunu All_Hall dizisi içinde aşağıdaki gibi belirleriz

All_Hall[4,0].index:=18

**Programdaki fonksiyonlar ve alt programlar 7.2.3.:** Programda kullanılan fonksiyon ve alt programlardan en önemlileri aşağıda açıklanmıştır.

function Hall_elemanimi(h1,h2:HALL_TYPE) : boolean;

Bu fonksiyon, birinci bileşen h1, ikinci bileşen h2 olan [h1,h2] çarpımı ile verilen kelimenin Hall elemanı olup olmadığını belirler. Bunu Hall kümesinin tanımında verilen şartlara bakarak belirler.

procedure Hall_Bul();

Bu alt program istenilen uzunluktaki tüm Hall kelimelerin belirler ve All_Hall değişkeninin içerisine atar.

procedure Hallkelime(h:HALL_TYPE; var eleman:string);

Bu alt program h şeklinde verilen bir Hall elemanını braketli hale getirir.

procedure Hall_Yazdir();

Bu alt program elde edilen tüm Hall kelimelerini ekrana yazdırır.

procedure HallToLyndon(h:HALL_TYPE; var eleman:string);

Bu alt program h Hall elemanını Lyndon kelimesine çevirir.

**Programın Delphi7 dilindeki yazılımı 7.2.4.:**

```
unit Ebubekir_Hall;

interface

uses
  Windows,  Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,  Dialogs, StdCtrls, ComCtrls, Grids;

type
  HALL_TYPE = record
    index:longword;   // bu elemanın bu uzunluktaki indexi
    I1:longword;      // birinci bileşen indexi
    I2:longword;      // ikinci bileşen indexi
```

```
  L1:byte;        // birinci bileşen uzunluğu
  L2:byte;        // ikinci bileşen uzunluğu
end;

 TForm1 = class(TForm)
  Label1: TLabel;
  Edit1: TEdit;
  Label3: TLabel;
  Button1: TButton;
  Button2: TButton;
  Label2: TLabel;
  Edit2: TEdit;
  Label5: TLabel;
  Edit3: TEdit;
  Edit4: TEdit;
  Edit5: TEdit;
  Edit6: TEdit;
  Edit7: TEdit;
  Label6: TLabel;
  Label7: TLabel;
  Label8: TLabel;
  Label9: TLabel;
  Label10: TLabel;
  ProgressBar1: TProgressBar;
  ProgressBar2: TProgressBar;
  Label16: TLabel;
  Label17: TLabel;
  Edit8: TEdit;
  Edit9: TEdit;
  Label18: TLabel;
  Label19: TLabel;
```

```
    GroupBox1: TGroupBox;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    Label20: TLabel;
    RadioButton5: TRadioButton;
    StringGrid1: TStringGrid;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  hbase,hbase1,hbase2,hbaseg :  HALL_TYPE;
  All_Hall: array [1..20,0..1000000] of HALL_TYPE;
  // özel olarak "All_Hall[4,0].index:=7;" nin anlamı
  //  4 uzunluklu 7 kelime var demektir.
  X: array[1..20] of string= ('x','y','z','a','b','c','d','h','i','j',
                              'k','l','m','n','o','p','r','s','t','u');
  rankX,max_len,m1,m2,m3,m4,m5:longword;
  enaz,encok:longword;
  Hall_eleman_sayisi:longword;

  Form1: TForm1;

implementation

{$R *.dfm}
//*************************************************************
procedure Hall_in(var h:HALL_TYPE;index,I1,I2,L1,L2:longword )  ;
```

```
 // H ye değer atar
begin
  h.index:=index;
  h.I1:=I1; h.I2:=I2;
  h.L1:=L1; h.L2:=L2;
end;
//*****************************************************************
function X_len(h:HALL_TYPE):longword;
// h nin uzunluğunu veren fonksiyon
begin
  X_len:=h.L1+h.L2;
end;
//*****************************************************************
function Cm1_len(h:HALL_TYPE):longword;
// h nin Cm1 uzunluğunu verir
var    h1,h2:HALL_TYPE;
       uz:longword;
begin
  uz:=0;
  if X_len(h)<m1 then uz:=0
  else if X_len(h)=m1 then  uz:=1
  else begin                        // X_len(h)> m1  olur
     h1:=All_Hall[h.L1,h.I1];
     if X_len(h1)>=m1 then uz:=Cm1_len(h1);
     h2:=All_Hall[h.L2,h.I2];
     if X_len(h2)>=m1 then uz:=uz+Cm1_len(h2);
    end;
  Cm1_len:=uz;
end;
//*****************************************************************
function Cm1m2_len(h:HALL_TYPE):longword;
```

```
var    h1,h2:HALL_TYPE;
       uz:longword;
begin
  uz:=0;
  if Cm1_len(h)<m2 then uz:=0
  else if Cm1_len(h)=m2 then  uz:=1
  else begin
     h1:=All_Hall[h.L1,h.I1];
     if Cm1_len(h1)>=m2 then uz:=Cm1m2_len(h1);
     h2:=All_Hall[h.L2,h.I2];
     if Cm1_len(h2)>=m2 then uz:=uz+Cm1m2_len(h2);
    end;
  Cm1m2_len:=uz;
end;
//*****************************************************************
function Cm1m2m3_len(h:HALL_TYPE):longword;
var    h1,h2:HALL_TYPE;
       uz:longword;
begin
  uz:=0;
  if Cm1m2_len(h)<m3 then uz:=0
  else if Cm1m2_len(h)=m3 then  uz:=1
  else begin
     h1:=All_Hall[h.L1,h.I1];
     if Cm1m2_len(h1)>=m3 then uz:=Cm1m2m3_len(h1);
     h2:=All_Hall[h.L2,h.I2];
     if Cm1m2_len(h2)>=m3 then uz:=uz+Cm1m2m3_len(h2);
    end;
  Cm1m2m3_len:=uz;
end;
//*****************************************************************
```

```
function Cm1m2m3m4_len(h:HALL_TYPE):longword;
var    h1,h2:HALL_TYPE;
       uz:longword;
begin
  uz:=0;
  if Cm1m2m3_len(h)<m4 then uz:=0
  else if Cm1m2m3_len(h)=m4 then  uz:=1
  else begin
     h1:=All_Hall[h.L1,h.I1];
     if Cm1m2m3_len(h1)>=m4 then uz:=Cm1m2m3m4_len(h1);
     h2:=All_Hall[h.L2,h.I2];
     if Cm1m2m3_len(h2)>=m4 then uz:=uz+Cm1m2m3m4_len(h2);
    end;
  Cm1m2m3m4_len:=uz;
end;
//*****************************************************************
function Cm1m2m3m4m5_len(h:HALL_TYPE):longword;
var h1,h2:HALL_TYPE;
    uz:longword;
begin
  uz:=0;
  if Cm1m2m3m4_len(h)<m5 then uz:=0
  else if Cm1m2m3m4_len(h)=m5 then  uz:=1
  else begin
    h1:=All_Hall[h.L1,h.I1];
    if Cm1m2m3m4_len(h1)>=m5 then uz:=Cm1m2m3m4m5_len(h1);
    h2:=All_Hall[h.L2,h.I2];
    if Cm1m2m3m4_len(h2)>=m5 then uz:=uz+Cm1m2m3m4m5_len(h2);
  end;
  Cm1m2m3m4m5_len:=uz;
end;
```

```
//*************************************************************
function Hall_elemanimi(h1,h2:HALL_TYPE) : boolean;
begin
  Hall_elemanimi:=false;
  if X_len(h1)>X_len(h2) then  exit;
  if X_len(h1)=X_len(h2) then
    if (h1.index<h2.index) then begin
      Hall_elemanimi:=true;  exit
    end
    else exit;
  // demekki küçük   yani len(h1)<len(h2)
  if (X_len(h1) < h2.L1) then exit;
  if (X_len(h1) > h2.L1) then  begin
    Hall_elemanimi:=true;
    exit
  end;
  // demekki eşit  ozaman indexlerine bakalım
  if h1.index>=h2.I1 then  Hall_elemanimi:=true;
  // değilse zaten false idi. ve çıkar.
end;
//*************************************************************
function HCm1_elemanimi(h:HALL_TYPE;var kactaneCm1var:longint ) : boolean;
// Hall bazındaki h elemanı aceba HCm nin elemanimi?
begin
  kactaneCm1var:=Cm1_len(h);
  if kactaneCm1var>0  then HCm1_elemanimi:=true else HCm1_elemanimi:=false;
end;
//*************************************************************
function HCm1m2_elemanimi(h:HALL_TYPE;var kactaneCm1m2var:longint ) :
boolean;  // Hall bazındaki h elemanı aceba HCm nin elemanimi?
begin
```

```
  kactaneCm1m2var:=Cm1m2_len(h);
  if kactaneCm1m2var>0  then HCm1m2_elemanimi:=true else
HCm1m2_elemanimi:=false;
end;
//*************************************************************
function HCm1m2m3_elemanimi(h:HALL_TYPE;var kactaneCm1m2m3var:longint
) : boolean;  // Hall bazındaki h elemanı aceba HCm nin elemanimi?
begin
  kactaneCm1m2m3var:=Cm1m2m3_len(h);
  if kactaneCm1m2m3var>0  then HCm1m2m3_elemanimi:=true else
HCm1m2m3_elemanimi:=false;
end;
//*************************************************************
function HCm1m2m3m4_elemanimi(h:HALL_TYPE;var
kactaneCm1m2m3m4var:longint ) : boolean;  // Hall bazındaki h elemanı aceba
HCm nin elemanimi?
begin
  kactaneCm1m2m3m4var:=Cm1m2m3m4_len(h);
  if kactaneCm1m2m3m4var>0  then HCm1m2m3m4_elemanimi:=true else
HCm1m2m3m4_elemanimi:=false;
end;
//*************************************************************
function HCm1m2m3m4m5_elemanimi(h:HALL_TYPE;var
kactaneCm1m2m3m4m5var:longint ) : boolean;  // Hall bazındaki h elemanı aceba
HCm nin elemanimi?
begin
  kactaneCm1m2m3m4m5var:=Cm1m2m3m4m5_len(h);
  if kactaneCm1m2m3m4m5var>0  then HCm1m2m3m4m5_elemanimi:=true else
HCm1m2m3m4m5_elemanimi:=false;
end;
//*************************************************************
```

```
procedure Hallkelime(h:HALL_TYPE; var eleman:string);
// h yi kelimeye çevirir
var h1,h2:HALL_TYPE;
begin
  if (h.L1=1)and (h.L2=0) then eleman:=X[h.I1];
  if (h.L1=1)and (h.L2=1) then eleman:=eleman+'['+X[h.I1]+','+X[h.I2]+']';
  if (h.L1=1)and (h.L2>1) then begin
    eleman:=eleman+'['+X[h.I1]+',';
    h1:=All_Hall[h.L2,h.I2];
    Hallkelime(h1,eleman);
    eleman:=eleman+']';
  end;
  if (h.L1>1) then begin
    eleman:=eleman+'[';
    h1:=All_Hall[h.L1,h.I1];
    Hallkelime(h1,eleman);
    eleman:=eleman+',';
    h2:=All_Hall[h.L2,h.I2];
    Hallkelime(h2,eleman);
    eleman:=eleman+']';
  end;
end;
//************************************************************
procedure HallToLyndon(h:HALL_TYPE; var eleman:string);
// halltype ındaki bir h elemanını Lyndon kelimesine çevirir
var
  h1,h2:HALL_TYPE;
  i,j:integer;
  gec:char;
  mineleman:string;
```

```
begin
  if (h.L1=1)and (h.L2=0) then eleman:=X[h.I1];
  if (h.L1=1)and (h.L2=1) then eleman:=eleman+X[h.I1]+X[h.I2];
  if (h.L1=1)and (h.L2>1) then begin
    eleman:=eleman+X[h.I1];
    h1:=All_Hall[h.L2,h.I2];
    HallToLyndon(h1,eleman);
    eleman:=eleman;
  end;
  if (h.L1>1) then begin
    eleman:=eleman;
    h1:=All_Hall[h.L1,h.I1];
    HallToLyndon(h1,eleman);
    eleman:=eleman;
    h2:=All_Hall[h.L2,h.I2];
    HallToLyndon(h2,eleman);
    eleman:=eleman;
  end;
{
 Hall elemanı assosiyatif sapport haline geldi
 örnegin: [a,[b,c]] elemanı abc gibi oldu
 şimdi bu kelimeden Lyndon kelimesi bulmalıyız.
 Bunun devirli permütasyonundan
 faydalanarak assosiyatif supportlarından
 en küçüğünü yani primitif olanını  bulacağız
}
  mineleman:=eleman;
  for j:=2 to length(eleman) do begin
    gec:=eleman[1];
    for i:=2 to length(eleman) do begin
      eleman[i-1]:=eleman[i]
```

```
   end;
   eleman[length(eleman)]:=gec;
   if  mineleman>eleman then mineleman:=eleman;
  end;
  eleman:=mineleman;
end;
//*****************************************************************
procedure X_i_H1_e_aktar(n:longword);
var i :Longword;
begin
  for i:=1 to n do  Hall_in(All_Hall[1,i],i,i,0,1,0);
  All_Hall[1,0].index:=n;
end;
//*****************************************************************
procedure TForm1.Button2Click(Sender: TObject);
begin
 Halt
end;
//*****************************************************************
Procedure EkranDegerleriniYenile; // formdaki değişiklikleri yeniler sayıları atar
var
  s:string;
  i:longword;
begin
  s:=form1.edit1.text;  val(s,rankX,i);
  s:=form1.edit2.text;  val(s,max_len,i);
  s:=form1.edit3.text;  val(s,m1,i);
   if m1<2 then m1:=2; str(m1,s);form1.edit3.text:=s;
  s:=form1.edit4.text;  val(s,m2,i);
  s:=form1.edit5.text;  val(s,m3,i);
  s:=form1.edit6.text;  val(s,m4,i);
```

```
  s:=form1.edit7.text;  val(s,m5,i);
  s:=form1.edit8.text;  val(s,enaz,i);
  s:=form1.edit9.text;  val(s,encok,i);
  if (enaz> encok) then begin i:=enaz; enaz:=encok; encok:=i end;
  if (enaz<1) or (enaz>max_len)  then enaz:=1;
  if (encok>max_len) or (encok <1) then encok:=Max_len;
  str(enaz,s);form1.edit8.text:=s;
  str(encok,s);form1.edit9.text:=s;
end;
//*************************************************************
 procedure Hall_Yazdir();
var
  i,j,sayac,
  kactaneCm1var,
  kactaneCm1m2var,
  kactaneCm1m2m3var,
  kactaneCm1m2m3m4var,
  kactaneCm1m2m3m4m5var:longint;
  sm1,sm2,sm3,sm4,sm5,s1,elemang:string;
  On_ek,Son_ek : string;
  FHtml: TextFile;
  FText: TextFile;
begin
  AssignFile(FHtml, 'Ebubekir_Hall_Renkli.html'); Rewrite(FHtml);
  AssignFile(FText, 'Ebubekir_Hall.txt'); Rewrite(FText);
  EkranDegerleriniYenile;
  str(m1,sm1);
  str(m2,sm2);
  str(m3,sm3);
  str(m4,sm4);
  str(m5,sm5);
```

```
  if form1.RadioButton3.Checked then begin
    form1.StringGrid1.Visible:=true ;
    form1.StringGrid1.Cells[1,0]:='Hall Bazının elemanları';
    form1.StringGrid1.Cells[2,0]:='F';
    form1.StringGrid1.Cells[3,0]:='F'+sm1;
    form1.StringGrid1.Cells[4,0]:='F'+sm1+','+sm2;
    form1.StringGrid1.Cells[5,0]:='F'+sm1+','+sm2+','+sm3;
    form1.StringGrid1.Cells[6,0]:='F'+sm1+','+sm2+','+sm3+','+sm4;
    form1.StringGrid1.Cells[7,0]:='F'+sm1+','+sm2+','+sm3+','+sm4+','+sm5;
    form1.StringGrid1.Cells[8,0]:='Lyndon kelime';
  end;
  sayac:=0;
  Hall_eleman_sayisi:=0;
  for i:=1 to max_len do
    for j:=1 to All_Hall[i,0].index do
      inc(Hall_eleman_sayisi);
  if form1.RadioButton3.Checked then
form1.StringGrid1.RowCount:=Hall_eleman_sayisi+1;
  for i:=enaz to encok do
    for j:=1 to All_Hall[i,0].index do  begin
      inc(sayac);
      if form1.RadioButton3.Checked  then begin
        form1.StringGrid1.Cells[0,sayac]:='';
        form1.StringGrid1.Cells[1,sayac]:='';
        form1.StringGrid1.Cells[2,sayac]:='';
        form1.StringGrid1.Cells[3,sayac]:='';
        form1.StringGrid1.Cells[4,sayac]:='';
        form1.StringGrid1.Cells[5,sayac]:='';
        form1.StringGrid1.Cells[6,sayac]:='';
        form1.StringGrid1.Cells[7,sayac]:='';
      end;
```

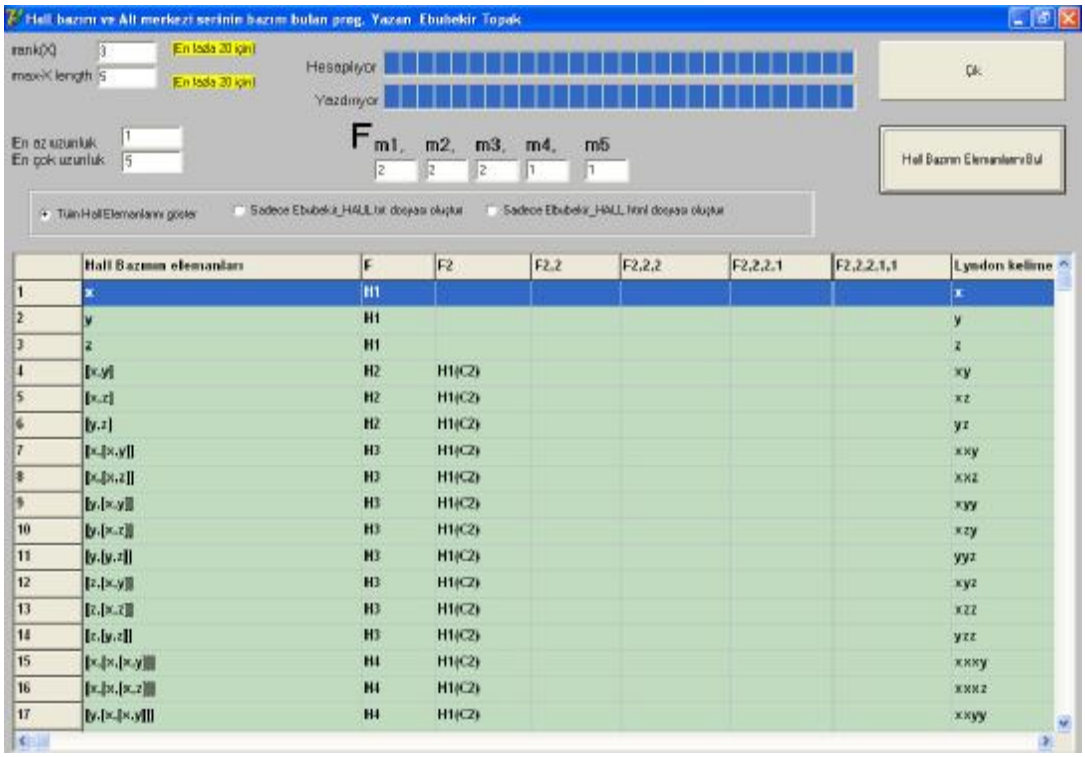
```
elemang:=''; HallToLyndon(All_Hall[i,j], elemang);
form1.StringGrid1.Cells[8,sayac]:=elemang;
form1.ProgressBar2.Position:=(sayac*100) div Hall_eleman_sayisi;
str(sayac,s1);
if form1.RadioButton3.Checked then
 form1.StringGrid1.Cells[0,sayac]:=s1;
str(i,s1);
if form1.RadioButton3.Checked then
 form1.StringGrid1.Cells[2,sayac]:='H'+s1;
elemang:=''; Hallkelime(All_Hall[i,j], elemang);
On_ek:='<p style="line-height: 10%"><span style="background-color:
#FFFFFF">';
kactaneCm1var:=0;
if HCm1_elemanimi(All_Hall[i,j],kactaneCm1var)
and (m1<>1) then begin
 On_ek:='<p style="line-height:10%"><font color="#FF0000">';
 str(kactaneCm1var,s1);
 if form1.RadioButton3.Checked  then
  form1.StringGrid1.Cells[3,sayac]:='H'+s1+'(C'+sm1+')';
 kactaneCm1m2var:=0;
 if HCm1m2_elemanimi(All_Hall[i,j],kactaneCm1m2var)
 and (m2<>1) then begin
  On_ek:='<p style="line-height:10%"><font color="#00FF00">';
  str(kactaneCm1m2var,s1);
  if form1.RadioButton3.Checked  then
   form1.StringGrid1.Cells[4,sayac]:='H'+s1+'(C'+sm1+','+sm2+')';
  kactaneCm1m2m3var:=0;
  if HCm1m2m3_elemanimi(All_Hall[i,j],kactaneCm1m2m3var)
  and (m3<>1) then begin
   On_ek:='<p style="line-height:10%"><font color="#0000FF">';
   str(kactaneCm1m2m3var,s1);
```

```
          if form1.RadioButton3.Checked  then
           form1.StringGrid1.Cells[5,sayac]:='H'+s1+'(C'+sm1+','+sm2+','+sm3+')';
          kactaneCm1m2m3m4var:=0;
          if HCm1m2m3m4_elemanimi(All_Hall[i,j],kactaneCm1m2m3m4var)
          and (m4<>1) then begin
           On_ek:='<p style="line-height:10%"><font color="#F0000F">';
           str(kactaneCm1m2m3m4var,s1);
           if form1.RadioButton3.Checked  then

form1.StringGrid1.Cells[6,sayac]:='H'+s1+'(C'+sm1+','+sm2+','+sm3+','+sm4+')';
           kactaneCm1m2m3m4m5var:=0;
           if HCm1m2m3m4m5_elemanimi(All_Hall[i,j],kactaneCm1m2m3m4m5var)
           and(m5<>1) then begin
            On_ek:='<p style="line-height:10%"><font color="#0F000F">';
            str(kactaneCm1m2m3m4m5var,s1);
           if form1.RadioButton3.Checked then

form1.StringGrid1.Cells[7,sayac]:='H'+s1+'(C'+sm1+','+sm2+','+sm3+','+sm4+','+sm
5+')';
           end;
         end;
        end;
      end;
   end;
   Son_ek:='</span></p>';
   if form1.RadioButton3.Checked then
    form1.StringGrid1.Cells[1,sayac]:=elemang;
   if form1.RadioButton4.Checked then
    writeln(FText,elemang);
   if form1.RadioButton5.Checked then
    writeln(FHtml,on_ek+elemang+son_ek );
```

```
  end;
  form1.StringGrid1.Visible:=true;
  CloseFile(FHtml);
  CloseFile(FText);
end;
//*************************************************************
procedure Hall_Bul();
var
  i,k,j,p,t,sayac:longword;
begin
  Hall_eleman_sayisi:=0;
  EkranDegerleriniYenile;
  X_i_H1_e_aktar(rankX);
  for k:=2 to max_len do begin
    form1.ProgressBar1.Position:=(k*100) div max_len;
    i:=1;j:=k-1;
    sayac:=0;
    while (i<=j) do begin
      for t:=1 to All_Hall[i,0].index do begin
        for p:=1 to All_Hall[j,0].index do begin
          if Hall_elemanimi(All_Hall[i,t],All_Hall[j,p]) then begin
            inc(sayac);
            Hall_in(hbase,sayac,t,p,i,j);
            All_Hall[k,sayac]:=hbase;
            inc(Hall_eleman_sayisi);
          end;
        end;

      end;
      i:=i+1;  j:=j-1;
    end;
```

```
    All_Hall[k,0].index:=sayac;
  end;
end;
//*****************************************************************
procedure TForm1.Button1Click(Sender: TObject);
begin
  Hall_Bul;
  Hall_Yazdir
end;
end.
```

**Örnek 7.2.5.:** Programın çalıştırılması sonucunda aşağıdaki parametreler girilerek görüntüdeki çıktı elde edilmiştir.



## 7.3. Koboyutu hesaplayan Delpi7 Programı

Aşağıdaki bilgisayar programı Ela AYDIN (1997) tarafından verilen sonlu takdimli bir $L$ Lie cebirinin herhangi bir $B$ alt cebirinin üreteçleri bilindiğinde koboyutu veren algoritmanın bilgisayar programıdır. Bunu yaparken 7.2.4 te verdiğimiz Hall elemanlarını bulan programı $L$ nin ve $B$ nin Hall Bazı elemanlarını bulmak için de kullanır.

**Programın Delphi7 dilindeki yazılımı 7.3.1.:**

```
unit ebubekirHall;
interface
uses
  Windows,  Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, Grids, ExtCtrls;
type
 HALL_TYPE = record
    index:longword;   // bu elemanın bu uzunluktaki indexi
    I1:longword;      // birinci bileşen indexi
    I2:longword;      // ikinci bileşen indexi
    L1:byte;          // birinci bileşen uzunluğu
    L2:byte;          // ikinci bileşen uzunluğu
 end;
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Button1: TButton;
    Button2: TButton;
    Label2: TLabel;
    Edit2: TEdit;
```

Label5: TLabel;
ProgressBar1: TProgressBar;
Edit8: TEdit;
Edit9: TEdit;
Label18: TLabel;
Label19: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Edit3: TEdit;
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Label14: TLabel;
Memo3: TMemo;
Label15: TLabel;
Memo4: TMemo;
Label17: TLabel;
Label20: TLabel;
Shape1: TShape;
Shape2: TShape;
Label4: TLabel;
GroupBox1: TGroupBox;
Label6: TLabel;
Edit7: TEdit;
Edit10: TEdit;
Edit11: TEdit;
Edit12: TEdit;

Edit13: TEdit;
Edit14: TEdit;
Edit15: TEdit;
Edit16: TEdit;
Edit17: TEdit;
Edit18: TEdit;
Label13: TLabel;
Label16: TLabel;
Label21: TLabel;
Label22: TLabel;
Label23: TLabel;
Label24: TLabel;
Label25: TLabel;
Label26: TLabel;
Label27: TLabel;
GroupBox2: TGroupBox;
Label28: TLabel;
Label29: TLabel;
Label30: TLabel;
Label31: TLabel;
Label32: TLabel;
Label33: TLabel;
Label34: TLabel;
Label35: TLabel;
Label36: TLabel;
Label37: TLabel;
Edit19: TEdit;
Edit20: TEdit;
Edit21: TEdit;
Edit22: TEdit;
Edit23: TEdit;

Edit24: TEdit;
Edit25: TEdit;
Edit26: TEdit;
Edit27: TEdit;
Edit28: TEdit;
GroupBox3: TGroupBox;
Label38: TLabel;
Label39: TLabel;
Label40: TLabel;
Label41: TLabel;
Label42: TLabel;
Label43: TLabel;
Label44: TLabel;
Label45: TLabel;
Label46: TLabel;
Label47: TLabel;
Edit29: TEdit;
Edit30: TEdit;
Edit31: TEdit;
Edit32: TEdit;
Edit33: TEdit;
Edit34: TEdit;
Edit35: TEdit;
Edit36: TEdit;
Edit37: TEdit;
Edit38: TEdit;
memo5: TMemo;
Label48: TLabel;
Edit39: TEdit;
Label49: TLabel;
Label50: TLabel;

```
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
//-------------------------------------------------
  hbase,hbase1,hbase2,hbaseg :  HALL_TYPE;
  All_Hall: array [1..20,0..1000000] of HALL_TYPE;
                // özel olarak "All_Hall[4,0].index:=7;" nin anlamı
                //  4 uzunluklu 7 kelime var demektir.
  X: array[1..10] of string;
  rankX,max_len:longword;
  enaz,encok:longword;
  Hall_eleman_sayisi:longword;
  rankR: longint; // L nin R bağıntılarının eleman sayısı
  R: array[1..10] of string;
//-------------------------------------------------
  // B<X> için parametreler_______ farklı olsun diye _1 ekledim
  hbase_1,hbase1_1,hbase2_1,hbaseg_1 :  HALL_TYPE;
  All_Hall_1: array [1..20,0..1000000] of HALL_TYPE;
  X_1: array[1..10] of string;
  rankX_1,max_len_1:longword;
  enaz_1,encok_1:longword;
  Hall_eleman_sayisi_1:longword;
Form1: TForm1;
implementation
{$R *.dfm}
```

```
//*****************************************************
procedure Hall_in(var h:HALL_TYPE;index,I1,I2,L1,L2:longword ) ;
      // H ye değer atar
begin
     h.index:=index;
     h.I1:=I1; h.I2:=I2;
     h.L1:=L1; h.L2:=L2;
end;
//*****************************************************
function X_len(h:HALL_TYPE):longword;
// X_len X_len X_len X_len X_len X_len
begin
  X_len:=h.L1+h.L2;
end;
//*****************************************************
function Hall_elemanimi(h1,h2:HALL_TYPE) : boolean;
begin
  Hall_elemanimi:=false;
  if X_len(h1)>X_len(h2) then  exit;
  if X_len(h1)=X_len(h2) then if (h1.index<h2.index) then begin
    Hall_elemanimi:=true;  exit end
  else exit;
     // demekki küçük   yani len(h1)<len(h2)
  if (X_len(h1) < h2.L1) then exit;
  if  (X_len(h1) > h2.L1) then  begin Hall_elemanimi:=true;  exit end;
       // demekki eşit  ozaman indexlerine bakalım
  if h1.index>=h2.I1 then  Hall_elemanimi:=true;   // değilse zaten false idi. ve
çıkar.
end;
//*****************************************************
```

```
procedure Hallkelime(h:HALL_TYPE; var eleman:ShortString   );   // halltype
ındaki bir elemanı kelimeye çevirir
  var h1,h2:HALL_TYPE;
begin
    if (h.L1=1)and (h.L2=0) then eleman:=X[h.I1];
    if (h.L1=1)and (h.L2=1) then eleman:=eleman+'['+X[h.I1]+','+X[h.I2]+']';
    if (h.L1=1)and (h.L2>1) then begin
      eleman:=eleman+'['+X[h.I1]+',';
      h1:=All_Hall[h.L2,h.I2];
      Hallkelime(h1,eleman);
      eleman:=eleman+']';
    end;
   if (h.L1>1) then begin
     eleman:=eleman+'[';
     h1:=All_Hall[h.L1,h.I1];
     Hallkelime(h1,eleman);
     eleman:=eleman+',';
     h2:=All_Hall[h.L2,h.I2];
     Hallkelime(h2,eleman);
     eleman:=eleman+']';
   end;
end;
//*************************************************
procedure Hallkelime_1(h:HALL_TYPE; var eleman:ShortString        );   //
halltype ındaki bir elemanı kelimeye çevirir
  var h1,h2:HALL_TYPE;
begin
    if (h.L1=1)and (h.L2=0) then eleman:=X_1[h.I1];
    if (h.L1=1)and (h.L2=1) then
eleman:=eleman+'['+X_1[h.I1]+','+X_1[h.I2]+']';
    if (h.L1=1)and (h.L2>1) then begin
```

```
      eleman:=eleman+'['+X_1[h.I1]+',';
      h1:=All_Hall_1[h.L2,h.I2];
      Hallkelime_1(h1,eleman);
      eleman:=eleman+']';
    end;
   if (h.L1>1) then begin
     eleman:=eleman+'[';
     h1:=All_Hall_1[h.L1,h.I1];
     Hallkelime_1(h1,eleman);
     eleman:=eleman+',';
     h2:=All_Hall_1[h.L2,h.I2];
     Hallkelime_1(h2,eleman);
     eleman:=eleman+']';
   end;
end;
//*****************************************************
procedure X_i_H1_e_aktar_1(n:longword);
var i :Longword;
begin
  for i:=1 to n do  Hall_in(All_Hall_1[1,i],i,i,0,1,0);
  All_Hall_1[1,0].index:=n;
end;
//*****************************************
procedure X_i_H1_e_aktar(n:longword);
var i :Longword;
begin
  for i:=1 to n do  Hall_in(All_Hall[1,i],i,i,0,1,0);
  All_Hall[1,0].index:=n;
end;
//*****************************************
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
Halt
end;
//*****************************************
Procedure EkranDegerleriniYenile; // formdaki değişiklikleri yeniler sayıları
atar
  var s:string;
      i:longword;
begin
 x[ 1]:=form1.Edit7.Text;
 x[ 2]:=form1.Edit10.Text;
 x[ 3]:=form1.Edit11.Text;
 x[ 4]:=form1.Edit12.Text;
 x[ 5]:=form1.Edit13.Text;
 x[ 6]:=form1.Edit14.Text;
 x[ 7]:=form1.Edit15.Text;
 x[ 8]:=form1.Edit16.Text;
 x[ 9]:=form1.Edit17.Text;
 x[10]:=form1.Edit18.Text;

 R[ 1]:=form1.edit19.Text;
 R[ 2]:=form1.edit20.Text;
 R[ 3]:=form1.edit21.Text;
 R[ 4]:=form1.edit22.Text;
 R[ 5]:=form1.edit23.Text;
 R[ 6]:=form1.edit24.Text;
 R[ 7]:=form1.edit25.Text;
 R[ 8]:=form1.edit26.Text;
 R[ 9]:=form1.edit27.Text;
 R[10]:=form1.edit28.Text;
 x_1[ 1]:=form1.Edit29.Text;
```

```
x_1[ 2]:=form1.Edit30.Text;
x_1[ 3]:=form1.Edit31.Text;
x_1[ 4]:=form1.Edit32.Text;
x_1[ 5]:=form1.Edit33.Text;
x_1[ 6]:=form1.Edit34.Text;
x_1[ 7]:=form1.Edit35.Text;
x_1[ 8]:=form1.Edit36.Text;
x_1[ 9]:=form1.Edit37.Text;
x_1[10]:=form1.Edit38.Text;

s:=form1.edit39.text;  val(s,rankR,i);

s:=form1.edit1.text;  val(s,rankX,i);
s:=form1.edit2.text;  val(s,max_len,i);
s:=form1.edit8.text;  val(s,enaz,i);
s:=form1.edit9.text;  val(s,encok,i);
if (enaz> encok) then begin i:=enaz; enaz:=encok; encok:=i end;
if (enaz<1) or (enaz>max_len)  then enaz:=1;
if (encok>max_len) or (encok <1) then encok:=Max_len;
str(enaz,s);form1.edit8.text:=s;
str(encok,s);form1.edit9.text:=s;

s:=form1.edit3.text;  val(s,rankX_1,i);
s:=form1.edit4.text;  val(s,max_len_1,i);
s:=form1.edit5.text;  val(s,enaz_1,i);
s:=form1.edit6.text;  val(s,encok_1,i);
if (enaz_1> encok_1) then begin i:=enaz_1; enaz_1:=encok_1; encok_1:=i
end;
if (enaz_1<1) or (enaz_1>max_len_1)  then enaz_1:=1;
if (encok_1>max_len) or (encok_1 <1) then encok_1:=Max_len_1;
str(enaz_1,s);form1.edit5.text:=s;
```

```
 str(encok_1,s);form1.edit6.text:=s;
end;
//*****************************************************
function L_elemanimi( e:string) : boolean;
        // nin R ye göre L de olup olmadığına karar verir
var i: longword;
begin
      L_elemanimi:=false;
      for i:=1 to rankR do if (Pos(R[i],e)>0) then exit;
      L_elemanimi:=true;
end;
//*****************************************************
function uzunluk_eleman( e:shortstring) : longword;
        // e nin X-uzunluğunu verir
var i: longword;
begin
   i:=0;
   while Pos(',', e) > 0 do begin
     i:=i+1;
     e[Pos(',', e)] := ' ';
   end;
   uzunluk_eleman:=i+1;
end;
//*****************************************************
function B_maxlen( k:longword) : longword;
        // B deki k uzunluklu elemanların içindeki
        // X-uzunlığu en büyük elemanın X-uzunluğu nu verir
var j,max: longword;
   elemang:shortstring;
begin
   max:=0;
```

```
    for j:=1 to All_Hall_1[k,0].index do begin
      Hallkelime_1(All_Hall_1[k,j],elemang);
      if max < uzunluk_eleman(elemang) then
        max:=uzunluk_eleman(elemang);
    end;
    B_maxlen:=max;
end;
//*****************************************************
 procedure Hall_Yazdir_R();  //bağıntılara göre L nin elemanları
var i,j:longword;
   elemang:ShortString  ;
begin
  form1.Memo3.Clear ;
  FOR i:=enaz TO enCOK DO BEGIN
      FOR j:=1 to All_Hall[i,0].index do begin
          elemang:=''; Hallkelime(All_Hall[i,j], elemang);
          if L_elemanimi(elemang) then form1.Memo3.Lines.Add(elemang);
      end;
  end;
end;
//*******************************************************
 procedure Hall_Yazdir_1_R(); //bağıntılara göre B nin elemanları
var i,j:longword;
   elemang:ShortString  ;
begin
  form1.Memo4.Clear ;
  FOR i:=enaz_1 TO enCOK_1 DO BEGIN
      FOR j:=1 to All_Hall_1[i,0].index do begin
          elemang:=''; Hallkelime_1(All_Hall_1[i,j], elemang);
          if L_elemanimi(elemang) then  form1.Memo4.Lines.Add(elemang);
      end;
```

```
     form1.Memo4.Lines.Add('');
  end;
end;
//*****************************************************
function  B_de_varmi(e:shortstring ):boolean;
                    // e elemanı B de var mı
var i,j:longword;
    elemang:shortstring;
begin
  B_de_varmi:=true;
  if not L_elemanimi(e)then exit;   // aslında yok ama var gibi gösterip iptal
edeceğiz
    FOR i:=1 TO 20 DO BEGIN
      FOR j:=1 to All_Hall_1[i,0].index do begin
          elemang:=''; Hallkelime_1(All_Hall_1[i,j], elemang);
          if L_elemanimi(elemang) and (e=elemang) then exit;
      end;
  end;
  B_de_varmi:=false;
end;
//--------------------------------
procedure Mod_Baz_Yazdir();
var
    baz_eleman: array [1..10000] of string;
    baz_eleman_sayisi: longword;
    i,j,k,m,n:longword;
    dahaoncevarmi:boolean;
    elemang: shortstring;
    S:boolean;  // Ela Aydın (1997) nın
                // tezindeki Sk kümesinin temsil ediyor
                // eğer S=false ise  Sk boş küme demektir
```

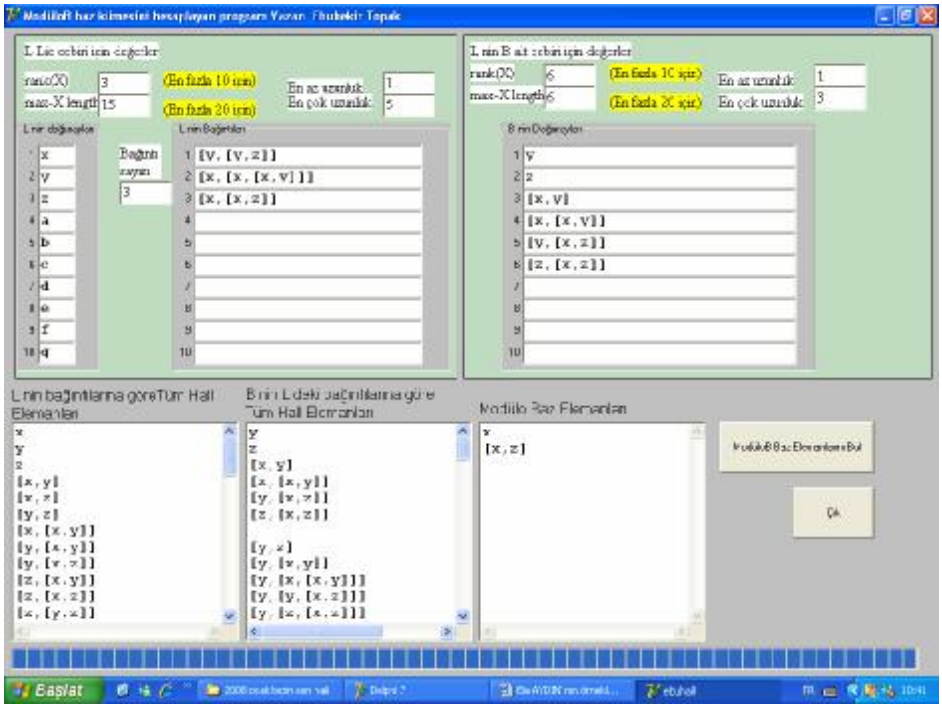
```
            // ve algoritma durur
            // S=true ise algoritma devam eder
begin
  form1.Memo5.Clear ;
  FOR k:=1 TO 10 DO BEGIN     // B nin en fazla 10 uzunluklu
     S:=false;              // elemanlarına kadar gidelim
     for i:=1 to B_maxlen(k) do begin
       FOR j:=1 to All_Hall[i,0].index do begin
         elemang:=''; Hallkelime(All_Hall[i,j], elemang);
         if B_de_varmi(elemang)=false then begin
           dahaoncevarmi:=false;
           for m:=1 to baz_eleman_sayisi do
             if baz_eleman[m]=elemang then dahaoncevarmi:=true;
               // daha önce yazılmamışsa yaz
           if dahaoncevarmi = false then begin
             S:=true;
             baz_eleman_sayisi:=baz_eleman_sayisi+1;
             baz_eleman[baz_eleman_sayisi]:=elemang;
           end;
         end;
       end;
       if S=false then begin
         for n:=1 to baz_eleman_sayisi do
           form1.Memo5.Lines.Add(baz_eleman[n]);
         exit;
       end;
     end;
  end;
end;
//*************************************************
procedure Hall_Bul();
```

```
var i,k,j,p,t,sayac:longword;
begin
  Hall_eleman_sayisi:=0;
  EkranDegerleriniYenile;
  X_i_H1_e_aktar(rankX);
  for k:=2 to max_len do begin
    form1.ProgressBar1.Position:=(k*100) div max_len;
    i:=1;j:=k-1;
    sayac:=0;
    while (i<=j) do begin
      for t:=1 to All_Hall[i,0].index do begin
        for p:=1 to All_Hall[j,0].index do begin
          if Hall_elemanimi(All_Hall[i,t],All_Hall[j,p]) then begin
            inc(sayac);
            Hall_in(hbase,sayac,t,p,i,j);
            All_Hall[k,sayac]:=hbase;
            inc(Hall_eleman_sayisi);
          end;   // if
        End;
      End;
      i:=i+1;  j:=j-1;
    end; //while
    All_Hall[k,0].index:=sayac;
  end; //for k
end;
//*****************************************************
procedure Hall_Bul_1();
var i,k,j,p,t,sayac:longword;
begin
  Hall_eleman_sayisi_1:=0;
  EkranDegerleriniYenile;
```

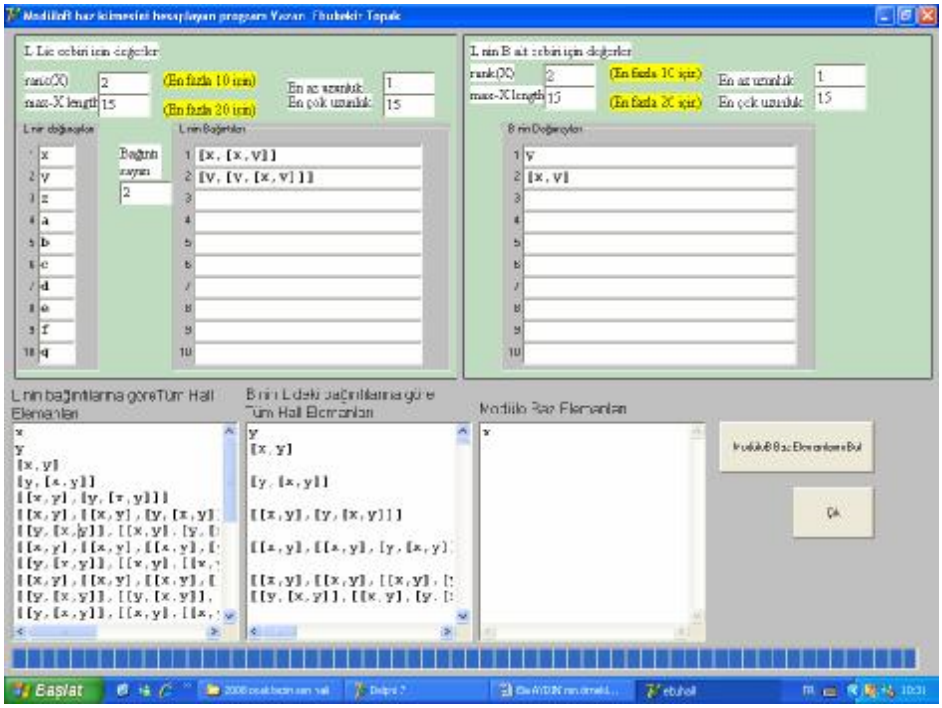
```
  X_i_H1_e_aktar_1(rankX_1);
  for k:=2 to max_len_1 do begin
  form1.ProgressBar1.Position:=(k*100) div max_len_1;
  i:=1;j:=k-1;
  sayac:=0;
  while (i<=j) do begin
    for t:=1 to All_Hall_1[i,0].index do begin
      for p:=1 to All_Hall_1[j,0].index do begin
        if Hall_elemanimi(All_Hall_1[i,t],All_Hall_1[j,p]) then begin
          inc(sayac);
          Hall_in(hbase,sayac,t,p,i,j);
          All_Hall_1[k,sayac]:=hbase;
          inc(Hall_eleman_sayisi_1);
        end;   // if
      End;
    End;
    i:=i+1;  j:=j-1;
  end; //while
  All_Hall_1[k,0].index:=sayac;
  end; //for k
end;
//****************************************************************
procedure TForm1.Button1Click(Sender: TObject);
begin
  Hall_Bul;
  Hall_Bul_1;
  Hall_Yazdir_R;
  Hall_Yazdir_1_R;
  Mod_Baz_Yazdir;
end;
//****************************************************************
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  form1.Memo3.Clear ;
  form1.Memo4.Clear ;
  form1.Memo5.Clear ;
end;
end.
```

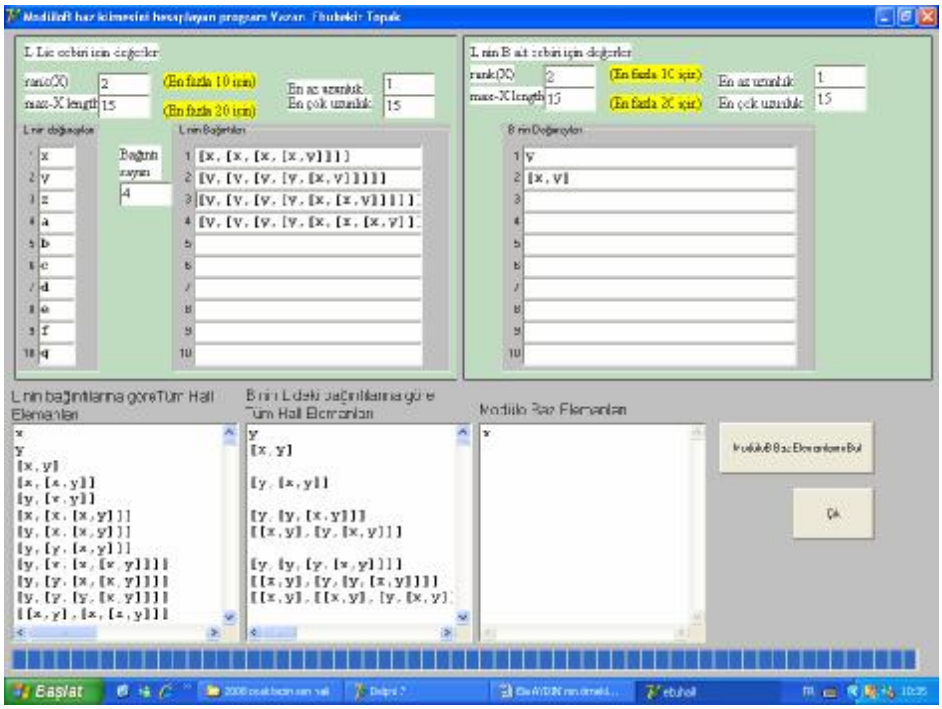
Aşağıda verilen bilgisayar çıktısı Örnek 6.1.1 in program tarafından elde edilen sonucudur.



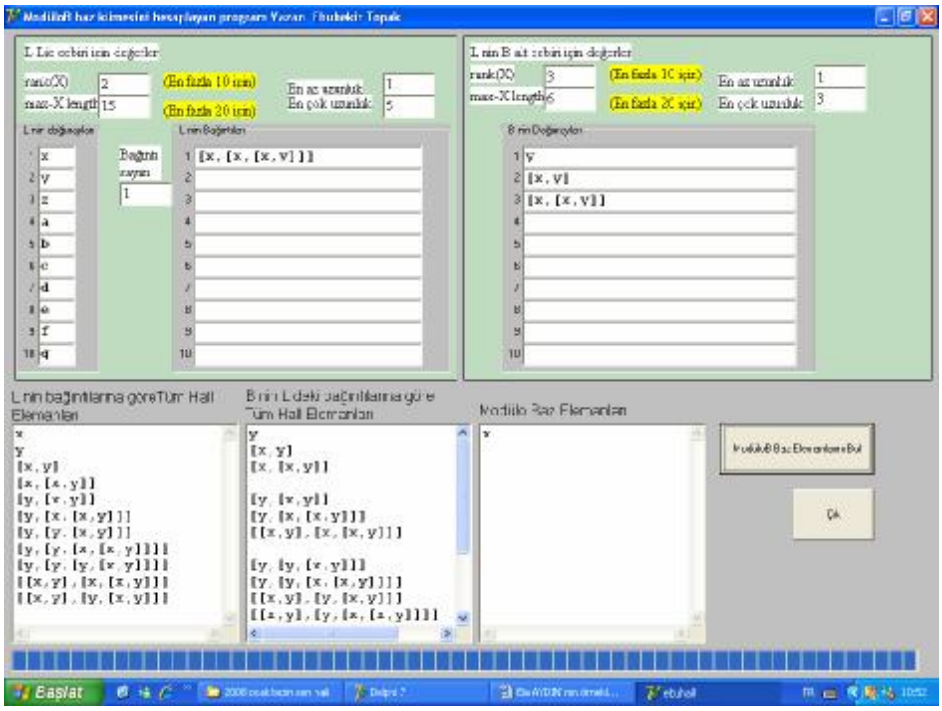
Aşağıda verilen bilgisayar çıktısı Örnek 6.1.2 nin program tarafından elde edilen sonucudur.



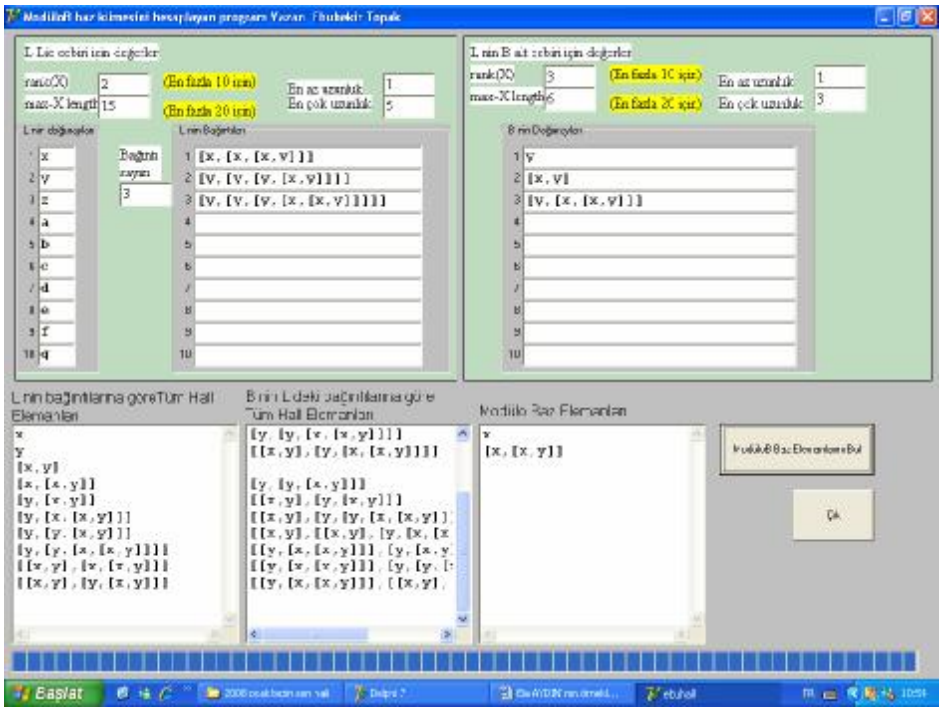
Aşağıda verilen bilgisayar çıktısı Örnek 6.1.3 ün program tarafından elde edilen sonucudur.



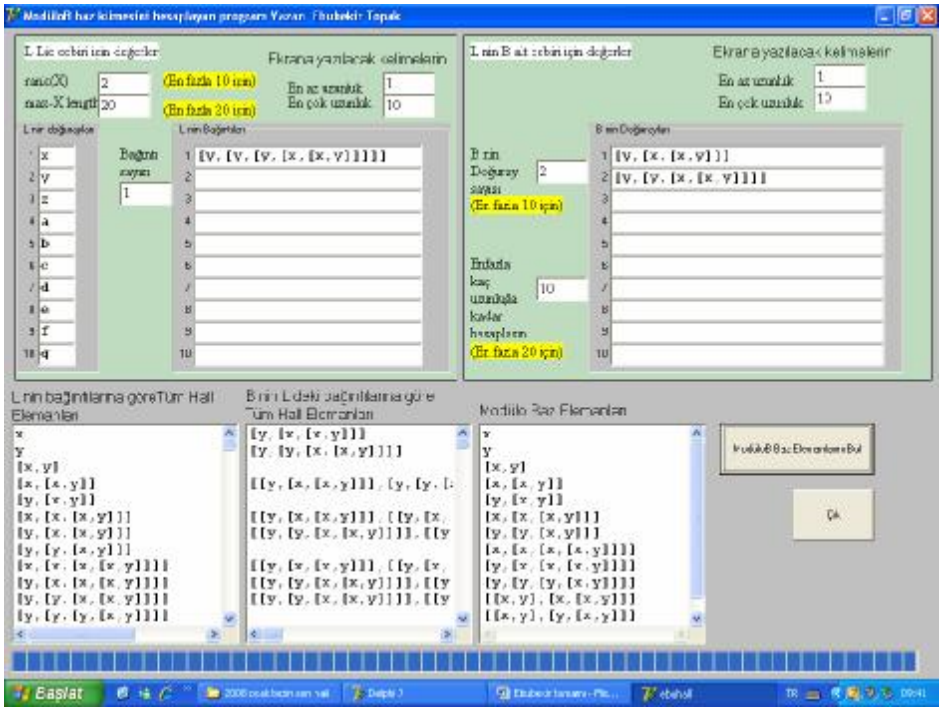
Aşağıda verilen bilgisayar çıktısı Örnek 6.1.4 ün program tarafından elde edilen sonucudur.



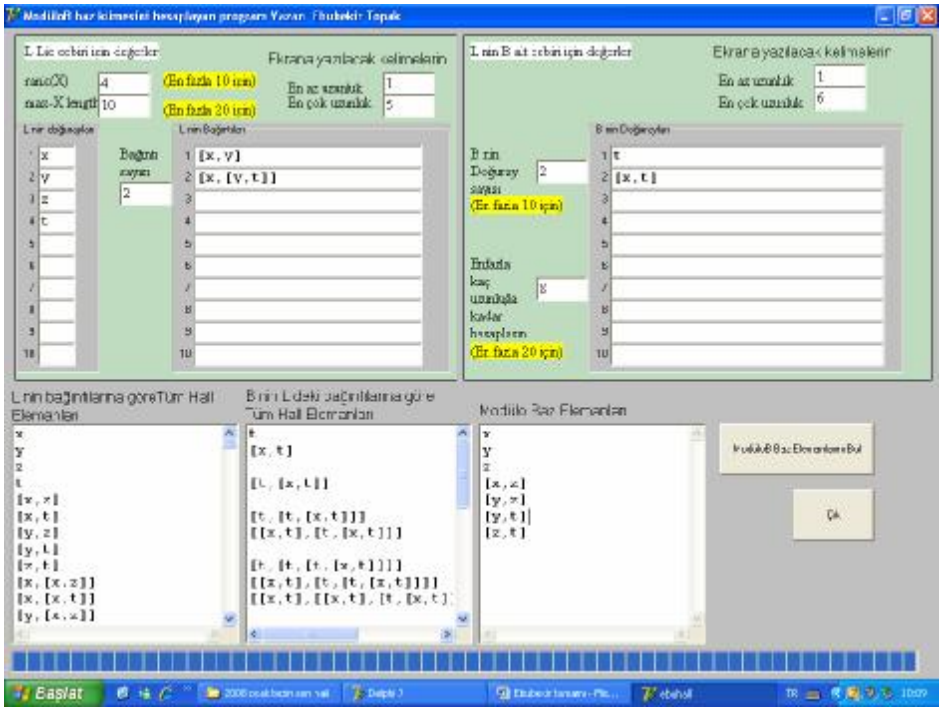
Aşağıda verilen bilgisayar çıktısı Örnek 6.1.5 in program tarafından elde edilen sonucudur.



Aşağıda verilen bilgisayar çıktısı Örnek 6.1.6 nın program tarafından elde edilen sonucudur.



Aşağıda verilen bilgisayar çıktısı Örnek 6.1.7 nin program tarafından elde edilen sonucudur.



## KAYNAKLAR


**ANDARY P. "**Finely homogeneous computations in free Lie algebras**"**
Discrete mathematics and theoretical computer science
1365-8050, 1, 1001, (1997)

**AYDIN E. "** Lie cebirlerinin sonlu koboyutlu alt cebirleri "
Çukurova Ü. Fen Bil. Enstitüsü 1997.

**BOURBAKI, N.** ''Groupes et abgebrès de Lie" Chapitre II.
Hermann, Paris (1972)

**CHAMPARNAUD J.-M., HANSEL G., PERRİN D.** **"**Unavoidable Sets of Constant Length." IJAC 14(2): 241-251 (2004)

**COHEN A. M., W, A. DE GRAF,** " Lie algeraic computations" Computer Physics Communications, 97 (1996), 53-62.

**DUVAL J.-P.** "Factorizing words over an ordered alphabet."
J. Algorithms, 4(4):363–381, (1983).

**FREDRICKSEN H., MAIORANA. J.** "Necklaces of beads in k colors and k-ary de Bruijn sequences." Discrete Math., 23(3):207–210, 1978.

**GERD V.P. KORNYAK V.V. "**Construction of finitely presented Lie algebras and superalgebras." Symbolic Computation 21, 337-349 (1996)

**GRAF, W, A.,** '' Lie algebras Theory and Algorithms'' First edition, Elsevier Science BV. 2000 the Netherlands''

**GRUENBERG, K.W.** '' Resudual properties of infinite soluble"
Proc. London Math. Soc. /, (1957),, 29-62

**KAAS H. M. VE OWREEN B.** ' Computations in free Lie algebras' Phil. Trans. R. Soc.
Lond. A (1999) 357, 957-981.

**KNUTH D.E.** "The Art of Computer Programming" volume 4. Addison Wesley, 2002. to appear, see
http://www-cs-faculty.stanford.edu/~knuth/news01.html.

**KUKIN, G.P.** "The subalgebras of a Lie p-algebras" Algebra Logika, 11(5):535-550 (1972)

**MORENO E.** "A note on the theorem of Fredericksen and Maiorana". Advances in Applied Mathematics, 2003. to appear.

**MUNTHE H.-KAAS AND OWREN B. "**Computations in a free Lie algebra**"** Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences Volume 357, Number 1754/April 15, 1999

**MYKKELTVEIT J.** " A proof of Golomb's conjecture for the de Bruijn graph." J. Combinatorial Theory Ser. B, 13:40–45, 1972.

**REUTENAUER C.** "Free Lie algebras." The Clarendon Press Oxford University Press, New York, 1993. Oxford Science Publications.

**SAWADA, J. F. RUSKEY** "Generating Lyndon brackets. An addendum to: Fast algorithms to generate necklaces, unlabeled necklaces and irreducible polynomials over GF(2)" Journal of algorithms, 0196-6774, 46, 1-21 (2003)

**SCHUTZENBERGER M. P.** " On a factorization of free monoids." Proc. Amer. Math. Soc., 16:21--24, (1965)

**SHIRSOV, A. I.** '' On free Lie rings " (Two parts)
I. Mat. Sbornik (Russian),42,(1957), 425-40
II. Mat. Sbornik (Russian),45,(1958), 113-22

**SHIRSOV, A. I.** '' Bases of free Lie algebras " (Two parts)
Algebra i Logika (Russian), 1, (1962), 14-19

**SMEL'KİN, A.L.** '' Free polynilpotent groups '' (Three parts)
I. Soviet Mth. Dokl. (Engl. Trans.) 4, (1963) 950-953
II. Izvest. Akad. Nauk S.S.S.R. Ser. Mat. (Russian) 28, (1964), 91-122
Trans. Amer. Math. Soc. (Engl. trans.) (1966) 270-304
III. Dokl. Akad. Nauk . S.S.S.R. 169, (1966), 1024-25

**STEWART, I.** '' Lie algebras '' Lecture notes in mathematics, No.127 (Springer, Berlin-Heidelberg-New York, 1970)

**WARM, M.** '' Basic commutators ''
Philos. Trans. Roy. Sos. London, Ser A 264 (1964), 343-432

**WARM, M.** '' Basic for polynilpotent groups ''
Proc. London Math. Soc. Ser 3, 24, (1972)

**WİTT, E.** '' Die unterrine der freien Liesschen ringe"
Math. Zeitschrift 64 (1956), 195-216

## ÖZGEÇMİŞ

1968 yılında Adana'nın Kozan ilçesinde doğdum. Öğrenimimi sırasıyla Kozan 2 Haziran ilkokulu, 50.yıl Orta okulu ve Adana Karşıyaka Lisesi'nde tamamladım. 1986 yılında 100. Yıl Üniversitesi Fen - Edebiyat Fakültesi Matematik bölümüne kayıt oldum. 1987 yılında Çukurova Üniversitesi Fen-Edebiyat Fakültesi Matematik Bölümü'ne yatay geçişle girdim. 1990 yılında lisans öğrenimimi tamamlayıp aynı yıl bölümde yüksek lisansa başladım ve Ç.Ü. İskenderun Meslek Yüksek Okulu makine bölümünde 1 yıl ücretli öğretim görevlisi olarak çalıştım. 1992 yılında askerliğimi asker-öğretmen olarak yaptıktan sonra 5 yıl Şanlıurfa Davut Zeki Akpınar Lisesinde matematik öğretmenliği yaptım. 1998 yılında Adana Anafartalar lisesine atandım. Halen orada göreve devam etmekteyim. 1994 yılında, mesai arkadaşım olan sevgili Ertuğrul UYGUN ile beraber "Matematiğin Renkli Dünyası" adlı , Fraktal geometrinin ürünleri olan resimlerden oluşan iki farklı salonda sergi düzenledim. Evli ve üç çocuk babasıyım.