

TÜRK MATEMATİK DERNEĞİ YAYINLARI

Sayı : 22

B. A. TRAKHTENBROT

**ALGORİTMALAR VE OTOMATİK
HESAP MAKİNALARI**

Ceviren:

Talât TUNCER

İSTANBUL — 1964

TÜRK MATEMATİK DERNEĞİ YAYINLARI

Sayı : 22

B. A. TRAKHTENBROT

**ALGORİTMALAR VE OTOMATİK
HESAP MAKİNALARI**

Çeviren:

Talât TUNCER

İSTANBUL — 1964

BU KİTAP HAKKINDA

Algoritmalar ilk çağdan itibaren bilinip kullanılmakla beraber, algoritma teorisi hemen tamamıyla yirminci asırda gelişmiştir. Teoriye esas olan algoritmik olarak çözülebilme veya çözülememe meselesidir. Yazar **algoritma** terimini tam tamına tarif ederken algoritmalarla hesap makinaları arasındaki sıkı bağı gözönüne almıştır. Gerçekten, birçok problemlerin algoritmik çözümezliğini, **Turing makinası** denilen özel tipten bir hesap makinası kullanarak ispat etmiştir.

Okuyucu kitabı okuyabilmek için, orta derecede bir cebirin ötesinde, matematiğin öbür dallarına ihtiyaç duymayacaktır. Bununla beraber oldukça karışık bir lojik düşünce dizisi izlemek zorundadır.

İncelenen birçok konuların referansları bibliyografya kısmında verilmiştir.

BU YAYINLAR HAKKINDA

Matematiğin kendi değeri yanında, fizik, kimya ve dolayısıyla mühendislik, askerlik gibi pratik sahalara ve bilhassa son zamanlarda biyoloji, ekonomi ve hattâ sosyal bilimlere yardımı hızla arttığından, bu bilim her millet için hayati bir önem kazanmıştır.

Öteyandan, matematik de bu bilimlerin problemlerini çözebilmek için gerek metot, gerekse fikir bakımından gelişmek zorundadır.

Bundan dolayı bir çok memleketlerde bu sahaya daha fazla sayıda istidatları çekmek ve bunları erkenden keşfetmek, en nihayet bunların eğitimi için her türlü fedakârlığa katlanmak en önemli bir millî eğitim siyaseti olmuştur.

Yeni istidatları erkenden keşfetmek için düşünülen tedbirlerin başında, matematik kültürünü geniş kitlelere yaymak gelir.

İkinci Dünya Harbinden sonra birçok memleketlerde, gençlerin tecessüslerini tahrik etmek ve bunların matematik bilimlerine karşı ilgilerini arttırmak için yeni bir tip matematik literatürü meydana gelmiştir. Bu çeşit literatürde aranılan özellikler kısaca şunlar olmalıdır: a) Problem vaz'ı sun'î olmamalı, b) Bunları anlamak için fazla ön bilgiye ihtiyaç bulunmamalı, c) Okuyucuyu aktif işbirliğine ve bir şeyler keşfetmeğe sevketmeli.

İşte Türk Matematik Derneği bu cereyanı memleketimize de getirmek maksadiyle bu yayınlara başlamış bulunmaktadır. Bu yayınlar, resmî müfredata bağlı ders veya yardımcı kitaplar olmayıp, konuları yukarıki prensiplere uygun olarak

seçilmiş eserlerdir. Bunların anlaşılması için lise matematiğinin bir kısmı ile okuyucunun sağduyusu ve iyi niyeti kâfidir.

Tamamen hizmet olan bu teşebbüsümüzün mali kaynağı, Milli Eğitim Bakanlığımızın ve Ford Foundation'nun bağışlarıdır.

Türk Matematik Derneği

İÇİNDEKİLER

GİRİŞ	1
Bölüm 1. Sayısal Algoritmalar	5
1. Euclide Algoritması	6
2. Sayısal Algoritmalar	7
3. Diophant Denklemleri	9
Bölüm 2. Oyunlar İçin Algoritmalar	12
4. «Onbir kibrit» oyunu	12
5. «Çift kazanma» oyunu	14
6. Bir oyunun ağacı	15
7. Kazanma stratejisine ait algoritma	19
Bölüm 3. Bir Lâbirentte Yol Bulmağa Yarıyan Bir Algoritma	25
8. Lâbirentler	25
9. Lâbirent algoritması	27
10. Lâbirent algoritmasının ispatı	31
Bölüm 4. Kelime Problemi	35
11. Asosyatif hesaplar	35
12. Kelime denkliği problemi	37
13. Kelime problemleri ve lâbirentler	39
14. Algoritmaların kurulması	40
15. Bir karenin otomorfizmaları	47
Bölüm 5. Otomatik Kontrollü Hesap Makinaları	52
16. İnsan tarafından hesaplama	52
17. Hesap makinaları	54
18. Makina ihtarları	56
Bölüm 6. Programlar (Makina Algoritmaları)	60
19. Lineer denklemler için bir program	60
20. İterasyon	62
21. Euclide'sel algoritma	66
22. Hesap makinalarının işlemesi	67
23. Hesap makinalarının kullanılışı	68

Bölüm 7. «Algoritma» nın Daha Kesin Tarifi Gereği	71
24. Algoritmaların varlığı	71
25. Sonuç çıkarılma (istidlâl) problemi	73
26. «Algoritma» tarifinin formülleştirilmesi	77
Bölüm 8. Turing Makinası	80
27. Turing makinalarının tarifi	80
28. Turing makinasının işlemesi	85
Bölüm 9. Turing Makinalarında Algoritmaların Gerçeklenişi .	89
29. Ondalık gösteriliste n yi $n + 1$ e çevirme	89
30. Ondalık gösterilis haline değiştirme	91
31. Toplama	94
32. Mükerrer toplama ve çarpma	96
33. Euclide'sel algoritma	97
34. Algoritmaların terkibi	101
Bölüm 10. Algoritmalar Teorisinin Temel Hipotezleri	104
35. Temel hipotezler	104
36. Tarihi hatırlatmalar	106
Bölüm 11. Genel Turing Makinası	108
37. Taklit algoritması	108
38. Genel Turing makinası	109
39. Şifreli semboller gurubu	111
40. Genel Turing makinası için algoritma	113
Bölüm 12. Algoritma ile Çözülemyen Problemler	115
41. Church teoremi	115
42. Kendi kendine hesaplanabilme problemi	116
43. Örtme problemleri; çevrilebilme problemi	118
44. Tarihi açıklamalar	119
Bölüm 13. Genel Kelime Problemine Ait Bir Algoritmanın Mevcut Olmaması	123
45. Çevrilebilirliğin tayinine ait bir algoritmanın mevcut olmayışı	124
46. Kelime denkliği probleminin çözülemeliği	131
SONUÇLAYICI AÇIKLAMALAR	134
BİBLİYOGRAFYA	136

GİRİŞ

Şimdi bir çok nevi matematik ve lojik problemlerin çözümünde kullanmakta olduğumuz otomatik yüksek hız hesap makinalarının gelişmesinde, 1948 denberi büyük ilerlemeler kaydedilmiştir. Bu makinaları önceki hesap makinalarından ayırdeden özellik otomatik kontroldur. Bir kere bir makinaya ilk veriler ve program uygulanıp ta makina işletilince son cevaba kadar hiç kimse karışmadan makina çalışır. Modern elektronik makinaların üretimi çok büyüktür. Bunlar bir saniyede 20.000 kadar aritmetik işlem yapar *, ki bütün bir günde iyi bir hesap makinası başında becerikli bir operatörün yapabileceğinden çok daha fazladır.

Otomatik makinaların uygulama alanı gittikçe artmaktadır: bu makinalar karışık denklem sistemlerini çözmekte, bir dilden öbürüne çevriler yapmakta, satranç oynamaktadır ilâ. Bunlar büyük fabrikalarda teknolojik işemleri kontrol etmede kullanılmaktadır. Keza, denel verilerin çarçabuk ve doğru şekilde analiz edilebilmesi keyfiyeti, bazı ilmi alanlarda şimdiye kadar pratik olmayan araştırma metotlarının kullanılmasını mümkün kılmaktadır. Genel olarak, otomatik hesap makinaları bir takım götürü işlerden tamamen kurtulmuş kimseler için kuvvetli bir âlet olarak şimdiden bilinmektedir.

Fakat bu gerçek başarılar, bütün kuvvet makinaları hakkında yapılan hayali tahminlerin ve mevcut olmayan aldatici şeylerin ortaya çıkmasına sebep olmuştur. Özel-

*) Bu sayı devamlı surette artmaktadır.

likle her hangi bir problemi çözebilen ve böylece yaratıcı araştırma yerine geçen otomatların ve «dev elektronik beyin» lerin hikâyesini zikretmemiz gerekir. Bu durum karşısında, hesap makinaları ile gerçekte ne *yapılabılmıştır* sorusu pek önemli ve sıralı olur.

Bu soru, matematik lojiğin önemli bir dalı olan modern algoritma teorisinde bir görüş noktasından cevaplandırılmıştır.

Matematik lojik, «hesaplama işlemi», «matematik ispat» ve «algoritma» gibi kavramaların tabiatını inceler. Modern elektronik makinaların keşfinden bir kaç yıl önce, «algoritma» nın tarifi ve otomatik hesap makinasına ait genel bir şema (Turing makinası, bölüm 8, 1936 da algoritma teorisine hakkında bir yazıda * açıklanmıştır) yapılmıştır, ve algoritmalarla makinalar arasındaki sıkı bağlantı açığa çıkarılmıştır. Bu, otomatik makinalar üzerine bir dizi önemli teoremler için temel teşkil etmiştir. Bu kitapta bunlardan bazıları üzerinde incelemeler yapacağız.

Bölüm 1-4 te bir algoritmanın nasıl olduğunu açıklayacak ve bazı matematik ve lojik problem sınıflarının çözümüne ait algoritmalar kuracağız.

Bölüm 5-6 da, elektronik hesap makinasının kuruluşunun prensiplerini, hem de *programlamayı* (makina ile hesaplanmak üzere algoritmalar teşkili) tartışacağız.

Bölüm 7-13 te, Turing makinasının esası üzerine dayanan algoritmalar teorisinden bir çok önemli hususlar göstereceğiz.

Bazı ispatların uzun oluşu onları bu hacimde bir kitaba almağa imkân vermemektedir, ama tam teferrüatın bu eksikliği konunun genel çizgilerini kavramayı kolaylaştıracaktır.

Modern otomatik makinalara *elektronik* adı verilmiştir, çünkü bunların yapılmasında ana eleman elektron

*) Kitabın sonundaki bibliyografyaya bakınız.

tüpü (veya daha yenilerde tranzistör) dür. Elektronik tekniğin kullanılması makina ile yapılan kişisel işlemlerde büyük bir zaman kazanılmasını sağlamaktadır. Bununla beraber, bu makinaların temel özelliği — otomatik kontrol — elektronların kullanımına bağlı değildir. Esas itibariyle, elektronik makinaların çözebildiği problemlerin aynısını (daha yavaş olmakla beraber) çözebilen otomatik bir hesap makinası yapılabilir; modern hesap makinalarının gelişmesini sadece elektronik ilmindeki ilerlemelerin sonucu olarak gözüne alamayız. Gerçekten, otomatik hesap makinalarına ait ilk şema * (Turing makinası) bir mekanik sisteme bağlı olarak verilmişti.

*) *Cevirenin notu:* Yazarın açıkça Charles Babbage'nin *Analytical Engine*'ndan haberi bulunmadığı anlaşıyor, bu makina her ne kadar imâl edilmemişse de ilkin 1830 da düşünülmüştür. Babbage 1000 kelimelik bir muhtıradan ibaret bir mekanik cihaz, bir aritmetik takım, bir kontrol takımı, bir giriş-çıkış cihazı tasavvur etti. Bu makina programı, zimbalanmış bir kartlar yığını olarak ihtiva etmekteydi.

1. SAYISAL ALGORİTMALAR

Algoritma kavramı matematiğin en temel kavramlarından biridir. *Algoritma* ile, verilen tipte herhangi bir problemin çözümünü verecek olan bir işlemler dizisini belirten bir ihtarlar listesi anlaşılır. Tabii bu, deyimın kesin matematik tarifi değildir, ama böyle bir tarifi anlamını vermektedir. Bu tarif tabii olarak doğmuş olup eski çağlardan beri kullanılmış olan algoritma kavramını aksettirmektedir.

En basit algoritmalar ondalık şekilde yazılmış sayılarla ilişkin dört aritmetik işlemin yapılmasına ait kurallardır. «Algoritma» deyimini bu kuralları dokuzuncu asır gibi erken çağda veren ortaçağ özbek matematikçisi, Elharzemî'nin adından gelmektedir*. Meselâ, iki çok rakkamlı sayının toplamı, her biri sadece iki rakam ihtiva eden (bunun biri üzerine, bir önceki işlemde çıkarılanı göstermek üzere çizgi çekilebilir) bir dizi basit işlemlerden ibarettir. Bu işlemler iki şekildedir: 1) karşılıklı rakkamların toplamını kaydetme, 2) birinden çıkarılanı sola işaretleme. İhtarlar bu işlemlerin yapılmasına ait gerçek sırayı (sağdan sola) verir. Basit işlemler sırf formel olup, herhangi bir özel probleme başvurulmadan, ilk ve son defa olarak kaydedilebilen bir toplama cetveli kullanılarak kendi kendine yapılabilir.

*) Şimdi kullandığımız rakkamların esas olan hint rakamlarıyla yazılmış ilk hesap kitabı ünlü Türk Matematikçisi *Ebu Abdullah bin Musa Elharzemî'nin Kitabü fi Hesabülhindî'sidir*. Daha önce yazdığı *Kitabül muhtasar fi hesabülcebîr velmukavele* Avrupaya *Algebra* ve *Almucabala* şeklinde geçmiştir ki *Cebîr* ve *Algebra* kelimeleri oradan gelmektedir. (Ceviren).

Öbür üç aritmetik işlemi için, kare kök v.s. almak için durum aynıdır. Bunlara karşılık gelen ihtarların (algoritmalar) formel karakteri özellikle kare kök almaya ait işlemde açıkça bellidir.

1. EUCİLİDE ALGORİTMASI

Daha güç bir misâl olmak üzere aşağıdaki tipten bütün problemlerin çözümünde kullanılan Euclide *algoritmasını* gözönüne alacağız.

Pozitif iki a ve b tam sayıları verildiğine göre bunların en büyük ortak bölenlerini bulmak. Aşikâr olarak pozitif a ve b farklı tam sayı çiftleri mevcut olduğuna göre bir çok farklı problem de mevcuttur. Bu problemlerden herhangi biri, verilen iki sayıdan birincisi büyük olanı, ikincisi küçük olanı bulunmak üzere azalan bir sayılar dizisi kurmakla çözülebilir. Üçüncü sayı birincinin ikincisi ile bölünmesinden elde edilen kalandır, dördüncü sayı ikincinin üçüncüye bölünmesinden elde edilen kalandır, ilâ. Bu işleme bölmelerden biri kalan vermeyinciye kadar devam edilir. Bu son bölmedeki kalan o takdirde aranan sayıdır.

Bölmeler tekrarlanan çıkarmalara getirilebildiğinden, böyle herhangi bir probleme ait algoritma aşağıdaki ihtarlar listesinden herhangi birinin şekline konulabilir.

Ihtar 1. a , b sayı çiftini gözönüne alınız. Bundan sonraki ihtara geçiniz.

Ihtar 2. Gözönündeki iki sayıyı karşılaştırınız (yani birincinin ikinciye eşit, bundan büyük veya bundan küçük olup olmadığını tayin ediniz). Bundan sonraki ihtara geçiniz.

Ihtar 3. Sayılar eşitse, o takdirde bunların her biri

aranan sonuçtur, hesaplama son bulur. Eğer eşit değilse bundan sonraki ihtarla geçiniz.

Ihtar 4. Birinci sayı ikinciden küçükse bunları bir-biri ile değiştiriniz ve bundan sonraki ihtarla geçiniz.

Ihtar 5. İkinci sayıyı birinciden çıkarınız ve gözönündeki iki sayı yerine sırayla çıkarmadan elde edileni ve kalamı koyunuz. Ihtar 2'ye geçiniz.

Böylece beş ihtarın hepsini yaptıktan sonra tekrar ikinci ihtarla, sonra üçüncü, sonra dördüncü, sonra beşinciye geçiniz, sonra tekrar bir defa daha ikinciye, üçüncüye ilâ dönünüz, ihtar 3 teki verilen şartlarla karşılaşınca ya yani gözönüne alınan iki sayı eşit oluncaya kadar devam ediniz. Bu olunca problem çözülmüş ve hesaplama sona ermiş olur.

Algoritmaların daima bu titiz formalitelerle ortaya konmayışı gerçek iken bilinen herhangi bir algoritmanın bu formel görünüşle ortaya konuşunun mümkün olmasına dair kimsenin aklında şüphe kalmaz.

Euclide algoritmasına ait ihtarlarda, metodun kurulduğu temel işlemler, çıkarma, karşılaştırma ve iki sayıyı birbiriyle değiştirme işlemleridir. Kolayca görülür ki bu ihtarlar daha fazla parçalara ayrılabilir, meselâ, ihtar 5, sayılardan birinin diğerinden çıkarılmasına ait ayrı bir algoritma halinde genişletilebilir. Bununla beraber, böyle hallerde aritmetik işlemlerine ait kurallar pek basit ve alışılmış durumda olduğundan, algoritmayı çok teferrüatlı olarak anlatmak gereksizdir.

2. SAYISAL ALGORİTMALAR

Algoritmalar, *sayısal algoritmalar* adı verilen dört aritmetik işleminin kullanılışı üzerine kurulmuştur *. Bun-

*) Bu deyimın kesin anlamı olmadığı hatırdâ tutulmalıdır.

lar hem elemanter hem de yüksek matematikte önemli bir rol oynar ve çoğu defa sözlü ihtarlar veya çeşitli neviden formüller ve şemalarla verilir. Meselâ

$$a_1x + b_1y = c_1$$

$$a_2x + b_2y = c_2$$

iki bilinmiyenli denklem sisteminin çözülmesinde kullanılan bir algoritma

$$x = \frac{c_1b_2 - c_2b_1}{a_1b_2 - a_2b_1}, \quad y = \frac{a_1c_2 - a_2c_1}{a_1b_2 - a_2b_1}$$

formülleriyle verilir ki burada işlemler, keza bunların sıraları tamamen belirlenmiştir. Formüller, verilen tipte (yani $a_1b_2 - a_2b_1 \neq 0$ olmak şartıyla $a_1, a_2, b_1, b_2, c_1, c_2$ katsayıları herhangi sayılar) bütün problemler için aynı işlem zincirini verir.

Bununla beraber şunu söylemek ilgi çekicidir ki, genel olarak özel bir problemin çözülmesinde kullanılması gereken işlemler sayısından söz edince bu sayı daha önce bilinmiyen demektir, bu sayı özel probleme bağlı olup sadece algoritma yapıldığı sırada ortaya çıkmıştır. Bu Euclide algoritması halidir ki burada gerekli çıkarmaların sayısı a ve b sayılarının özel seçilişine bağlıdır.

Öbür bir çok işlemler dört hesap işlemine getirildiğinden sayısal algoritmalar geniş şekilde kullanılmıştır. Çoğu defa böyle bir getiriş, tam bir cevap vermez, istenen herhangi bir doğrulukta bir cevap verir. Buna bir kare kök almaya ait algoritma ile misâl verilir. Bir dizi bölme, çarpma ve çıkarma ile, bir kök istenen doğrulukla hesap edilebilir. Matematğin özel bir dalında (*sayısal analiz*) benzer metotlar, integrasyon, türev alma ve türlü neviden

denklemlerin çözülmesi gibi daha karışık işlemlerin aritmetik işlemlerine getirilmesi geliştirilmektedir.

Matematikte, *bir problem sınıfı bunların çözülmesine ait bir algoritma bulununca çözülmüş sayılmaktadır*. Böyle algoritmaların bulunması matematiğin tabii amacıdır. Meselâ, cebirsel bir denklemin köklerinin sayısı (ve katlılığı) nın tayinine ve gösterilen herhangi bir doğruluk derecesinde köklerin hesaplanmasına ait algoritmalar bulunmuştur.

Verilen tipteki bütün problemleri çözmek için bir algoritma yoksa, matematikçi bu tipten bazı problemleri çözen bir metot, — her ne kadar öbür hallere uygulanamazsa da — meydana getirmeye zorunlu olabilir.

3. DIOPHANT DENKLEMLERİ

Bugünkü matematiğin bir algoritmayı haiz olmadığı bir problem sınıfına ait misâl olmak üzere mümkün bütün Diophant denklemlerini gözönüne alalım. Yani P , katsayıları tam sayı olan bir polinom olmak üzere

$$P = 0$$

denkleminin tam sayılı çözümlerini arıyalım *. Böyle denklemlere ait misâller

$$x^2 + y^2 - z^2 = 0$$

$$6x^{18} - x + 3 = 0$$

dır. Birincisi üç bilinmiyenli bir denklem, ikincisi ise bir bilinmiyenli bir denklemdir. (Genel olarak herhangi bir

*) Diophant denklemlerinin bazan başka tarifleri verilmektedir.

sayıda bilinmiyenli denklemleri gözönüne alabiliriz.) Böylece yukardaki ilk denklem

$$x = 3, \quad y = 4, \quad z = 5$$

tam sayılı çözümlerini haizdir. Fakat ikincisinin, herhangi bir x tam sayısı için

$$6x^{18} > x - 3$$

olduğu kolayca gösterildiğinden tam çözümleri yoktur.

1901 de, Pariste uluslararası bir matematik kongresinde ünlü Alman matematikçisi David Hilbert çözülmüş yirmi problemten ibaret bir liste sunmuş ve matematikçi topluluğunun dikkatini bunların çözülmesinin önemine çekmiştir. Bunlar arasında şu vardı (Hilbert'in Onuncu Problemi): *Verilen herhangi bir Diophant denkleminin tam sayılı çözümünün olup olmadığının bulunması.*

Bir bilinmiyenli Diophant denklemlerinin özel çözümüne ait böyle bir algoritma bilinmektedir. Eğer katsayıları tam sayı olan bir

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0$$

denkleminin tam bir x_0 çözümü varsa, o takdirde a_0 , x_0 ile bölünebilir. Bu husus aşağıdaki algoritmanın gözönüne alınmasını gerektirir.

1) a_0 sayısının bütün bölenlerini bulunuz (bunlardan sadece sonlu bir sayıda vardır, ve bunların bulunmasına ait bir algoritma mevcuttur).

2) Bunların herbirini sıra ile denklemin sol yanına koyunuz ve elde edilecek değeri hesaplayınız.

3) Eğer bölenlerden herhangi biri sol yan için bir sıfır değeri verirse, o takdirde bu bölen denklemin bir

çözümüdür; eğer bölenlerin hiç biri sıfır değerini vermezse, o takdirde denklemin tam sayılı çözümü yoktur.

Hilbert problemi bir çok ünlü matematikçiye etki etmiş ve edegelmış, fakat iki veya daha fazla bilinmeyenli genel hal için aranan algoritma henüz bulunamamıştır. Üstelik, şimdi anlaşıyor ki pek muhtemel olarak böyle bir algoritma hiç bir zaman bulunmayacaktır. Kötümser görünen bu tahminin tam anlamı ilerde okuyucuya açıklanacaktır (Bölüm 44).

Şimdiye kadar verilen misâllerde aşikâr ki sayısal algoritmalar (ve gerçekten, genel olarak algoritmalar) aşağıdaki özellikleri haizdir:

Algoritmaların belirleyici tabiatı. Bir algoritmanın, hesaplamanın her adımında takip edilecek tam yolu veren sonlu sayıda bir ihtarlar listesi halinde verilmesi gerekir. Böylece, hesaplama, hesaplıyana bağlı olmaz; herhangi bir anda ve herhangi biri tarafından arka arkaya tekrarlanabilen belirleyici metottur.

Algoritmaların genel oluşu. Bir algoritma, herhangi bir ilk veride yapılabilen ve her bir halde doğru sonucu veren bir hesaplamayı tarif eden bir *tek* ihtarlar listesidir. Başka deyişle, bir algoritma sadece bir özel problemin değil, fakat benzer problemlerden ibaret bütün bir sınıfın nasıl çözüleceğini anlatır.

2. OYUNLAR İÇİN ALGORİTMALAR

Önceki bölümde gözönüne alınan misâller aritmetik, cebir ve sayılar teorisinden alınmıştır. Bunlar matematiğin ve genel olarak klâsik matematiğin bu kollarının çok tipik problemlerindendir. Bu ve bundan sonraki bölümlerde biraz farklı karakter taşıyan problem sınıflarını inceleyeceğiz; her ne kadar böyle *lojik* problemlerle âdi *matematik* problemler arasında kesin bir çizgi çizmek güçse de bunlara matematik problemler demektense *lojik* problem demek daha yerinde olur. Bu çizginin nereden çizildiğine ve onun hangi kenarı üzerinde bulunduğumuza bakmadan, ödevimiz benzer problemlerden ibaret bazı sınıflardaki herhangi bir problemi çözmekte kullanılan bir tek metot veren bir algoritmanın bulunması olarak kalır, aradaki biricik fark, burada gözönüne alınan hallerde algoritmalar artık sayısal olmaktan çıkar.

4. «ONBİR KIBRİT» OYUNU

Şans keyfiyetinin sonucuna bağlı olmayıp oyuncunun hünerine bağlı bir çok oyunlardan * biri de «Onbir kibrit» oyunudur.

*) Oyun kelimesinin, şu misâllerde de belirtildiği gibi iki anlamı vardır. «Satranç insanı düşündüren bir oyundur» ve «bugün oynadığımız satrançta dört oyun yaptım». Bunlardan birincisi için *oyun (game)* deyimini alıkoyacağız. İkincisi için *düzen (play)* kelimesini kullanacağız. Bir oyunun bir düzeninde oyuncunun yaptığı kişisel karara bu düzenin bir *hareketi* denir. (Ceviren)

Onbir eşya, meselâ kibrit, masa üzerinde durmaktadır. İlk oyuncu 1, 2 veya 3 kibrit alıyor. Sonra ikinci oyuncu kalan kibritlerden 1, 2 veya 3 ünü alıyor. Daha sonra yine ilk oyuncu, 1, 2 veya 3 kibrit almağa koyuluyor. Oyuncular hiç kibrit kalmayıncaya kadar bu işi yapmağa devam ediyorlar. Son kibriti almak zorunda kalan oyuncu oyunu *kaybedendir*. İlk turu yapan A oyuncusunun rakibi B oyuncusunu daima son kibriti almak sorunda bırakabil-diği bir şema mevcut mudur?

Oyunun incelenmesi göstermektedir ki aşağıdaki ihtarlar takip edilirse A, B yi son kibriti almak zorunda bırakabilir.

1. *Birinci hareket.* A iki kibrit alır.
2. *Daha sonraki hareketler.* Eğer B son hareketinde l ($l \leq 3$) kibrit alırsa, o takdirde A, $4 - l$ kibrit alır.

Bu ihtarlar listesinin şu anlamda tam olduğu gösterilebilir ki rakibinin ne yaptığına bakmadan, liste A'nın yapabildiği biricik hareketi belirtir.

Böyle tam ihtarlar listesine oyunlar teorisinde *strateji* denir. Eğer A oyuncusunun bir kaç strateji kullanınca mutlaka kazanması gerekirse, buna *A ya ait kazanma stratejisi* denir.

A için verilen strateji gerçekte bir kazanma stratejisidir, çünkü B'nin ne yaptığına bakmadan A ya yenmeyi mümkün kılar. Bu aşağıdaki iki misâlde gösterilmiştir:

A B A B A B
2 2 2 1 3 1

A B A B A B
2 3 1 1 3 1.

Bundan başka A, ilk harekette ya bir veya üç kibrit alsaydı o takdirde B yi muhakkak kazandıran bir stratejinin olabildiği gösterilebilirdi.

5. «ÇİFT KAZANMA» OYUNU

Şimdi «çift kazanma» oyununu gözönüne alalım. Oyun bir masa üzerinde 27 kibritle başlar. Oyuncular değişmek suretiyle bir kibritten dört kibrite kadar alırlar. Bütün kibritler alınınca çift sayıda kibrit alan oyunu kazanır.

İlk başlayan A oyuncusuna ait kazanma stratejisi aşağıda gösterilmiştir:

1. *Birinci hareket.* A iki kibrit alır.
2. *Öbür oyuncu B, çift sayıda kibrite malikken daha sonraki hareketler.* Henüz masa üzerinde duran kibritlerin sayısını altı ile bölmekle elde edilen kalan r olsun. Eğer $r = 2, 3, 4$ veya 5 ise o takdirde A (sırasıyla) 1, 2, 3 veya 4 kibrit alır.
3. *B nin tek sayıda kibriti olduğunda daha sonraki hareketler.* $r = 0, 1, 2$ veya 3 ise masada en az dört kibrit varsa, o takdirde A bunların hepsini alır.

Meselâ, A nın bu stratejiyi kullanarak yendiği aşağıdaki oyunu gözönüne alalım:

A	B	A	B	A	B	A	B	A	B	A	B
2	1	1	3	1	3	4	1	4	2	4	1.

Önceki oyunda olduğu gibi A ya ait bir kazanma stratejisi bulduğumuz yolun bir açıklamasını yapacak ve sadece böyle bir stratejiyi sunmakla yetineceğiz. Şunu kaydedelim ki düşünceler, söz konusu oyunun teferruatına büyük ölçüde bağlıdır ve epeyce hüner ve beceriklilik ister.

6. BİR OYUNUN AĞACI

İlk amacımız, bir hayli büyük sınıftaki her oyuna ait bir «en iyi» strateji verecek olan bir algoritma bulmaktan ibarettir. Şekli karışıklıklardan kaçınmak için, kullanılan kavramların tam tariflerini verecek yerde, ara sıra sadece açıklayıp misâller vermek suretiyle onları anlatacağız.

Anlatılan iki oyunun önce aşağıdaki özelliklerini kaydedelim:

1. Oyun, hareket yapmakta sıra değiştiren iki oyuncu ile oynanır.

2. Oyun tam şu iki sonucun biriyle biter:

a) Ya ilk hareketi yapan oyuncu, A kazanır (bu sonuç aşağıda «+» ile gösterildi.) veya b) öteki oyuncu B kazanır («-» ile gösterildi).

3. Her bir hareket, uygun bir hareket takımının *oyuncu tarafından seçilmesinden* ibarettir (kaydedelim ki seçim meselâ zar atmak gibi bir kaç şans olayının sonucu olmaktan ziyade oyuncu tarafından verilen karardır).

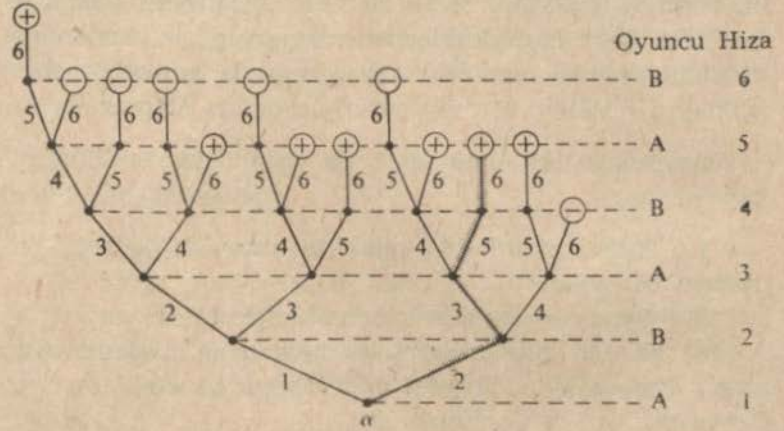
4. Oyunun herhangi bir yerinde her iki oyuncu da daha önce ne hareketler yapıldığına ve daha neler yapabileceğine dair tam bilgiye sahiptirler.

5. Bir oyunda yapılan hareketlerin sayıları için bir üst sınır vardır.

Bundan böyle ilk önce, söz konusu her oyunun 1-5 özelliklerine malik olduğunu farzedeceğiz.

Her iki oyuncunun birden bir kazanma stratejisine sahip olamayacakları açık ve ortada olan bir husustur. Daha az açık olanı her oyunda oyunculardan biri için bir kazanma stratejisinin mevcut olduğudur. Bu farzın isbatına gelmeden önce bir oyunun *ağaç* şeklinde uygun bir grafik gösterilişi nasıl haiz olabileceğini göstereceğiz.

Aşağıdaki oyuna («Onbir kibrit» in sadeleştirilmiş bir şekli) karşılık gelen ağaç Şek. 1 de resimlendirilmiştir: bir masa üzerinde altı kibrit var; her bir oyuncu sıra ile bir veya iki kibrit alır. Oyunu kaybeden son kibriti alandır.



Ağacın *tepeleri* oyunun bir düzeninde vâki olabilen değişik durumları göstermektedir. Bir köşeden çıkan *dallar* oyuncunun yapabileceği mümkün olan seçimleri göstermektedir.

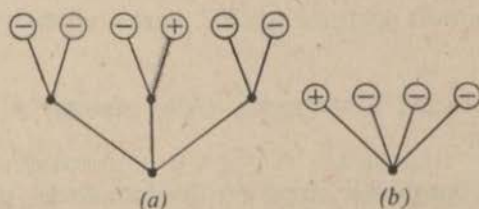
Misâlimizde her bir hareket için iki imkân (sonuncusu hariç) mevcuttur. Kesin olsun diye herhangi bir tepeden bir kibrit almağa karşılık gelen sola bir dal ve iki kibrit almağa karşılık gelen sağa bir dal çıkarıyoruz. Oyunun bir düzeni ağacın α alt tepesini (ağacın tabanı) bir uç tepeye (yani hiç dalın çıkmadığı bir tepe) birleştiren kırık bir çizgiye karşılık gelir. Düzenin sonu her bir uç tepeye işaretlenmiştir.

Şekil 1 de her bir dal, düzenin bu adımında toplanmış bulunan kibritlerin toplam sayısı ile işaretlenmiştir. Taralı çizgi A'nın kazandığı

A	B	A	B
2	1	2	1

düzenini göstermektedir.

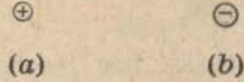
Uç tepelerden başka her tepeden bir *hıza çizgisi* geçiriyoruz. (Bak. Şekil: 1) Bir ağaçta vâki olan en üst *hıza çizgisi*ne ağacın *basamağı* denir, basamak, verilen oyunun bir düzeninin en büyük uzunluğuna eşittir (beşinci özellik böylece, gözönüne alınan bütün düzenlerin sonlu bir *basamağı* olduğunu farzetmektedir). Tek *hıza çizgili* tepeler A'nın hareketi olan durumlara, çift *hıza çizgili*ler ise B'nin hareketi olan durumlara karşılık gelmektedir.



Sekil: 2

Buraya kadar ağacı, sadece kuralları başka bir şekilde daha önce verilmiş bir oyunun bir nevi grafik resmi olarak gözönüne aldık. Bununla beraber, bir ağacı bir oyunun tarifi olarak almaktan bizi kimse alıkoymaz. Meselâ, şekil: 2a, her düzenin tam iki hareketten ibaret bulunduğu bir oyunu tarif etmektedir: A, mümkün üç hareketten birini yapmakla başlıyor ve bundan sonra B ise mümkün iki hareketten birini yapıyor. Bu oyunda A'nın yendiği mümkün bir tek düzen bulunmaktadır (taralı kırık çizgi). Şek. 2b'deki ağaç bir tek hareketle oynanan bir oyunu tarif etmektedir. Çeşitli sebeplerden, Şek. 3a ve 3b'de gösterildiği gibi «hareketlerden» değil «oyunlardan söz-

mek te faydalı olurdu. Bu «oyunlarda» oyuncuların hiç biri bir şey yapmaz; biri kendiliğinden kazananı bildirir.



Şekil: 3

Her bir (uç tepe olmıyan) tepeye, kendisi herhangi bir oyuna karşılık gelen daha alçak basamaktan bir «alt ağacın» tabanı olarak bakılabilir.

Bir oyunun bir ağaç ile gösterilişi, bir kimseye, A ya ait herhangi bir stratejiyi, tek hiza çizgili tepeleri buna bitişik daha yüksek (çift) hiza çizgil tepelere birleştiren bir oklar sistemi şeklinde grafik olarak göstermeyi mümkün kılar.

Bir stratejiyi göstermede, oklar sistemi şu özellikleri taşımaktadır:

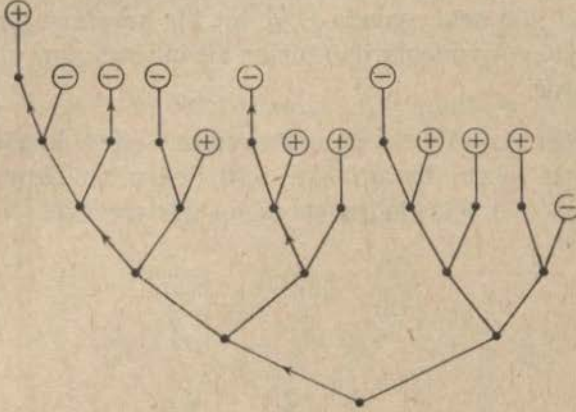
1. Herhangi bir tepeden birden fazla ok çıkmaz (yani, A oyuncusuna ait bir strateji, verilen herhangi bir durumda seçimini bir tek şekilde belirlemelidir).



Şekil: 4

2. Bir (çift hiza çizgili) γ tepesine gelen bir ok mevcutsa, o takdirde daha yüksek ilk hiza çizgili herhangi bitişik bir köşe bir okun başlangıcı olmalıdır. Bu şart,

B nin ne yaptığına bakmaksızın A ya ait bir hareket verecek stratejinin varlığını güven altına alır. B ye ait strateji benzer şekilde tarif edilir.



Şekil: 5

A nın daima tam bir kibrit aldığı «altı kibrit» oyununda A ya ait strateji Şekil 5 te gösterilmiştir. Bu strateji A nın kaybettiği 2 ve kazandığı 2 düzen sağlamaktadır.

7. KAZANMA STRATEJİSİNE AİT ALGORİTMA

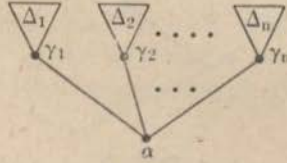
TEOREM : 1-5 şartlarını gerçekliyen herhangi bir oyunda, oyuncularından birine ait bir kazanma stratejisi vardır.

Bu teoremin isbatı, herhangi bir oyunda oyuncularından birine bir kazanma stratejisi sağlayacak bir algoritmanın açıklanmasından ibaret olacaktır. Algoritma, oyundaki mümkün en uzun düzenin uzunluğu olan v üzerine indüksiyonla kurulur * (v , oyunun ağacının basamağıdır).

*) Bu seriden yayımlanmış I.S. Sominski'nin *İndüksiyon Metodu'na* bakınız.

$v=0$ hali. Bu halde bir oyun hiç bir hareketten ibaret değildir. Bu neviden sadece iki oyunun ağaçları Şekil: 3 te gösterilmiştir. Hiç bir şey yapmama «stratejisi» (yani hiç oksuz gösterileni) Şekil: 3a daki oyunda A ya ait, Şekil: 3b deki oyunda B ye ait bir kazanma stratejisidir. Böylece, oyuncuların birine ait bir kazanma stratejisi mevcuttur.

v den $v+1$ e geçiş. Teoremin v den küçük veya buna eşit bütün basamaklar için doğru olduğunu kabul edelim; $v+1$ için doğru olacağını göstereceğiz.



Şekil: 6

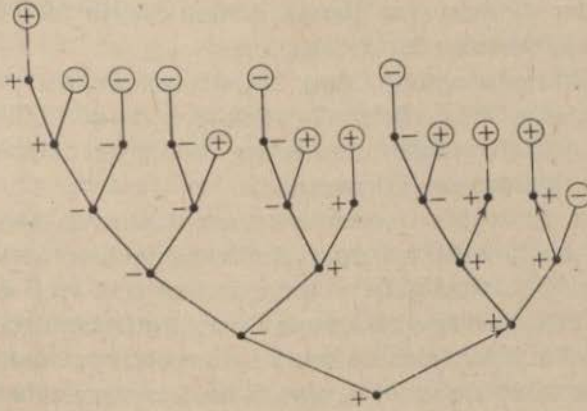
Bu halde oyunun ağacı Şekil: 6 da gösterilen şekli haizdir, burada üçgenler, tabanları bütün ağacın α tabanına komşu $\gamma_1, \gamma_2, \dots, \gamma_n$ tepeleri olan alt ağaçları temsil etmektedir. Alışıldığı gibi A yı ilk hareketi yapan oyuncu olarak alalım. O vakit $\Delta_1, \Delta_2, \dots, \Delta_n$ alt üçgenleri B nin ilk hareketi yaptığı oyunları (veya hareketsiz oyunları) gösterir; bundan başka, bu oyunların hepsi en az v basamağındadır.

İndüksiyon hipotezine göre teorem bütün v ler için doğrudur. Şayet bu oyunlardan herhangi biri meselâ Δ_1 için A nın kazanma stratejisi varsa, o takdirde bütün oyun için A nın bir kazanma stratejisi vardır: α ile γ_1 yi birleştiren oku, Δ_1 ye ait A nın kazanma stratejisine bititirmek yetmektedir. Öte yandan şayet bütün $\Delta_1, \Delta_2, \dots, \Delta_n$ alt oyunları için B nin bir kazanma stratejisi varsa, o takdirde B nin bütün oyun için kazanma strate-

jisi vardır; o, kazanma stratejisini sadece alt oyunları için birleştirir. (Yani onun kazanma stratejisi alt oyunlara ait kazanma stratejilerinin herhangi birine giren bütün okların bir topluluğu olacaktır.)

α nın, B nin durumunu gösterdiği hal tamamen aynı şekilde incelenir.

Bu, ispatı ve algoritmanın açıklanmasını sonuçlandırır. «Altı kibrit» oyunlu işlemi şekillendirelim (Şekil: 7).



Şekil: 7

Önce, diyagramın üstünden altına giderek her bir tepeye, bu tepeyi taban olarak alan alt oyuna ait A nın veya B nin kazanma stratejisini haiz oluşuna göre artı veya eksi işareti koyalım.

Altı hiza çizgisinin biricik uçsuz tepesinin artı işaretiyle işaretlenmesi gerekir. Şimdi 5 hiza çizgisinin köşelerini soldan sağa doğru gözönüne alalım. Bu hiza çizgisinde A nın hareketi bulunduğunu hatırlatalım. En soldaki tepenin artı ile işaretleneceği açıktır, çünkü 6 hiza çizgisindeki biricik tepe artı ile işaretlenmiştir. 5 hiza çizgisindeki kalan uçsuz tepelerin eksi ile işaretlenmeleri

gerekir. 4 hiza çizgisinin tepeleri arasında (burada B nin hareketi vardır) dört eksi tepe, yani 5 hiza çizgisindeki eksi tepelere bitişik olanlarla birlikte üç artı tepe buluruz.

Bu işleme devam ederek 1 hiza çizgisine ulaşırız ki bunun tek tepesi (ağacın tabanı) artı işareti almalıdır. Böylece, A bir kazanma stratejisini haizdir. Şimdi alttan üste doğru giderek ok koyalım. Tabandan 2 hiza çizgisindeki artı tepeye bir ok koyalım (misâlimizde böyle sadece bir tepe vardır). 3 hiza çizgisinde ok başına bitişik olan her tepeden bu tepeyi 4 hiza çizgisindeki bir artı tepeye birleştiren bir ok çizelim.

Misâlimizde şimdi tam bir stratejimiz var ve işlem burada sona erer (Şekil: 7). Şekilden de görüldüğü gibi, A bu stratejiyi kullanınca sadece iki düzen mümkündür, ve her ikisinde de A kazanır.

Bu bölümdeki teoremle bunun dayandığı algoritmanın, 6 bölümündeki 1 ve 2 özelliklerinin gerçekleşmediği hali de içine almak üzere (yani sadece 3, 4 ve 5 özelliklerinin doğru olduğu bir oyuna) genelleştirilebildiğine dikkat edelim. Meselâ iki A ve B oyuncusu arasındaki oyunları gözönüne alalım ki A veya B nin kazanmasıyla birlikte oyunda berabere kalmak ta mümkün olsun (meselâ bir nevi dokuz-taş oyununda olduğu gibi). Burada öyle olabilir ki oyunculardan hiç birinin kazanma stratejisi bulunmaz; bu halde algoritma her bir oyuncuya, ona en az bir, bir beraberlik (eğer karşısındaki yanlış bir hareket yaparsa kazanma) sağlayan bir strateji verir.

Öte yandan satranç oyunu 5 inci şartı sağlamaz olur *. Bununla beraber, yeni bir kural eklemekle, meselâ 40

*) Eğer uluslar arası yarışma kuralları takip edilirse bu söylenilen doğru olmaz. Satranç, aynı hareketler dizisinin arada üç defa vâki oluşunda oyunun berabere sayıldığı eski «Alman kuralı» ile oynandığı vakit 5 inci özelliğin gerçekleşmiş oluşuna dair bir ispat için, *Zeitschrift für Mathematische Logik*, Vol. 2 (1956), Sayfa 215-17 de F. Bagemihl, «Transfinitely Endless Chess» e bakınız.

hareketten sonra berabere kalınmış olunacak demekle 3, 4 ve 5 özelliklerine malik bir oyun meydana getiririz. Böylece, oyunculardan birine hiç olmazsa berabere bırakacak bir strateji sağlanmış olur. Bunu bulmak için, 40 hareketle sınırlanmış satranca ait ağacı çizmek ve en elverişli bir strateji elde etmek üzere metodu kullanmak yetiştir. Beyaz için (A oyuncusu) bir kazanma stratejisinin mevcut olduğu anlaşılırsa o takdirde oyun, bu stratejiyi sıkı sıkıya takip etmek şartıyla beyazın lehine önceden belirlenmiş olur. Benzer şekilde eğer siyahın bir kazanma stratejisi var olsaydı veya her iki oyuncunun bir berabere kalma stratejisi var olsaydı oyunun sonucu yine önceden belirlenmiş olurdu. Böylece yukarıda gözönüne alınan algoritmanın satranca uygulaması, «Altı kibrit» oyunu için yapmağa zorunlu olduğumuz gibi oyunun tam bir incelenmesine gitmemizi gerektirir.

Bununla beraber, neden satranç, büyük hüner ve marifet ister? Burada algoritma ile ihtar edilen metodun *pratik bakımdan imkânsızlığı* ile karşılaşmış olduk. Yirmi (beyaza ait mümkün ilk hareketin herbiri için bir) dal satranca ait ağacın tabanından çıkar. Yüksek hiza çizgili tepelerin herbirinden çıkan dal sayısı da genel olarak pek büyüktür. Ağacın basamağı böylece büyük olur. Bununla beraber, gözönüne alınan sınıftan (tabii, sonlu bir adım sayısında) herhangi bir oyuna ait elverişli stratejiyi bulmakta bize imkân veren tam bir ihtarlar listesine sahip olup olmadığımız sorusuna olumlu bir cevap vermemiz gerekir. Böylece bu ihtarlar listesiyle bildirilen metot *düşünüş bakımından mümkün* ama *pratik bakımdan mümkün* değildir, çünkü büyük sayıda işlemler gerekmektedir.

Önceki metodu belirli bir oyuna uygulamanın pratik oluş derecesi oyunun karışıklığına, ihtiva ettiği işlemleri yapma çabukluğumuza ve bunun üzerine harcamayı göze

aldığımız toplam zamana bağlıdır. Biz esas olarak, bittabi pratik bakımdan mümkün olan metotlarla ilgilendik. Bununla beraber, pratik olan metotlarla pratik olmıyanlar arasında kesin bir matematik kriter mevcut değildir. Pratik bakımdan hesaplamaya elveren araçlara bağlıdır, ve bu da meselâ tekniğin ilerlemesiyle değişebilir. Böylece, yüksek hız hesap makinalarının ortaya çıkışı ile şimdiye kadar imkânsız olan bir çok metotlar uygulama alanında gerçekleştirilebilir hale gelmiştir.

Yukarıda gözönüne alınan algoritma kaba ise de bunun mevcut oluşu önemli bir husustur. Zira şimdiye kadar, Diophant denklemleri hakkındaki Hilbert problemine ait *nasıl olursa olsun hiç bir algoritma* bulunmamıştır!

Yine de kaba olanı bile olsa, bir algoritmanın bulunuşu, sadeleştirilebilir veya daha çok uygun algoritma kurulabilir ümidini verebilir.

Yukarıda gözönüne alınanlardan ötürü hesabedilebilir bir metottan veya genel olarak, bir takım algoritmalar ile ihtar edilen bir metottan söz edince daima, uygulama alanında şimdiki hesap usullerimiz bunu yapmağa yeterli olmasa bile sadece istenen sonucu verebilecek bir metodu kasetmiş olacağız.

3. BİR LÂBİRENTTE YOL BULMAĞA YARIYAN BİR ALGORİTMA

8. LÂBİRENTLER

Yunan mitolojisi Minotaur azmanını bulup öldürmek için bir labirente * giren kahraman Theseus'den bahseder. Ariadne ona yardım etmiş, bir ip yumağı vermiş ve yumağın bir ucundan da tutmuştur. Theseus, lâbirentte derinlere girdikçe yumağı çözmüş, o vakit ipi tekrar dolayarak çıkış yolunu yine güvenle bulmuştur.

Bu eski menkıbeyi hatırlatan şey, Amerikan matematikçi ve mühendisi Claude Shannon'un modern «Lâbirentteki Fare» sidir **. «Fare» özel bir lâbirentin bir yerine konmuş, bir «peynir parçası» da bir başka yerine konmuştur. Fare dolambaçlı yolu bulunan lâbirenti «peynir» i buluncaya kadar dolaşmak ister. O vakit eğer gene ilk hareket ettiği yere konulursa başıboş dolaşmadan doğruca «peynir» e gider. («Fare» birinci teşebbüste «ardardaki» dönüşleri kaydeden yardımcı bir batarya devresiyle elektromanyetik olarak kontrol edilir, öyle ki ikinci geçişte «fare» sadece bunları yapar.)

Burada bir lâbirent içinden geçen bir yol bulma ile ilgili bir problem (veya, daha doğrusu, böyle bir problem

*) Lâbirent, çok fazla sayıda taşlarla meydana getirilmiş bir bina idi, taşlar öyle düzenlenmişti ki çıkış yolu ancak pek güçlükle bulunabilmekteydi, buna dolambaçlı yol denilebilir. (Çeviren)

**) C. E. Shannon. «Computers and Automata», *Proceeding of the IRE*, Cilt. 41 (1953), sayfa 1234.

duğunu ve Theseus'un onu yakalamak için Ariadne'nin beklemekte olduğu A kavşağından yola koyulduğunu farzedelim; bu halde problemi çözmemiz gerekir: A kavşağından M kavşağına geçilebilir mi? * Eğer geçilebilirse, Theseus, yürüdüğü yol ne olursa olsun Minotaur'u bulmalı ve sonra *basit bir yoldan* Ariadne'ye dönmelidir. Eğer geçilemezse, Ariadne'ye dönmesi gerekir.

Sayırsız mümkün lâbirentler vardır, ve belli bir lâbirent için A ile B nin birbirine göre konumlarına göre bir çok imkânlar olacaktır. Başlangıçta Theseus, ne lâbirentin yapısını ne de Minotaur'un bulunduğu yeri bilmektedir, problemin çözümü, herhangi bir lâbirentte ve A ile M nin birbirine göre herhangi izafi konumu için kullanılabilen genel bir araştırma metodu şeklinde olmalıdır. Başka türlü söylersek, çözüm, verilen tipten herhangi bir problemin çözümüne ait bir algoritma olmalıdır.

9. LABİRENT ALGORİTMASI

Böyle bir algoritmayı kurmak için, özel bir araştırma metodu vereceğiz. Araştırmanın her adımında koridorları üç sınıfa ayıracağız: Theseus'un hiç geçmediği (bunlara *yeşil* koridorlar diyeceğiz), bir kere geçtiği (*sarı*), ve iki kere geçtiği (*kırmızı*). Bundan başka, Theseus herhangi bir kavşaktan bir komşu kavşağa şu iki yoldan biriyle geçer:

1. *İpliği yumaktan çözerek*. Theseus, yol aldıkça, Ariadne'nin ipliğini yumaktan çözerek, yeşil bir koridor boyunca komşu bir kavşağa geçer; bu koridor o taktirde sarı sayılır.

2. *İpliği yumağa sararak*. Theseus, yol aldıkça

*) $A \neq M$ olduğunu farzetmek tabiidir.

Ariadne'nin ipliğini yumağa sararak, sarı bir koridor boyunca komşu bir kavşağa geçer; bu koridor o taktirde kırmızı sayılır.

Kaydedelim ki Theseus kırmızı bir koridordan gitmeyi iyi karşılamaz. Theseus'un daha sonra yeşil bir koridoru kırmızısından ayırabileceği bir takım işaretler koyduğunu kabul edelim. O, sarı koridorları da ayırdedebilir, zira bu koridorlar boyunca Ariadne'nin ipliği serilidir. Her bir hareketin seçimi, Theseus'un bulunduğu kavşakta karşılaştığı duruma bağlıdır. Bu durumlar aşağıdakilerden biri veya bir kaç olabilir:

1. *Minotaur*. Minotaur'un belli bir kavşakta olduğu anlaşılır.
2. *İlmik*. Ariadne'nin ipliği daha önce belli bir kavşaktan geçmiştir; başka deyişle, kavşaktan çıkan en az iki başka yeşil koridor vardır.
3. *Yeşil*. Kavşaktan çıkan en az bir yeşil koridor vardır.
4. *Ariadne*. Ariadne belli bir kavşaktadır.
5. *Beşinci hâl*. Yukarki durumların hiç biri bulunmaz.

Araştırma metodumuz artık aşağıdaki cetvel yardımıyla anlatılabilir:

<i>Durum</i>	<i>Hareket</i>
1. Minotaur	Durma
2. İlmik	Tekrar yumağı dolama
3. Yeşil	Yumağı çözme
4. Ariadne	Durma
5. Beşinci hâl	Tekrar yumağı dolama

Theseus herhangi bir kavşakta bulunmakla yapacağı hareketi şöyle kararlaştırır: 1 durumuyla başlayarak,

kavşakta vâki olmıyan bir durum bulana kadar evvelki cetvelin sol sütununu sıra numarasına göre yoklar. Bundan sonra sağ sütundaki buna karşılık olan hareketi yapar. Bir «Dur» emriyle karşılaşınca kadar hareketine devam eder.

Bu metodun uygulanışı aşağıdaki üç iddiadan doğruya çıkarılan bir sonuçtan çıkmaktadır:

1. A ile M lâbirentte nerede bulunursa bulunsun, Theseus sonunda, sonlu sayıda bir hareketten sonra ya A da ya M de bir «Dur» emriyle karşılaşmalıdır.

2. «Dur» emri M de gelirse, o taktirde Minotaur'a geçilebilir. Bununla beraber, A dan M ye basit bir yol boyunca Ariadne'nin ipliği serili olacaktır; ipliği yumağa sararak Theseus bu yol boyunca Ariadne'ye dönebilir.

3. Eğer «Dur» emri A da gelirse, o taktirde Minotaur'a *geçilemez*.

Bu iddiaları ispat etmeden önce, metodun kullanılmasına dair iki misâl vereceğiz.

MİSÂL 1. Aramanın, lâbirentin A kavşağında başladığını (Şek. 8) ve Minotaur'un F kavşağında bulunduğunu farzedelim. Metodumuzu takip eden bir arama Cetvel 1 de verilmiştir. (Yeşil koridoru geçmenin seçilmesi keyfi olduğundan; başka mümkün arama örneklerinin bulunacağı tabiidir.)

Bu misâlde görüyoruz ki Minotaur'a geçilemez. Şimdi son iki sütundan sarıyla sonlanan koridorları seçersek A dan F ye giden aşağıdaki basit yolu buluruz. AB , BC , CD , DF .

MİSÂL 2. Minotaur yine F de olmak üzere eğer arama Şek. 8 deki K kavşağında başlarsa Cetvel 2 deki araştırma örneğini elde ederiz. Bu halde Minotaur'a geçilemez.

CETVEL 1

Hareket No.	Theseus'un kavşakta karşılaştığı durum	Hareket	Seçilen koridor	Koridorun hareketten sonraki hâli
1	Yeşil	Çözme	AB	Sarı
2	»	»	BC	»
3	»	»	CD	»
4	»	»	DH	»
5	»	»	HI	»
6	Beşinci Hâl	Dolama	IH	Kırmızı
7	»	»	HD	»
8	Yeşil	Çözme	DB	Sarı
9	İlmik	Dolama	ED	Kırmızı
10	Yeşil	Çözme	DF	Sarı
11	Minotaur	Dur	...	

CETVEL 2

Hareket No.	Theseus'un kavşakta karşılaştığı durum	Hareket	Seçilen koridor	Koridorun hareketten sonraki hâli
1	Yeşil	Çözme	KN	Sarı
2	»	»	NL	»
3	»	»	LM	»
4	»	»	MN	»
5	İlmek	Dolama	NM	Kırmızı
6	Beşinci Hâl	»	ML	»
7	»	»	LN	»
8	»	»	NK	»
9	Ariadne	Dur	...	

10. LÂBİRENT ALGORİTMASININ İSPATI

Şimdi sayfa 29 da ileri sürülen 1-3 özelliklerini ispat edeceğiz.

İddia 1 in ispatı. Önce aşağıdaki durumların bir ve ancak birinin cari olduğu aramanın her bir halinde, Theseus tarafından yapılan hareketlerin, sayısı üzerine indüksiyonla ispat yapalım:

a) Lâbirentte sarı koridor hiç yoktur, Theseus ta A (Ariadne) kavşağındadır.

b) Lâbirentte bir veya daha fazla sarı koridor vardır ve, Theseus'un aynı sırada geçtiği sarı koridorlar, A dan Theseus'un şimdi bulunduğu yere giden bir yol teşkil etmektedir.

Bundan başka, Theseus'un bir kırmızı koridordan hiç geçmediği tesbit edilir.

Aşikâr ki, Theseus ilk hareketini yapınca a ve b hallerinin ikisinden biri caridir; o zaman o, A da olup bütün koridorlar yeşildir. Şimdi $(n-1)$ nci hareketten sonra a ve b hallerinin birinin cari olduğunu farzedelim, n nci hareketten sonra da birinin cari olması gerektiğini ispat etmeliyiz (tabii, şu şartla ki $(n-1)$ nci hareket bir dur emrine çıkmasın).

Önce, $(n-1)$ nci hareketten sonra a) halinin cari olduğunu farzedelim. O vakit, bundan sonraki hareketi ya bir yeşil koridor boyunca A dan bir K komşu kavşağa gidiş (öyle ki n nci hareketten sonra sadece bir AK yeşil koridorlu b halini elde ederiz) ya da bir dur (öyle ki n nci hareketten sonra a halini elde ederiz) olmalıdır.

Bundan sonra $(n-1)$ nci hareketten sonra b) halinin cari olduğunu ve $AA_1, A_1A_2, \dots, A_{s-1}K$ yörüngesini teşkil eden s tane sarı koridor mevcut olduğunu farzedelim.

n nci hareketin seçimi K kavşağındaki şartlara bağlıdır; imkânlar şunlardır:

1. *Minotaur*. n nci hareket, aynı sarı koridorları bırarak K kavşağında durmaktır (n nci hareketten sonra b hali).

2. *İlmik*. Theseus bu halde kırmızı haline gelen KA_{s-1} sarı koridoru boyunca ipi yumağa dolar. Sarı yörünge daha kısa bir koridor olur. Eğer koridorların s sayısı birden büyükse, n nci hareketten sonra $s-1$ sarı koridorlu b halini elde ederiz; eğer s bir ise, a halini elde ederiz.

3. *Yeşil*. Theseus sarıya çevrilen bir yeşil koridor boyunca ipi yumaktan çözer. Bu halde $s+1$ sarı koridorlu b halini elde ederiz.

4. *Ariadne*. Theseus bu iki a ve b halinden hiç birine göre harekete kalkmaz, çünkü $K=A$ ise, bunu önceliyen, ilmik hali de cari olur.

5. *Beşinci hâl*. Eğer ilk dört durum cari değilse, Theseus ipi dolar. Tıpkı ilmik durumunda olduğu gibi bu eğer $s=1$ ise a haline ve $s>1$ ise b haline getirilmiş olur.

Şimdi a ve b hallerinden muhakkak birinin cari olması gerektiğini çıkarmış olduk. Aşikâr ki Theseus, kırmızı bir koridordan hiç geçmediği için ikiden fazla koridordan geçmez. Koridor sayısı sonlu olduğundan işlem muhtemelen bir durmaya gelmelidir; böylece, son hareket ya Minotaur'un veya Ariadne'nin bulunduğu yerde olan bir kavşağa yapılmalıdır.

2 iddasının ispatı. Dur emriyle Minotaur'un kavşağında karşılaşılırsa, o taktirde Minotaur'a aşikâr olarak geçilebilir. Keza Ariadne'nin ipliği, hemen şimdi gösterdiğimiz gibi sarı koridorlar boyunca kalkış noktasına geri dönen bir yörünge teşkil eder. Bu, Theseus'un her defasında bir ilmik tamamlaması keyfiyetinden gelen basit bir

yol olduğundan, o, ipliği yumağa dolar, böylece ilmiği ortadan kaldırır.

3 iddiasının ispatı. Ariadne'nin kavşağında bir dur halinde, önce şunu kaydedelim:

a) Lâbirentin her koridoru ya iki kere (bir kırmızı koridor) geçilmiştir veya hiç (bir yeşil koridor) geçilmiştir; başka deyişle, lâbirentte hiç sarı koridor yoktur ve ip tamamen tekrar dolanmıştır (zira aksi taktirde ilmik hali cari olur ve durma olmazdı).

b) A dan çıkan bütün koridorlar kırmızıdır, çünkü biri yeşil olsaydı, «Yeşil», listede «Ariadne» den önce geldiğinden, Theseus onu durmadan geçmiş olurdu.

Şimdi 3 iddiasının doğru olmadığını, Minotaur'a bir $AA_1, A_1A_2, \dots, A_M$ yoluyla geçilebilir olduğunu farzedelim. A dan çıktığından bu yolda bulunan ilk koridor kırmızı olmalıdır; Theseus, Minotaur'a erişmediğinden son koridor da yeşil olmalıdır. AA_{i+1} , yolun ilk yeşil koridoru olsun. Bu demektir ki A_i den çıkan bir yeşil bir de kırmızı koridor vardır. Şimdi Theseus'un A_i kavşağından son defa geçtiğini düşünelim. İpi dolamak suretiyle o, A_i den çıkan şimdi kırmızı olan koridorlardan biri boyunca hareket etmiş olmalıdır, bu demektir ki o, ya 2 durumuyla (ilmik), veya 5 durumuyla karşılaşmış olmalıdır. Fakat bu 5 olamazdı çünkü A_i den çıkan bir AA_{i+1} yeşil koridoru mevcuttur; şu halde bir ilmik olmalıdır. Bu ise hemen aşağıdaki delille çelişme teşkil etmektedir: son defa A_i de bir ilmik olsa idi, ondan çıkan en az iki sarı koridor mevcut olmuş olurdu; bu taktirde de birini kırmızı yapmak ama en az birini yine sarı bırakmak suretiyle o, bunlardan biri boyunca yürümüş olurdu. Fakat aramanın sonunda hiç bir sarı koridor kalmadığını görüyoruz, bu demek ki, Theseus, daha sonra A_i den çıkan sarı koridorların kalanlarından geçmiş olmalıdır, ve böylece A_i den tekrar geçmiş olur ki bu da A_i den son defa geçtiğine

ilişkin olan farzımıza aykırı düşer. Bu, üçüncü iddiayı ispat eder.

Bu arama metodunda bir şans unsuru bulunmaktadır. Durum 3 (yeşil) için, bundan sonraki hareket tek şekilde belirlenmiş değildir; kavşaktan çıkan bir çok yeşil koridor varsa, bunlardan biri ilk olarak alınabilir. Bu, son bölümde bütün algoritmalarda bulunması gerekli diye sözünü ettiğimiz deterministik özeliği bozar. Bu şans unsuru bununla beraber kolaylıkla yokedilebilir ve gerçek bir algoritma elde edilir. Belli bir kavşaktan çıkan bir çok yeşil koridor varsa, bunlardan birini seçmekte sadece itibarı olarak karar veririz; meselâ, ilk koridoru daima Theseus'un kavşağa girdiği birinden itibaren saat ibrelerinin dönme yönünden seçeriz. Şans olayını ihtiva eden böyle problemlerin incelenmesi özellikle oyun teorisinde ve bunun iktisada uygulanmasında büyük teorik ve pratik önem taşır.

Tam bu nevin belirlenmesi (belli bir anlamda), hareketlerin seçiminin, sadece oyuncuların kararına bağlı değil gelişigüzel seçimlere de bağlı olduğu geniş bir oyun sınıfı için «en iyi» stratejidir. Bu şekildeki ihtarlar listesi, evvelki bölümde rastgele seçimi bulunmayan oyunlar için anlatılan algoritmaların doğrudan doğruya bir genelleştirilmesidir. Fakat bu mesele üzerine dokunmayacak ve algoritmalar olmak üzere rastgele seçimler ihtiva eden ihtar listelerini gözönüne almayacağız. Bir algoritmanın tam belirlenmesi ve onun belirttiği işlemin akışının biricik oluşu önemli özelliklerindendir.

Söz konusu algoritmaya (şans unsurunu yok ettikten sonra) tekrar dönerek, çok daha özel şekildeki lâbirentler için daha basit arama metotları verilebildiğini kaydetmek gerekir. Aynı zamanda, keyfi bir lâbirent olması genel hali için, bir algoritmanın, bir nevi seçip-ayıklama işleminden başka bir şey olamayacağını farzetmek tabii gibi gelmektedir. Bundan ötürü, bir algoritmanın bizim verdiğimizden daha basit kurulabileceği şüphe götürür.

4. KELİME PROBLEMİ

Kelime problemi. Theseus'un aramasının daha ileri bir genelleştirilmesidir. Theseus'un araması keyfi sonlu bir lâbirentte yapılmakta ise, kelime problemi belli bir anlamda sonsuz bir lâbirentteki aramadır. Problem, modern cebirin *asosyatif sistemler teorisi* ve *guruplar teorisi* adıyla bilinen dallarında ortaya çıkmakla beraber bu teorilerin çerçevesi dışına taşar. Bu problemin çeşitli ifadeleri, iki sovyet matematikçisi A. A. Markov ile P. S. Novikov ve bunların öğrencileri tarafından başarılı sonuçlarla incelenmiştir.

11. ASOSYATİF HESAPLAR

Önce bazı başlangıç kavramları ithâl edeceğiz. Bir *alfabe* deyince herhangi sonlu farklı semboller cümlesini kastediyoruz; bu sembollerden herbirine alfabenin bir *harfi* denir. Meselâ, $\{a, v, z, ?\}$, aşağıdaki harflerden yapılmış bir alfabadir: Yunanca α , gotik v , lâtince z , ve soru işareti. Verilen bir alfabedeki bir *kelime*, alfabeden alınmış harflerden ibaret herhangi bir dizidir. Meselâ, *abaa* ve *bbac*, $\{a, b, c\}$ alfabesinde bir kelimedir. Eğer bir L kelimesi başka bir L kelimesinin kısmı ise, L nin M de geçişinden bahsedeceğiz. Böylece, *abcbcbab* kelimesinde *bc* kelimesinin iki *geçişi* olur, biri ikinci harfle başlamakta, öbürü dördüncü harfle. Bir kelimenin, bir

takım *uygun yerleştirmeler (sübstitüsyonlar)* ile diğerine dönüşümünü gözönüne alacak ve bunları

$$P - Q \quad \text{veya} \quad P \rightarrow Q$$

şeklinde yazacağız, burada P ile Q aynı alfabadeki iki kelimedir.

Bir R kelimesine $P \rightarrow Q$ direkt sübstitüsyonun uygulanması, ancak ve ancak, P , R de geçerse mümkündür. $P - Q$ endirekt sübstitüsyonunun uygulanması ya Q nun bir geçişine ait P sübstitüsyonunun veya P nin bir geçişine ait Q sübstitüsyonundan ibarettir. Bundan böyle esas olarak endirekt sübstitüsyonlarla meşgul olacak ve yanlış anlama olabilmedikçe onlara sadece sübstitüsyon diyeceğiz.

MİSÂL. $ab - bcb$ sübstitüsyonu $abcbcbab$ kelimesine dört türlü uygulanabilir: bcb nin iki geçişinden her birini yerleştirmekle

$$\underline{a} \, ab \, cbab \quad \text{ve} \quad abc \, \underline{ab} \, ab$$

kelimeleri çıkar, ab nin iki geçişinden herbirini yerleştirmekle de

$$\underline{bcb} \, cbcbab \quad \text{ve} \quad abcbcb \, \underline{bcb}$$

çıkarmak. Bu sübstitüsyon $bacb$ kelimesine uygulanamaz.

Bir alfabadeki kelimelerin tümü ile bir uygun sübstitüsyonlar sonlu cümlesini *asosyatif hesap* olarak tarif ediyoruz.

Bir asosyatif hesabı belirlemek için alfabeyi ve uygun sübstitüsyonlar cümlesini vermek yetmektedir.

12. KELİME DENKLİĞİ PROBLEMİ

Eğer bir R kelimesi müsait bir S kelimesine dönüşebilirse R , S ye komşudur diyeceğiz. Kaydedelim ki R , S ye komşuysa, o takdirde S de R ye komşudur. Bir

$$R_1, R_2, \dots, R_{n-1}, R_n$$

kelime dizisinde R_1, R_2 ye, R_2, R_3 e, $\dots$, ve R_{n-1}, R_n e komşuysa R_1 den R_n ye *geçilebilir zincir* denecektir. Eğer R kelimesinden S kelimesine geçilebilir bir zincir mevcutsa, o takdirde bittabi, S den de R ye geçilebilir bir zincir mevcuttur; bu halde R ile S denktir deriz ve bu keyfiyeti $R \sim S$ ile gösteririz. Eğer $R \sim S$ ve $S \sim T$ ise bu takdirde $R \sim T$ olacağı aşikârdır. Daha ilerde aşağıdaki teoremi kullanacağız.

TEOREM. $P \sim Q$ olduğunu farzedelim; bu takdirde P bir R kelimesinde geçerse, R ye $P \rightarrow Q$ süstitüsüyonunun uygulanması R ye denk olan bir kelime sağlar.

İspat. R yi SPT şeklinde yazabiliriz, burada S, P nin geçişini önceliyen R nin kısmıdır, T ise ondan sonra gelen R nin kısmıdır. Bu halde dönüşmüş kelime SQT dir. $P \sim Q$ olduğundan bir

$$P, P_1, \dots, P_m, Q$$

geçilebilme zinciri mevcuttur.

Bu takdirde, kolayca görülebildiği üzere,

$$SPT, SP_1T, \dots, SP_mT, SQT$$

kelimeler dizisi SPT den (yani, R den) SQT dönüşmüş kelimesine geçilebilir. Böylece teorem ispat edilmiş olur.

MİSAL. Aşağıdaki asosyatif hesabı gözönüne alalım:
Alfabe:

$$\{a, b, c, d, e\}$$

Uygun süstitüsyonlar:

$$\begin{array}{ll} ac \text{ --- } ca & abac \text{ --- } abacc \\ ad \text{ --- } da & eca \text{ --- } ae \\ bc \text{ --- } cb & edb \text{ --- } be \\ bd \text{ --- } db \end{array}$$

Bu hesapta sadece üçüncü süstitüsyon *abcde* kelimesine uygulanır, öyle ki ona komşu biricik kelime *acbde* dir. Bundan başka, *abcde* ~ *cadebd* dir ki böyle oluşu

$$\underline{abcde}, \underline{acbd}, \underline{cabde}, \underline{cadbe}, \underline{cadedb}$$

geçilebilir zincirinden gösterilebilir.

Dönüşümlerin hiçbirisi *aaabb* kelimesine uygulanmaz; böylece ona komşu kelime yoktur. Özellikle bu, başka hiç bir kelimenin ona denk olmadığı anlamına gelir.

Her asosyatif hesap kendine ait *kelime denkliliği problemi*ni haizdir:

Hesapta verilen herhangi iki kelimenin denk olup olmadıklarını tayin etmek.

Herhangi bir hesapta mümkün olabilen kelimelerin sayısı sonsuz olduğundan her bir hesap, bu tipten sonsuz sayıda problemi haizdir. Bununla beraber, *herhangi* bir kelime *çiftinin* denkliliği mevcut olup olmadığının tayinine ait bir algoritma şeklinde bir çözüm bulmayı ümit ederiz.

Kelime probleminin tıpkı bir sunî bilmece gibi olduğu ve ona ait bir algoritma bulunmasının gerçek hiç bir

faydası olmadığı sanılabilir. Fakat bu gerçekten uzaktır, problem pek tabii ortaya çıkmakta, büyük teorik ve pratik önem taşımakta, bir algoritma bulunmasına çaba harcanmasına tamamen hak kazanmaktadır. Bununla beraber, bu soruların tartışmasını şimdilik ileriye bırakacağız ve başka hususları gözönüne almaya devam edeceğiz.

13. KELİME PROBLEMLERİ VE LÂBİRENTLER

Kelime denkliği problemi ile Theseus problemi arasında bir bağıntı bulunduğunu şöyle gösterebiliriz: her bir kelime için bir «kavşak» ve iki komşu kelimeyi temsil eden herbir kavşak çiftini birleştiren bir «koridor» inşa etsek, o takdirde herhangi asosyatif hesabı sonlu sayıda kavşakları ve koridorları olan bir lâbirentle temsil edebiliriz. Sadece sonlu sayıda süstitüsyonlar uygun olduğundan, herbir kavşak sadece, kendinden çıkan sonlu sayıda koridora malik olacak ve hattâ koridorsuz kavşaklar bile bulunabilecektir (meselâ, bu bölümün misâlindeki *aaabb* kelimesi gibi). Bir *R* kelimesinden bir *S* kelimesine giden geçilebilir bir zincir, lâbirentte bir kavşaktan öbürüne giden bir yolla gösterilebilir ki bu, kelimelerin denkliğinin, bir kavşaktan ötekine geçilebilir olmağa tekabül ettiği anlamına gelmektedir. Böylece bu temsilde kelime problemi sonsuz bir lâbirentteki bir arama problemi halini almaktadır.

Ortaya çıkan güçlüklerden bazılarını daha iyi görmek için, *kısıtlanmış kelime problemi*ni gözönüne alalım:

Verilen bir asosyatif hesapta herhangi iki R ve T kelimeleri için, k süstitüsyonundan daha fazla bir zincirle birinin diğeri haline dönüşümünün kabil olup olmadığını tayin etmek (burada *k* keyfi fakat sabit pozitif tam sayı).

Bu problem için bir algoritma yani lâbirent probleminde kullanabildiğimiz gibi bir ayıklama algoritması kurulması kolaydır. R kelimesi kaydedilir, sonra R ye komşu bütün kelimeler, daha sonra bunlara komşu bütün kelimeler, ilâ, k defa kaydedilir. O takdirde, R nin k sübstitüsüyondan fazla kullanmıyarak T ye dönüşüp dönüşmemesinin araştırılması, T nin bu listede görünüp görünmemesinin araştırılması demektir.

Kısıtlanmamış kelime problemine dönersek, durum çok farklıdır. R den T ye giden geçilebilme zinciri herhangi bir uzunluktakine (şayet mevcutsa) dönebildiğinden genel olarak ayıklama işleminde ne zaman durulacağını bilmek için elimizde bir yol mevcut değildir. Meselâ, daha önce metodu $10^{20} = 100.000.000.000.000.000.000$ kere tekrar ettiğimizi farzedelim, böylece R nin 1, 2, 3, ..., veya 10^{20} uygun sübstitüsyonla tekrar edilebildiği bütün kelimelerin bir listesini elde ederiz. Ayrıca T nin bu listede gözükmediğini farzedelim. T nin R ye denk olmadığını söyleyebilir miyiz? Cevap, tabii söyleyemeyiz şeklindedir, çünkü R ile T nin denk olmak imkânı henüz mevcuttur, ama bunları birleştiren en kısa geçilebilme zinciri 10^{20} den fazla adımı haizdir. (Bir alıştırma olarak, okuyucu, denk olan fakat 10^{20} veya daha az uzunluktaki hiç bir geçilebilme zinciriyle birleşmemiş olan bir kelime çiftini haiz bir hesap kurabilir.)

14. ALGORİTMALAR KURULMASI

Bir algoritma elde etmek için basit bir çıkışın bulunuşu düşüncesini bırakıp, bizzat dönüşüm mekanizmasının analizi üzerine kurulmuş başka fikirleri incelememiz gerekir. Meselâ bölüm 12 nin Misâlindeki hesapta *abaacd* ve *acbdad* kelimelerinin denkliği problemini alalım. Bu

kelimelerin aşağıdaki şekilde denk olmadığını ispat edebiliriz: Uygun süstitüsyonların herbirinde, her kenar a lardan aynı sayıda ihtiva etmektedir. Bundan ötürü, bu hesaptaki herhangi bir geçilebilme zincirinde, her kelime a nın geçişlerini aynı bir sayıda haiz olmalıdır. $abaacd$ ve $acbdad$ kelimelerindeki a nın geçiş sayısı aynı olmadığından, bu kelimeler aynı bir geçilebilme halkasına ait değildirler dolayısıyla denk değildirler.

Bir geçilebilme zincirinin bütün uzuvlarında ortak olan böyle bir özellik, *geçilebilme invariantı* adını alır. Bir geçilebilme invariantı bir algoritma bulunmasında bize faydalı olabilir.

MİSÂL 1. Aşağıdaki asosyatif hesabı gözönüne alalım:

Alfabe:

$$\{a, b, c, d, e\}$$

Uygun süstitüsyonlar:

$$ab \text{ --- } ba \quad ae \text{ --- } ea \quad be \text{ --- } eb \quad de \text{ --- } ed$$

$$ac \text{ --- } ca \quad bc \text{ --- } cb \quad cd \text{ --- } dc$$

$$ad \text{ --- } da \quad bd \text{ --- } db \quad ce \text{ --- } ec$$

Burada müsait süstitüsyonlar bir kelimedeki gözüken harflere değil, fakat sadece bunların sırasına etki eder. Eğer iki kelimeden her biri ancak ve ancak öbürü kadar her kelimenin aynı sayıda geçişini ihtiva ederse bu iki kelimenin denk olduğunu göstermek güç değildir. Böylece şu gösterilen halde denkliği tayin etmek için basit bir algoritma elde ederiz: Herbir kelimedeki herbir kelimenin geçiş sayısını sayınız, ve sonuçları mukayese ediniz.

Şimdi «kelime» ve «uygun süstitüsyon» kavramlarının bir genelleştirilmesini işe sokuyoruz. Verilen bir

alfabede âdi kelimelere ilâveten hiç harfi olmıyan boş kelime kabul edeceğiz ve bunu Λ ile göstereceğiz. Keza

$$P - \Lambda$$

şeklindeki süstitüsyonlara cevap vereceğiz. P ye ait Λ nın süstitüsyonu basitçe P nin bir R kelimesinde bir geçişi varsa onu tamamen atacağız anlamına gelir; Λ ye ait P nin süstitüsyonu, P nin R kelimesi içine herhangi bir noktaya, R nin ilk harfinden önce, yahut R nin iki kelimesi arasına veya R nin son harfinin sonuna ithâl edilebilir anlamına gelmektedir.

Şimdi aşağıdaki misâli gözönüne alıyoruz:

MİSÂL 2. $\{a, b, c\}$ alfabesiyle bir asosyatif hesap ve:

$$b - acc$$

$$bb - \Lambda$$

$$ca - accc$$

$$cccc - \Lambda$$

$$aa - \Lambda$$

uygun süstitüsyonlar cümlesi verilmiş olsun, bu hesapta kelime denklğine ait bir algoritma bulalım.

Önce bir yardımcı algoritma, *irca algoritması* kuralım. Bu, herhangi bir kelimeyi, bunun *irca edilmiş kelimesi* denilen özel bir şekle dönüştürmeğe yarıyan bir algoritmadır.

Doğrudan doğruya olan (direkt) süstitüsyonların aşağıdaki sıralanmış sistemini gözönüne alalım. Herhangi bir R kelimesi verildiğine göre, listede buna uygulanabilen ilk doğru süstitüsyonu bulalım ve soldan ilk mümkün uygulama yerine yerleştirilmesini yapalım. Bu denk bir R' kelimesi verir. Listedeki R' ye uygulanabilen ilk süstitüsyonu bulalım ve onu uygulayalım. Sonlu sayıda bir

sübstitüsyondan sonra hiç bir dönüşümün uygulanamadığı bir S kelimesi elde ederiz; *algoritmanın R kelimesini S ye irca ettiğini* söyleriz *.

İrca algoritmasının, herhangi bir R kelimesini şu sekiz kelimeden (irca edilmiş kelimeler) birine dönüştürdüğünü gösterebiliriz:

$A, c, cc, ccc, a, ac, acc, accc$.

1. Eğer b kelimesi R kelimesinde gözükmekteyse, o takdirde ilk doğru (direkt) sübstitüsyonun ardarda uygulanması b nin her geçişini b kalmayınca kadar acc haline değiştirir; b herhangi diğer bir sübstitüsyonda gözükmeyişinden, herhangi bir irca edilmiş kelimede de gözükmeyiz.

2. a harfi herhangi irca edilmiş bir kelimede c nin doğrudan doğruya sağında gözükmeyiz, çünkü ikinci doğru sübstitüsyon ca nın her geçişini $accc$ haline değiştirmektedir.

3. Herhangi irca edilmiş bir kelimede ardışık iki a olamaz.

4. c harfi son sübstitüsyondan ötürü bir satırda üç defadan fazla bulunamaz.

Sekiz kelimenin yukardaki listesi bütün irca edilmiş kelimeleri havidir, çünkü yukarda verilen incelemeye göre irca edilmiş bir kelime en az üç c den sonra gelen hiç olmazsa bir a dan ibarettir.

Aşikâr olarak, her kelime kendi irca edilmişine denktir; bundan ötürü iki kelime ancak ve ancak bunların irca edilmiş kelimeleri ya aynı veya denkse birbirine

*) Verilen bir alfabe'deki kelimeleri doğru sübstitüsyonların sıralanmış bir cümlesine dönüştüren bunun gibi algoritmalara *normal algoritmalar* denir. Büyük teorik ve pratik faydası olan normal algoritmalar A. A. Markov tarafından geliştirilmiş ve kendisinin *Theory of Algorithms* adlı kitabında gösterilmiştir, bu kitap İngilizceye çevrilmiştir.

denktir. Fakat yukarda listelenmiş bulunan sekiz irca edilmiş kelimedenden ikisinin denk olmayacağını kısaca ispat edeceğiz. Bundan, iki kelimenin ancak ve ancak aynı kelimeye irca edilirse denk olduğu çıkar. Buna dayanarak kelime problemi için bir algoritma kurabiliriz: *incelenmiş olan iki kelimedenden her birine irca algoritması uygulayınız, ve elde edilen sonuçları mukayese ediniz; sonuçlar birbirini tutarsa ilk kelimeler denktir; aksi takdirde denk değildir.*

Meselâ, *cacb* ile *bb* kelimelerini verdiğimizizi farzedelim. Önce irca edilmiş kelimeleri bulalım:

- 1) *cacb*, *cacacc*, *acccacc*, *acccacccce*,
accaccccccc, *acacccccccccc*,
aacccccccccccccc, *ccccccccccccccc*,
cccccccccc, *cccccc*, *cc*.
- 2) *bb*, *accb*, *accacc*, *acaccccc*, *aacccccccc*,
cccccccc, *cccc*, Λ .

Sonuç. *cacb* ve *bb* kelimeleri, irca edilmiş kelimeleri olan *cc* ile Λ nin farklı oluşundan ötürü denk değildir *.

Sekiz irca edilmiş kelimedenden ikisinin denk olmadığının ispatı. İlk önce, iki *R* ile *S* kelimesini birleştiren, hiçbir *b* harfini ihtiva etmiyen bir geçilebilme zinciri varsa, bu takdirde içindeki harflerden hiçbir *b* harfini ihtiva etmiyecek şekilde bunları birleştiren bir geçilebilme zinciri kurmak mümkündür. Verilen geçilebilir bir zincirdeki her kelimeyi alarak ve ona *b* harfinin her geçişi için *acc* kelimesini yerleştirmekle bunu yapacağız. Bu bize, içinde her ardışık iki kelimenin ya (hesap anlamında) komşu veya özdeş olduğu kelimeler dizisi verir. Şayet şimdi

*) *bb* nin irca edilmiş kelimesinin Λ oluşu keyfiyeti, ilk listemizdeki *bb* — Λ sübstitüsyonunun gereksiz olduğunu gösterir, ki irca algoritmasına onu niye almadığımızı açıklar.

kendinden önceki kelimeyle uygunluk gösteren her kelimayı çıkarırsak bize bir geçilebilme zinciri kalır. Bundan başka, $b - acc$ süstitüsüonu zincirde kullanılmamıştır.

Önce, kaydedelim ki a harfinin geçiş sayısının çift veya tek oluşu her uygun süstitüsyonun her iki tarafında aynıdır. Aynı şey c harfinin geçişi için de doğrudur. Bu, a nın (veya c) nin geçişleri sayısının *paritesinin* (yani «çiftliği» veya «tekliğinin») bu tipten herhangi bir geçilebilme zincirine ait bir geçilebilme değışmezi olduğu anlamına gelir. Bundan derhal a nın bir geçişini ihtiva eden irca edilmiş dört kelimededen hiç birinin a ları hiç ihtiva etmiyen dört irca edilmiş kelimededen herhangi birine denk olmadığı çıkar. Benzer şekilde, bir veya üç c yi ihtiva eden dört kelimededen hiçbiri, hiç c ihtiva etmiyen veya iki c ihtiva eden dört taneden herhangi birine denk değildir. Bundan ötürü, şimdi aşağıdaki dört çift kelimenin denk olmamazlığını kurmak mecburiyetindeyiz:

$$\Lambda, cc; c, ccc; a, acc; ac, accc.$$

Eğer ilk üç çiftin herhangi biri denkse, o takdirde, daha önce (sayfa 37) de ispat ettiğimiz teoremden dördüncü çiftin de denk olacağı çıkar. Böylece, sadece dördüncü $ac, accc$ çiftine ait denk olmamazlığı kurmak yetişir.

Önce bazı deyimler tarif edeceğiz. Bir R kelimesindeki a nın bir geçişinin indisinden a nın verilen geçişinin sağındaki kelimedeki c nin geçiş sayısını kastediyoruz. R kelimesinin indisinden a nın R deki bütün geçişlerinin indisleri toplamını kastediyoruz*. $aa - \Lambda$ ile $cccc - \Lambda$ süstitüsyonlarından hiçbiri bir kelimenin indisinin paritesini değıştirmmez. Eğer aa bir boş kelimeye yerleştirilirse,

*) Meselâ $acba$ kelimesinde a nın ilk geçişinin indisi 2 dir, a nın ikinci geçişinin indisi de 0 dir. Kelimenin indisi $2 + 0 = 2$ dir.

kelimenin indisi a nın iki geçişinin indislerinin toplamı kadar artar. Fakat bu indislerin her ikisi aynı olmalıdır, bu demektir ki kelimenin indisi bir çift sayı kadar artar böylece paritesi (teklik - çiftliği) değişmez. Benzer şekilde, aa yerine boş bir kelime yerleştirilirse, kelimenin indisi bir çift sayıya irca edilmiş olur.

$cccc$ yi ilâve etmede a nın bazı geçişlerinin indisleri 4 kadar yükselir, diğerleri ise aynı kalır; böylece, kelimenin indisi 4 ün bir katı kadar artar. $cccc$ yi çıkarma ise benzer etkiler yapar.

Nihayet, $ca - accc$ süstitüsüyonu herhangi bir kelimenin indisinin paritesini değiştirir. Bunu göstermek için,

$$PcaQ \quad \text{ve} \quad PacccQ$$

kelimelerini mukayese edelim.

Kelimenin P kısmında a nın her geçişinin indisi 2 kadar değişir, Q kısmında ise a nın her geçişinin indisi değişmez kalır. P ile Q arasında a nın bir tek geçişinin indisi 3 kadar değişir. Böylece kelimenin indisi tek bir sayı kadar değişir, öyle ki paritesi değişmiş olur.

Şimdi ac ve $accc$ kelimeleri aynı pariteden indisleri haizdir (1 ve 3 — her ikisi de tek). Bundan ötürü, bunlar denkse, o takdirde bunları birleştiren herhangi bir geçilebilme zincirinin $ca - accc$ süstitüsyonlarından çift bir sayıda ihtiva etmesi gerekir (evvelce zikredildiği gibi b nin geçişlerinin bulunmadığı zincirlerle yetineceğiz).

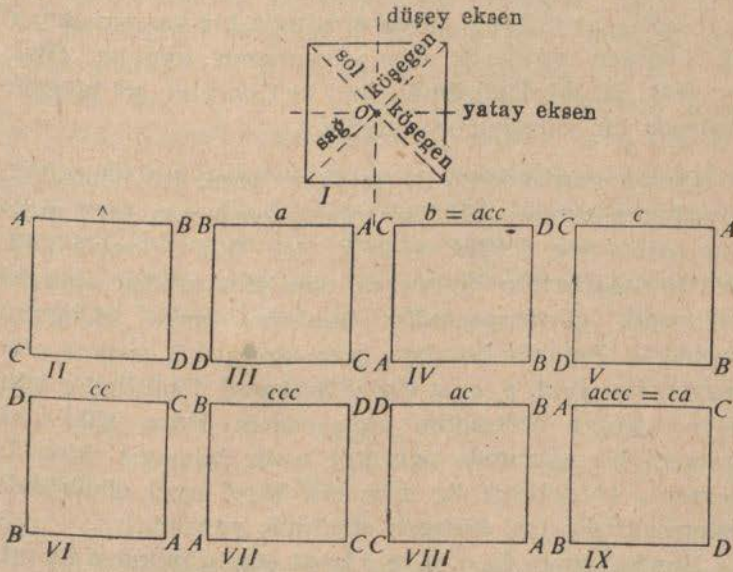
Fakat bu şart bir çelişmeye çıkar. $ca - accc$ süstitüsyonunun herbir uygulanaşı c nin geçişleri sayısını 2 kadar değiştirir, öyle ki çift sayıda uygulamalar c nin geçişleri sayısını 4 ün bir katı kadar değiştirir. Aşıkâr ki, $cccc - \Lambda$ süstitüsüyonu c nin geçişleri sayısını 4 kadar değiştirir, $aa - \Lambda$ süstitüsüyonu ise c nin geçiş sayılarını hiç değiştirmez. Söylediklerimizden, şayet ac ile $accc$ denk iseler, o takdirde bunlardaki c nin geçişleri sayısının 4 ün

bir katı kadar değişmesi gerektiği sonucunu çıkarırız ki bu doğru değildir. Bundan ötürü, ac ile $accc$ denk değildir, demek ki, iddia edildiği gibi, irca edilmiş kelimelerden hiçbirisi denk değildir.

Misâl 2 deki kelime probleminin mufassal çözümü, genel olarak kullanılan kavram ve metotları bir çok bakımlardan açıklar. Artık bu problemin modern cebirdeki ilgi ve önemini göstermek kalır. Gelecek bölümde özel bir misâl gözönüne alarak bunu yapacağız.

15. BİR KARENİN OTOMORFİZMALARI

Herhangi bir kare alalım (Şek. 9 I). Aşağıdaki üç otomorfizmayı (yani, kareyi kendine dönüştüren dönüşümler) gözönüne alalım:



Şekil: 9

1) O merkezinden geçen düşey bir eksendeki aynada yansıma (simetri);

2) O merkezinden geçen yatay bir eksendeki aynada yansıma (simetri);

3) O merkezi etrafında saat ibreleri yönünden 90° kadar dönme. Bunlara *basit dönüşümler* diyecek ve bunları sırayla a , b ve c ile göstereceğiz. Şek. 9 un III, IV ve V inci kısımları Şek. 9 II deki karenin köşelerinin basit dönüşümlerin herbiriyle nasıl değiştiğini göstermektedir.

Basit işlemlerin iki veya daha fazlasının ardarda uygulanışıyla kare kendine de dönüşür. Alışıldığı gibi iki dönüşümün *çarpımının*, bunların ardarda uygulanışı anlamını taşıyacağını tarif olarak kabul edeceğiz. Yine çarpmaya ait bilinen notasyonu kullanacak ve sonuç dönüşüm için *çarpım* kelimesini kabul edeceğiz. Meselâ, cc çarpımı 90° lik ardarda iki dönmenin sonucudur. yani, 180° kadar bir dönmedir. ac çarpımı 90° kadar bir dönmeyi takip eden düşey eksen etrafında bir yansımadır, ki sağ köşegen etrafında bir yansımanın aynıdır (Bak. Şek. 9 I). Bu iki dönüşümün $(ac)(cc)$ çarpımı sol köşegen etrafında bir yansımanın aynıdır.

Çarpımımız komütatif değildir; yani, iki dönüşümü çarpığımız sırada değişiklik olursa çarpımda fark meydana gelir: Şek. 9 VIII ve 9 IX, Şek. 9 II de verilen ilk karenin köşelerinin ac ve ca dönüşümlerinden sonraki sıralanışını göstermektedir; bunların farklı olduğunu görebiliriz. Bununla beraber, çarpmamız, âdi çarpma gibi asosyatiftir; yani, p , q ve r gibi herhangi dönüşümler için $(pq)r = p(qr)$ özdeşliğini elde ederiz. Buna şükür ki herhangi bir çarpımda parantez nedir bilmeyiz. Meselâ, $(ac)(cc)$, $((ac)c)c$ ile $accc$ nin hepsi aynı dönüşümü göstermektedir (sol köşegen etrafında yansıma).

Bundan sonra üç a , b ve c basit dönüşümünden ibaret

cümle ile ve basit dönüşümlerin sonlu fakat keyfi sayıdaki çarpımlarıyla gösterilebilen bütün dönüşümlerle ilgileneceğiz. Çünkü çarpımımızın asosyatifliğinden ötürü parantezleri kaldırırız ve elemanter işlemleri kendi sıralarıyla basitçe kaydederek bu çarpım dönüşümleri gösteririz, meselâ, abb , $cabb$, $accc$ gibi. Demek ki bunları $\{a, b, c\}$ alfabesindeki kelimeler olarak gösterebiliriz.

$\{a, b, c\}$ alfabesinde bazı kelimelerin sonsuz sayıda olduğu aşikârdır, fakat iki farklı kelimenin aynı dönüşümü gösterdiği vâki olabilir. Bu halde kelimelerin eşit olduğunu söylemek ve eşitlik için alışılan notasyonu kullanmak tabiidir. Okuyucu

$$b = acc, \quad (1)$$

$$ca = accc \quad (2)$$

eşitliklerini kolayca gerçekliyebilir. Bunları ispat için, aynı bir kareye denklemin iki yanındaki dönüşümleri uygulayınız ve elde edilen kareleri mukayese ediniz.

Keza, aa , bb , $cccc$ kelimelerinden herbirinin aynı dönüşümleri temsil ettiğini görmek kolaydır; buna karenin köşegenlerini ilk konumlarında bıraktığından *özdeşlik* dönüşümü denir. Bu dönüşümle gerçek olarak hiç bir şey değişmediğinden bunu Λ boş kelimesiyle gösteririz. (Bu kabulün evvelki iddiayı değiştirmedini gerçeklemeyi okuyucuya bırakıyoruz.) Böylece

$$aa = \Lambda, \quad (3)$$

$$bb = \Lambda, \quad (4)$$

$$accc = \Lambda \quad (5)$$

dir.

(1) — (5) eşitliklerinin bölüm 14 ün Misâl 2 sindeki asosyatif hesabın uygun süstitüsyonlarıyla mukayese,

bu hesapla bir karenin dönüşümleri arasında bir bağ kuran aşağıdaki ifadeye götürür:

Bir karenin basit dönüşümlerinin iki çarpımı, eğer ancak ve ancak bunları gösteren kelimeler bölüm 14, Misâl 2 deki hesapta denk ise birbirine eşittir.

(1) — (5) denklemlerinden, herhangi uygun bir süstitüsyonun keyfi bir S kelimesine uygulanışının eşit bir kelime husule getirdiği çıkar. Meselâ, $ca - ac$ süstitüsyonunun $bcac$ kelimesine uygulanışı $bacccc$ kelimesini verir, fakat çarpım asosyatif olduğundan $bcac = b(ca)c$ ve $bacccc = b(accc)c$ yazabiliriz. Eşit çarpanların çarpımı olduğundan bu iki denklemin sağ tarafları eşittir, ki sol tarafların eşit olması gerektiği anlamına gelmektedir. Böylece; *herhangi iki komşu kelime eşittir.*

Şimdi eğer iki kelime asosyatif hesabımızda denk ise, o takdirde bunlara karşılık gelen dönüşümlerin eşit olduğunu göstermek kolaydır. Eğer $S \sim T$ ise, o takdirde bunlara karşılık gelen geçilebilir zincirde her bir iki komşu eleman eşittir ve bundan ötürü $S = T$ dir.

Karşıtı da doğrudur: *eğer iki kelime eşitse, bunlar denktir.* Eğer iki kelime eşitse, o takdirde bunların irca edilmiş kelimeleri de eşittir. (Bu hemen şimdi ispatladığımız ifadeden çıkar.) Fakat irca edilmiş kelimelerden ikisi eşit değildir; Şek. 9 II - IX da, Şek. 9 II deki kareye irca edilmiş kelimelere karşılık gelen dönüşümlerin uygulanışının sonucunu göstermekle anlaşılabilir. Böylece, şayet iki kelime eşitse, bunların irca edilmiş kelimeleri aynı olmalıdır, ki bu kelimelerin denk olması demektir.

Sırf formel denklik fikrine müşahhas bir geometrik anlam verdik ve müşahhas geometrik problemin çözümü yerine denkliğin belirlenmesini koyduk. Üstelik, bu tipten geometrik problemin çözümüne ait genel bir metot olmak üzere hemen çıkarılmış bulunan algoritma yine ortaya çıkmaktadır. Durum, sırf şekli denklik probleminin özel geometrik, cebirsel, veya başka yorumları haiz olabildiği

öteki hesaplardakine benzemektedir. Matematiğin her kolunda, bir hesapta iki kelimenin denkliğini ilgilendiren bir ifade olarak formüle edilebilen teoremlerin var olduğunu mübalâgasız söyleyebiliriz. Bunu ilerde inceliyeceğiz (bölüm 25 e bakınız).

Sona gelmişken, bölüm 14 deki Misâl 2 nin hesabını vermiş olmakla hattâ, biraz daha basit, doğrudan doğruya (direkt olarak) bir algoritma kurabileceğimize işaret edelim. Zira, bu hesapta iki kelimenin denkliğini denemek için bir karenin buna karşılık dönüşümlerini çizmek ve elde edilen kareleri karşılaştırmak yetiştir.

PROBLEM. $\{a, b\}$ alfabesi ve

$$aaa = bb,$$

$$bbbb = \Lambda.$$

uygun süstitüsyonlarıyla tarif edilen asosyatif hesaba ait kelime problemini çözünüz.

5. OTOMATİK KONTROLLÜ HESAP MAKİNALARI

16. İNSAN TARAFINDAN HESAPLAMA

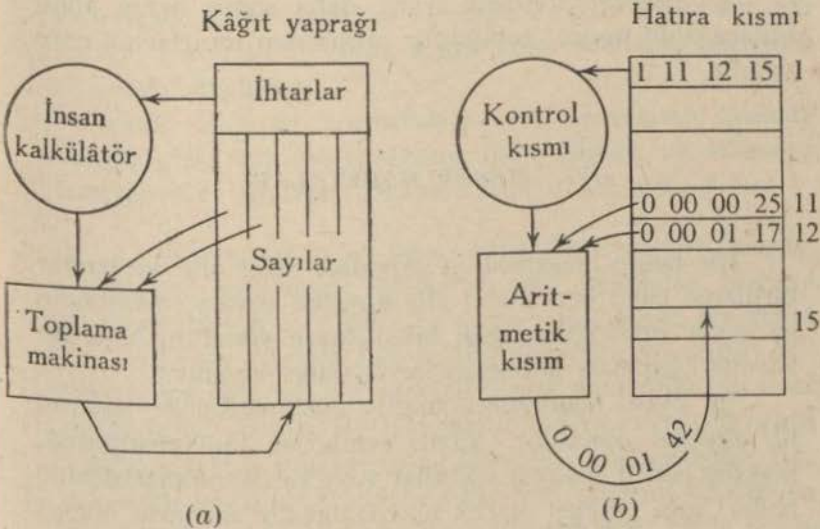
Bir algoritma, özellikle pratik olanı, ince ve karışık muhakemeler üzerine dayandığından algoritmaların kuruluşu yüksek derecede bir maharete ihtiyaç gösterir. Fakat bir algoritma bulununca maksadını bilmiyen bir kimse tarafından da bile kullanılabilir. O sadece, algoritma ile bir sürü gerekli işlemleri yapmağa ve tam yönetmeliği verildiği şekilde takip etmeğe muktedir olmak zorundadır. Büsbütün mekaniksel şekilde davranarak, algoritması düzenlenen tipten herhangi bir problemi çözebilir. Böyle bir kimsenin «bir makina gibi» hareket ettiğini söyleyebiliriz; bunu tabii, algoritmanın kesinlik tabiatını kastederek mecazi olarak söyleriz, ama ilim ve tekniğin ilerlemesiyle, cümle, gerçek bir anlam kazanır. Problemi çözerken ne yaptığını anlamıyan (veya anlamak istemiyen) bizim farazi şahıs yerine, şimdi bir makina kullanmak mümkündür. İşte otomatik kontrollü modern hesap makinası böyledir.

Ödevimiz böyle makinaların inşası ve işleminin temel prensiplerini anlatmak olacaktır.

Önce, insan kalkülâtör ile yapılan algoritmik işlemi inceliyeceğiz. Bir algoritmayı takipte, bir kalkülâtör çeşitli neveleri (veya *bilgileri*) alır, işler ve saklar. Çoğu defa rakamlar, harfler ve başka semboller kullanarak onları kâğıt üzerine kaydeder (onları *takdim eder*). Böyle

bir sembol gurubuna *alfabe* diyoruz. Meselâ, cebirde âdi harfler, sayılar, aritmetik işlemlerine ait işaretler, parantezler, v.s. ihtiva eden bir alfabe kullanıyoruz.

El ile yapılan bir hesap aşağıdaki şu üç âmili ihtiva eder (bak. Şek. 10 a).



Şekil: 10

1. *Bilginin biriktirilişi*, alışıldığı gibi bütün verilerin problemin çözülmesine ait ihtarlarla (algoritma) birlikte bir kâğıda kaydıyla yapılır. Pratikte, tabii, kalkülâtör her şeyi kaydetmez. Bazı şeyleri hatırında tutar (kâğıttan ziyade aklında saklar), bazılarını ise kartlarda veya cetvellerde bakar. Bununla beraber bu, bazı araçların bütün gerekli bilgilerin biriktirilmesi için şart olması esas keyfiyetini karanlıkta bırakmamalıdır. Böylece, şeklimizdeki kâğıt parçası deyiminden *bütün* bilgi saklanması yollarını göstermesi anlaşılmalıdır.

2. *Bilginin işlenmesi* gerekli basit işlemlerin algoritmayla yapılması anlamına gelir. Bu, mekanik cihazlar

kullanmak suretiyle yapılır; meselâ, aritmetik işlemler ilâve bir makina veya hesap cetveliyle yapılabilir. Hesaplanan her adımı kâğıttan bir takım bilgiler (meselâ sayılar) almaktan, onun üzerinde özel bir işlem yapmak ve sonucu kâğıdın belli bir yerine kaydetmekten ibarettir.

3. *İşlemin kontrolü*, yani, daha sonra hangi adım atılacağıнын tayini, kalkülâtör tarafından ihtarlarına göre yapılır.

17. HESAP MAKİNALARI

Bir hesap makinasının parçaları neler olur ve bunlar birbirine nasıl etki eder? Bu sorunun cevabı, makinanın az evvel anlattığımız gibi fakat insan yönetimi olmadan işlemler yapması gerektiği keyfiyetine dayanır.

İlk önce, makinanın, bilgiyi gösterebilmeğe yarıyan bir *alfabesi* olmalıdır. Yazılı semboller kullanmaktansa, makina çeşitli elektrik voltajlar veya çeşitli miknatıslanma halleri gibi fiziksel olarak ayırdedilebilir şartlarla bilgiyi göstermektedir.

Bazı düşünceler sebebiyle 0 ve 1 denilebilen tam iki sembolden ibaret bir alfabe (*ikili* alfabe) kullanmak elverişlidir. Meselâ bu alfabe yüksek bir voltajla ve alçak bir voltajla (veya akımla ve akımsız) kolaylıkla takdim edilebilir. Bundan başka, en basit lojik işlemlerin şu iki değerden birini alabilen değişkenler üzerinde yapıldığı hesaba katılmalıdır: doğru veya yanlış. Bununla beraber, bir alfabenin seçilişi ile bundaki bilgi takdim metodunun, yapı hakkındaki bilgimiz ve bir makinanın işleyişi üzerinde bir dayanağı yoktur. Bundan ötürü ikili bir alfabe ile ikili bir hesabın (daha alışık olduğumuz ondalık sistem yerine) modern makinalarda kullanıla geldiğine işaret etmekten başka teferrüata dalmıyacağız.

Böylece makina konulan bilgi gibi, hesaplama sırasında meydana getirilen bilgi de, birtakım fiziksel parametreler şeklindedir. İlgilendiğimiz bu hallerde, bütün bilgiler sayısal olarak şifrelenmiştir. Özel olarak, makinanın takip etmesi gereken algoritmanın kendisi bir sayılar listesi şeklinde şifrelenmiştir. Makinada yerine getirilmek üzere yazılmış bir algoritmaya *program* adı verilir. Program bilginin, makinanın üzerinde uğraştığı en önemli kısımdır.

Şekil 10 a da gösterilen el ile hesaplama halinde olduğu gibi makina, saklıyan, bilgi işliyen ve işlemeyi kontrol eden kısımları haiz olmalıdır (bak. Şek. 10 b):

1. *Hafıza (Hatıra) kısmı* kâğıt ödevi görür. Burada, makinanın «dilinden», bütün gerekli bilgilerle birlikte program kayıtlıdır. Aşikâr ki, böyle işleri gören bir kısım inşa edilebilir. Meselâ hatıra, üzerinde gerekli şifreli bilginin saklandığı âdi bir banttan ziyade manyetik bir bant (teyp, şerit) olabilir. Bir makinanın hatıra kısmı, *adresleri* denilen sayılarla markalanmış *hücrelere* bölünmüştür. Her bir hücre bilginin şifrelenmiş bir müfredatını saklamaya güçlüdür veya saklar; daha önce zikrettiğimiz böyle bütün bilgiler sayısal şekildedir.

Pratik makinalarda hatıra görevi katot ışınli tüplerle, manyetik kasnaklar, manyetik sargılar veya bu cihazların birkaçının bir araya getirilmesiyle yapılabilir. Bununla beraber, bunları birbirinden ayırdetmeğe zorunlu değiliz ve maksadımız için hatıra kısmını bir tek âletten meselâ bir manyetik banttan ibaret farzedeceğiz.

2. *Aritmetik kısmı* hernekadar kuruluş prensipleri farklı ise de kasa kalkülâtörünün oynadığı rolün aynıını oynar. Bu verilen bir işlemi (meselâ iki sayının toplanmasını), işleme katılan sayıları gösteren elektrik işaretlerini, elektronik olarak, sonucu gösteren elektrik işaretler

haline çevirerek yapar. Giriş işaretleri, saklandıkları hatıra hücresinden aritmetik kısma geçerler, sonucu gösteren çıkış işaretleri de, saklanacakları hücreye giderler. Bu şematik olarak Şek. 10 b de gösterilmiştir, burada 11 ve 12 hücrelerindeki sayılar toplanmış ve toplam 15 hücresinde saklanmıştır. Belirli bir zaman devresinde bu işlemin makinayla yapılması için, devrenin başında 11 ve 12 hücrelerinden aritmetik kısma, ve aritmetik kısımdan 15 hücresine bağlantılar yapılmalıdır ayrıca aritmetik kısım ayrılan işleme (bu halde toplamaya) hazır olmalıdır. Bütün bunlar kontrol kısmıyla yapılır.

3. *Kontrol kısmı* Şek. 10 a da kalkülâtörün kendisi tarafından yapılmış görevlerden sorumludur. Kontrol kısmı hesaplamanın her adımında gerekli işlemin yapılmasına ait şartları kurmalıdır. O, her bir işleme katılan bu «sayılar» ı (hatıra hücreleri, aritmetik kısma giriş hatları, v.s.) bağliyan bir telefon santrali gibi çalışır. Mecazi anlamda sözederek, kontrol bloku yönetime ait programı danışıp görüşür ve sonra makinanın yapılacak işleme katılan o parçalarına özel emirler verir.

18. MAKİNA İHTARLARI

Her makina yapabildiği belirli bir ihtarlar takımıyla karakterize edilir. Her program ihtarlarla yardımcı sayıların (parametreler) özel bir kombinezonundan ibarettir; bunlar hatıra hücresinde saklıdır. Bazı makinalar meselâ rus BESM gibi, ihtarlar için üç adresli* bir formayı haizdir; ihtar dört sayıdan ibaret bir

$\alpha \beta \gamma \delta$

*) En çok ticarî maksatla imâl edilen makinalar bir adresli makinlardır. MANIAC'lar gibi iki adresli makinlar da vardır.

dizisiyle gösterilir ki bunların birincisi işlemi, ondan sonrakisi muhtevalar üzerinde işlem yapılacak iki hücrenin adresleri, dördüncüsü ise içinde sonucun saklanacağı hücrenin adresini (hepsi üç adres) göstermektedir.

Her bir ihtar tek bir hücrede, rakamları gösterilen anlamları haiz dört guruba ayrılmış bulunan bir sayı şeklinde yapılmıştır. Meselâ, Şek. 10 b de, 1 hücresi 1 11 12 15 sayılarını ihtiva etmektedir ki «11 ve 12 hücrelerindeki sayıları topla (1 işlemi) ve sonucu 15 hücresinde sakla» ihtarına ait şifredir. (Rakamları soldan sağa doğru ikişer ikişer guruplara ayırdık. Bu kabul bundan sonra kullanılacaktır.)

Çok defa bir makinaya ait bir kaç düzüne ihtar vardır *. En çok ortak olarak kullanılanlardan bazıları şunlardır.

1. Aritmetik ihtarlar:

- a) 1 $\beta \gamma \delta$ — β daki sayıyı γ daki sayıya ekle toplamı δ da sakla.
- b) 2 $\beta \gamma \delta$ — β daki sayıdan γ daki sayıyı çıkar ve farkı δ da sakla.
- c) 3 $\beta \gamma \delta$ — β daki sayıyı γ daki sayıyla çarp ve çarpımı δ da sakla.
- d) 4 $\beta \gamma \delta$ — β daki sayıyı γ daki sayıyla böl ve bölümü δ da sakla.

2. Atlama ihtarları:

- e) 5 00 00 δ — δ da saklı bulunan ihtara devam et (şartsız atlama).

*) Genel eğilim çok sayıda ihtarlı makinalara doğru gitmektedir. Şimdi bir çok makinaların yüzlerce ihtarı vardır. Bunlardan biri (Rice Institute makinası) 8000 kadar ihtarı haizdir.

- f) 5 01 $\gamma \delta$ — Şayet γ ancak ve ancak pozitif bir sayı ihtiva ederse δ da saklı ihtarâ devam et.
- g) 5 02 $\gamma \delta$ — Şayet γ ancak ve ancak negatif bir sayı ihtiva ederse δ da saklı ihtarâ devam et.

3. «Dur» emri:

0 00 00 00 — Dur.

Sıraladığımız ihtarlar meyanında, üzerinde durmıya-
cağımız lojik ihtarlar denilenleri ve daha başkaları mev-
cuttur. Yukarda verilenler geniş ölçüdeki çeşitli program-
lar kurulmasına yetmektedir.

Ihtarlar makinayla çoğu defa hatıradâ geçtikleri sıra
dahilinde yapılır. Bu emirden dışarı çıkışlar sadece bir
atlama ihtarının yerine getirilmesiyle mümkündür *.

Makinanın işleyişi *devreler* halinde olur, ki bir devre
süresince bir tek ihtar yapılır. Her bir devrenin başında
yapılacak ihtarları ihtiva eden hatıra hücrelerinin muhte-
vaları kontrol kısmına getirilir. Bu halde kontrol kısmı
ihtarı yerine getirmek için gerekli bağlantıları yapar.
Bundan sonra daha sonraki ihtar kontrol kısmı tarafından
hatıradan getirilir, ve kontrol kısmı tarafından yapılır,
ilâ.; bu iş bir dur emriyle makina duruncaya kadar sürer.

Böyle bir kontrol kısmının teknik imkânı hayret
verici olmamalıdır. Esas olarak, telefon etmenin, elektrik
işaretler vasıtasıyla özel hata otomatik olarak geçirildiği
telefon santralının herhangi bir kadranındaki olandan
fazla bir tertibata ihtiyacımız olmaz.

Hesap makinalarının üç temel parçasını inceledik.

*) EDSAC ve IBM 650 gibi manyetik kaskaklı makinalar bu şekildeki
sıralı tarzda çalışmaz; bunun yerine her bir ihtar ondan sonraki
ihtarın yerini belirler.

Gerçekte başka önemli kısımlar da vardır, özellikle bilgiyi makinaya sokmağa ve makina operatörüne sonuçları iletmeğe yarayan kısımlar bunlardandır, fakat bunlar makinanın işlemesi prensiplerini inceliyecek veya onların matematik ve lojik imkânlarını açıklayacak kadar önemli değildir. Bu sebepten, bundan böyle bilginin makinaya konabildiğini ve onun doğrudan doğruya hatıra kısmı yoluyla okunabildiğini farzedeceğiz.

6. PROGRAMLAR

(Makina Algoritmaları)

Bu bölümde üç -adresli makina için yazılmış programlardan bir kaç örnek veriyoruz. Bu programlar daha önce incelenen algoritmalarından elde edilmiştir. Bunlar şu ifadenin anlamını açıklamak üzere tahlil edilmişlerdir: *Bir makinanın işlemesi ona konulan programla kontrol edilir.* Yani, işlemlerin yapıldığı emri nenin tayin ettiğini ve çoğu defa uzun işlem zincirleri ihtiva eden çözümlerin nispeten az sayıda bilgilerle nasıl kontrol edilebildiğini gösteriyoruz.

Makina ihtarlarına ait notasyon bölüm 18 de gösterilene benzer.

19. LİNEER DENKLEMLER İÇİN BİR PROGRAM

Aşağıdaki denklem sistemi için bir program yazalım:

$$ax + by = c$$

$$dx + ey = f.$$

Belirsizlikten kaçınmak için a, b, c, d ve f katsayılarının 51 le başlayan ardışık hatıra hücrelerinde saklandığını farzedelim:

*Adres**Muhtevalar*

51	<i>a</i>
52	<i>b</i>
53	<i>c</i>
54	<i>d</i>
55	<i>e</i>
56	<i>f</i>

31-50 hücrelerini de hesabın ara ve son sonuçlarına saklıyalım.

Denklemlerin çözümü

$$x = \frac{ce - fb}{ae - bd}, \quad y = \frac{af - cd}{ae - bd}$$

dir.

Çözümü elde etmek için altı çarpma, üç çıkarma ve iki bölme yapmalıdır. Bu yüzden programımız, 01 — 12 hücrelerinde saklı oniki ihtardan ibarettir:

*Adres**Muhtevalar (ihtarlar)*

01	3 53 55 31
02	3 56 52 32
03	3 51 56 33
04	3 53 54 34
05	3 51 55 35
06	3 52 54 36
07	2 31 32 37
08	2 33 34 38
09	2 35 36 39
10	4 37 39 40
11	4 38 39 41
12	0 00 00 00

İhtarlar kontrol kısmına getirilir ve birbiri ardınca yapılır. Bütün dizi tamamlandıktan sonra, 31 — 41 hücreleri aşağıdaki sayıları ihtiva eder:

<i>Adres</i>	<i>Muhtevalar</i>
31	<i>ce</i>
32	<i>fb</i>
33	<i>af</i>
34	<i>cd</i>
35	<i>ae</i>
36	<i>bd</i>
37	<i>ce — fb</i>
38	<i>af — cd</i>
39	<i>ae — bd</i>
40	<i>ce — fb</i>
	<i>ae — bd</i>
	<i>af — cd</i>
41	<i>ae — bd</i>

Burada hesaplanan bütün ara sonuçları keza nihai sonuç (40 ve 41 hücreleri) bulunmaktadır.

20. İTERASYON (ÜSTELEME)

Aşağıdaki n denklem sisteminin çözümlerini bulalım:

$$ax + by = c_i,$$

$$dx + ey = f_i, \quad i = 1, 2, \dots, n.$$

Bu probleme ait algoritma, bu tipten bir denklem sistemine ait olanının tam n defa tekrarıdır. Böylece, bölüm 19 dakinin tıpkı basit bir genişletilmesi olan $6n$

hatıra hücresinde saklı $6n$ parametresi ve toptan $11n + 1$ ihtarı bulunan bir algoritma kurmak kolay olurdu. Bu ihtarlardan ilk 11 i ilk sistemin çözümünü, ikinci 11 i ikinci sistemin çözümünü ilâ n defa hesabeder, $(11n + 1)$ inci ihtar da dur emridir.

Bununla beraber, programın hacmindeki bu büyük artış önemsenir değildir ve bundan kurtulunabilir. Kaydedelim ki 11 ihtardan ibaret her bir seri onu önceliyen serilerden ihtarlarda gözüken adreslerin değiştirilmesiyle elde edilebilir. Özellikle, şayet $6n$ parametre 7 hücresinden başlayarak sıra ile saklanırsa, o takdirde, bölüm 19 daki programda ilk altısının herbirindeki ilk iki adrese 6 yı eklemekle ikinci denklem sisteminin çözülmesine ait bir algoritma çıkar. Sonuçları ayrı tutmak için 10 ve 11 hücrelerindeki son adresleri de her seferinde 2 kadar artırmamız gerekir. Böylece bölüm 19 daki algoritmayı sekiz *adres-değiştirme* ihtarı vasıtasıyla daha genel hale genişletebiliriz. Önce 25 ve 26 hücrelerine iki parametre yerleştirelim:

<i>Adres</i>	<i>Muhteva</i>
25	0 06 06 00
26	0 00 00 02

İlave sekiz ihtar şunlardır:

<i>Adres</i>	<i>Muhteva</i>
12	1 01 25 01
13	1 02 25 02
14	1 03 25 03
15	1 04 25 04
16	1 05 25 05
17	1 06 25 06
18	1 10 26 10
19	1 11 26 11

01 - 19 ihtarları yerine getirildikten sonra, 40 ve 41 hücreleri evvelki gibi ilk denklem sisteminin çözümünü ihtiva eder, 01 - 06 ve 10 - 11 hücreleri ise aşağıdaki değiştirilmiş ihtarları ihtiva eder:

<i>Adres</i>	<i>Muhteva</i>
01	3 59 61 31
02	3 62 58 32
03	3 57 62 33
04	3 59 60 34
05	3 57 61 35
06	3 58 60 36
10	4 37 39 42
11	4 38 39 43

Böylece, şayet 01 - 19 ihtarları tekrar yerine getirilse, ikinci denklem sistemi çözülmüş ve çözüm 42 ve 43 adreslerinde saklanmış olur. Bundan başka, 01 - 06 ve 10 - 11 ihtarlarındaki adresler ondan sonraki ilmekte denklem sistemini çözecek tarzda değişmiş olur.

Bu 19 ihtarı elimizde bulunan denklem sistemi kadar tekrar ettikten (ve devamla bütün denklem sistemini çözdükten) sonra makinayı nasıl durduracağız? Bunun için 0 00 00 01 ve n (n denklem sistemi sayısıdır) parametrelerini sırayla 27 ve 28 hücrelerine yerleştiririz ve daha önce ortaya çıkarılan 19 ihtara aşağıdaki üçünü ilâve ederiz:

<i>Adres</i>	<i>Muhteva</i>
20	2 28 27 28
21	5 01 28 01
22	0 00 00 00

20 ihtarı, 01 - 19 ihtarlarının birbiri peşine tekrarıyla 28 hücrelerinin muhtevasına müncer olur. 21 ihtarı 01

adresine kontrol intikal ettiren bir sıçrama şartıdır ve 28 deki sayı pozitif olduğu sürece yerine getirilir; icra edildiği her seferde makınayı bundan sonraki denklem sistemini çözmek üzere öteki ilmeğe geçirir. Böyle n ilmekten sonra, bütün denklemler çözüldünce, 28 hücresinin muhtevası sıfıra müncer olur ve 21 ihtarının şartlı sıçraması yapılmaz. Bu takdirde makina ondan sonraki ihtara (hücre 22) geçer ki bu makınayı durdurur.

Söylenilenlerden açıkça anlaşılır ki n denklem sistemini çözmekten ibaret ortaya atılan problem 01 - 22 hücrelerinde bulunan programla çözülmüş olur. Bu programın yapısı aşağıdaki diyağramda gösterilmiştir *:

İhtarlar		Parametreler	
01	Aritmetik İşlemler	25	0 06 06 00
02		26	0 00 00 02
03		27	0 00 00 01
04		28	n
05			
06			
07			
08			
09			
10			
11			
12	Adres Değişmesi		
13			
14			
15			
16			
17			
18			
19			
20	İlmik sayımı		
21	Şartlı sıçrama		
22	Dur		

* Adres değişmesi pratikte öyle önemli olmağa yüztutmuştur ki en

21. EUCLİDE'SEL ALGORİTMA

Bundan sonra iki a ve b sayısının en büyük ortak bölenini bulmak için bir program düşüneceğiz. İlk a ve b sayılarını sırayla 12 ve 13 hücrelerine yerleştirip 14 ve 15 hücrelerini ara hesaplamalar için alıkoyuyoruz; makina durunca sonuç 15 te gözükür. Program, bölüm 1 de verilen Euclide'sel algoritmadan kurulabilir.

Adres	Muhteva	Açıklama
01	1 12 05 15	12 nin muhtavesını 15 e geçir
02	2 12 13 14	12 ile 13 ün künün farkını 14 e yerleştir
03	5 02 14 06	14 ün muhtevası negatifse 06 ya atla
04	5 01 14 09	14 ün muhtevası pozitifse 09 a atla
05	0 00 00 00	Dur
06	1 13 05 12	13 ün muhtevasını 12 ye naklet
07	1 15 05 13	15 in muhtevasını 13 e naklet
08	5 00 00 01	01 ihtarına şartsız atlama
09	1 13 05 12	13 ün muhtevasını 12 ye naklet
10	1 14 05 13	14 ün muhtevasını 13 e naklet
11	5 00 00 01	01 ihtarına şartsız atlama

İlk iki ihtar yapıldıktan sonra şunlar elde olunur:

Adres	Muhteva
12	a
13	b
14	$a - b$
15	a

büyük makinalar bunu daha otomatik olarak yapacak özel cihazlara sahiptir.

Eğer $a - b = 0$ (yani $a = b$) ise, bu takdirde 03 ve 04 şartlı nakileri atlanarak makinayı durduran 05 yerine getirilir. Bu halde 15 hücresi aşikâr olarak gerekli sonucu ihtiva eder (bölüm 1 deki 3 ihtarıyla mukayese et).

$a - b < 0$ (yani $a < b$) ise, bu takdirde 03 ihtar kontrolü 06 ya geçirir, ve bununla 07, a ile b nin 12 ve 13 hücrelerindeki yerlerini değiş tokuş eder (bölüm 1 deki 4 ihtarıyla mukayese et). O takdirde 08 ihtar kontrolü şartsız olarak 01 e tekrar nakleder ve makina diğer ilmiğe geçer.

$a - b > 0$ (yani $a > b$) ise, bu takdirde 03 ihtarına bakılmayıp 04 ihtar kontrolü 09 a nakleder. 09 ve 10 ihtarları 12 ve 13 teki a ve b sayıları yerine sırayla b ve $a - b$ sayılarını yerleştirir (bölüm 1 deki 5 ihtarıyla mukayese et). Bu halde 11 ihtar kontrolü şartsız olarak 01 e nakleder ki bundan sonra diğer ilmik başlar.

Bir ilmik dizisi, 12 ve 13 hücrelerinde bir

$$(a_1, b_1), (a_2, b_2), \dots, (a_i, b_i), (a_{i+1}, b_{i+1}), \dots$$

sayı dizisi 15 hücresinde de o kadar defa bir

$$a_1, a_2, \dots, a_i, a_{i+1}, \dots,$$

sayı dizisi meydana getirir ki 12 ve 13 hücrelerinde (a, b) eşit sayıları gözüksün. Bu halde 05 ihtar makinayı durdurur ve a sonucu 15 hücresinde bulunmuş olur.

22. HESAP MAKİNALARININ İŞLEMESİ

Evvelki üç misâl otomatik hesap makinalarının aşağıdaki iki temel prensibini açıkça aksettirir:

1. Programın ihtarları önceden ayrılan öyle bir sırayla dizilmiştir ki makina genel olarak programı yerine getirmeye devam eder. Bununla beraber makina, çeşitli ara sonuçların nevine bağlı olarak, kendi kendine hesabın akışını değiştirmeye kabiliyetlidir. Şartlı atlama ihtarları işe sokularak bu yapılır.

2. Makina, nispeten kısa bir program kullanarak oldukça uzun bir hesabı yapmağa elverişlidir, çünkü programın çeşitli kısımları otomatik olarak tekrar edilebilir veya gözden geçirilebilir. Program, sayısal olarak şifrelenmiş ve hatıra kısmında aynen diğer herhangi bir verinin şeklinde saklanmış olduğuna göre hesaplama süresinde makina tarafından kendisi de değiştirilebildiğinden (meselâ, bir kaç ihtarda adresleri artırarak veya azaltarak) bu mümkündür.

Bu prensipler aynı zamanda nevi itibarıyla (aritmetik) hesaba gelmediği bu kadar açık olan problemlere ait makinanın işleminin karakteristiğidir. Meselâ, Theseus algoritmasını (bir lâbirentte arama) veya kelime problemine ait bilinen bir algoritmayı bir takım asosyatif hesap cinsinden programlamak mümkündür. Fakat bunu yapmak için, makinanın, yukarda gözönüne alınan basit aritmetik hesapta kullanılanlarla birlikte bir kaç ilâve elemanter işlem de yapması gereklidir. Bu modern elektronik makinelerde yapılabilir, öyle ki aynı makina sadece program değiştirerek bir çok tipten problemi çözdürebilir.

23. HESAP MAKİNALARININ KULLANILIŞI

Sadece matematikte değil bir çok beşeri faaliyetlerde, şeklen tam bir ihtar dizisi halinde (algoritma) ifade edilebilmiş ve bir makina için programlanmış işlemler vardır. Meselâ, defter tutmada ve iktisadî plânlamada verilenin

tahlili ve işlenmesi ile dengenin derlenişi değişmez usüllere göre uzun basit işlemler zinciri ile yapılır. Öbür hallerde henüz kesin, mükemmel algoritmalar yoktur, bununla beraber bir gün böyle algoritmalar formülleştirilip mükemmeleştirilecektir. Meselâ bir dilden öbürüne tercüme hakkında bu gerçekleşmiştir. Herhangi iki dilde sentaksin (nahiv, cümle bilgisi), morfolojinin (kelime bilgisi) ve kelime kullanımının yeter derecede sistemli tahlili verilirse, tercümeyle ait tatmin edici bir algoritma, meselâ, bir dilden öbürüne ilmi veya ticari metinler kurmak mümkün olur. (Bazı dil çiftleri için şimdiden böyle algoritmalar elde bulunmaktadır.)

Kısaca bir makinanın başarılı şekilde bir oyun oynama imkânını düşünelim. Önce, prensip olarak bölüm 7 deki algoritmaya göre en elverişli stratejiye ait makina dilinden ihtarlar yazmak mümkündür. Uygulama alanında bu sadece epeyce küçük ağaçlı bir oyun, meselâ 4 ve 5 bölümlerinin, kibrit sayısının hayli az olduğu kibrit oyunları gibi oyunlar için mümkündür. Sonra, eğer biri şimdiden bir oyun için en elverişli bir strateji bilse bunu bir makina için programlar. Hakkında hiç bir stratejinin bilinmediği veya bilinen stratejilerin kullanılmasının kaba olduğu, karışık oyunlar halinde, bir kimse sadece oyunun kısmî tahlili üzerine kurulu metotlarla yetinmiye kendini zorlar, ki bu metotlar oyunun *taktikleri* denileni teşkil eder. Meselâ satrançta, şaha çok büyük sayıda, vezire daha az sayıda, ruhlara (kale) daha da az sayıda ilâ, paytaklar (piyade, piyon) en küçük değeri haiz olmak üzere noktalar ayırarak, parçalar (satranç taşları) için bir değer sistemi kurmak mümkündür. Bundan başka, konumların üstünlüklerine değer biçiyoruz (hareket kabiliyeti, satranç tahtası ortasının kontrolü, şahın korunması, v.s.). Beyazlara ve siyalara ait noktaların toplamı arasındaki fark oyunun her adımında beyazın konumsal üstünlüğünü

karakterize eder. En basit algoritma, verilen bir konuma ait mümkün bütün hareketlerin listelenmesi ve sonra değer sistemleri tesis edilişı görüş noktasından en büyük üstünlüğü verenin seçilişinden ibarettir. Daha iyi fakat daha karışık algoritma üç veya dört ardışık hareketin mümkün bütün kombinezonlarının incelenişinden ve bu esasta en elverişli hareketin seçilişinden ibarettir.

Söylediklerimizden, kaç çeşit zihinsel görevin belirli bir algoritmaya göre yapılabildiği daha berrak hale gelir. Bütün bu hallerde, prensip itibariyle, otomatik kontrollü bir makinaya ait işlemleri programlamak mümkündür. Özel olarak, dil tercümeleri ve satranç oyununa ait programlar yazılmış ve makinalarda başarıyla yürütülmüştür. (Bibliyografya [5] e bak.)

7. "ALGORİTMA,, NİN DAHA KESİN TARİFİ GEREĞİ

24. ALGORİTMALARIN VARLIĞI

Bundan önceki incelemelerimiz algoritmalarla hesap makinaları arasında bulunan sıkı bağı ortaya koymaktadır. Aşıkâr olarak, makinayla yapılabilen herhangi bir işlem bir algoritma şeklinde yazılabilir. Karşıt olarak, şimdiye kadar kurulmuş olan bütün algoritmalar, keza ilmin şimdiki durumunda kurulması umulabilenler, prensip itibariyle makinayla yapılabilir.

Son ifade biraz açıklamayı gerektirmektedir. Daha önce gördüğümüz gibi, algoritmanın hali hazır uygulaması genişlemeğe pek eğilimli gözükmektedir, gerekli bütün bilgilerin kayıt işi pek büyük olacağa benzemektedir. Öte yandan, makinanın hatıra kısımları sınırlı bir kapsamı haizdir (çünkü hatıra hücrelerinin sayısı sonlu ve herbirinin kapsamı sınırlıdır). Bundan ötürü, mevcut şartlar içinde bir algoritmanın, yerine getirilmesi imkânsız olmak durumuna yüz tutmaktadır.

Bu Euclide'sel algoritma ile açıklanabilir. İki sayının en büyük ortak bölenini bulmaktan ibaret olan pek basit problem eğer mevcut olandan daha fazla kâğıt ve mürekkebe ihtiyaç gösterirse elle yapılamaz. Benzer şekilde bir problem, makinada olandan daha fazla hatıra yeri gerektirirse çözülemez.

Bu hallerde, şayet sonlu sayıda adımlarla (hattâ bu sayı pek büyük te olsa) gerekli sonuca götürürse bir

algoritmaya *gerçeklenebilmesi mümkündür* deriz. Başka deyişle, sınırsız hatıra kapsamını haiz bir makinada algoritmayı kullanmak mümkün olur.

Algoritma fikriyle, sonsuz kapsamlı bir hatırası olan otomatik bir makina fikri arasındaki bağ, herbirinin daha açık anlaşılmasına yol açar. Bununla beraber, bunların bağı üzerine bütün belirttiklerimiz için, bu fikirlerin her ikisini kesin olarak henüz tarif etmiş değiliz. Algoritmalar (ve, aynı zamanda, otomatik hesap makinaları) kavramının tam matematik bir tarifi 1930'a kadar meydana getirilememiştir. Neden, bir kaç asır boyunca matematikçiler, şahsî kuruntu duymadan açık olmıyan bir algoritma kavramına gözyummuşlardır? Neden matematiksel incelemeye yeter derecede doğru bir tarif için şiddetli bir ihtiyaç ancak yakında meydana çıkmıştır?

Daha önce «algoritma» deyimini matematikte, *bir algoritmanın varlığının iddiasının daima böyle bir algoritmanın tarifiyle eşlik etmiş olduğu müşahhas algoritmalarla bağıntılı olarak geçmiştir*. Bu şartlar altında yalnızca, şekli ihtarlardan ibaret sistemin herhangi bir veriye uygulandığı zaman gerçekten aranan sonuca kendi kendine götürdüğünün gösterilmesi gerekli idi. Böylece her ne kadar her matematikçi, deyimini ne anlama geldiğine dair temel fikre sahip idiyse de, algoritma kavramının kesin tarifine ait ihtiyaç asla ortaya çıkmamıştır. Bununla beraber, matematik ilerlemenin akışında, bu durumu köklü olarak değiştiren olaylar birikmeğe başladı. Harekete getiriş kuvveti, matematikçilerin gittikçe genelleşen problem tiplerini çözmeğe gittikçe güçlü olan algoritmalar kurmak tabii isteği haline geldi.

Kare kök almağa yarıyan algoritmayı hatırlayınız. Bu problemi genelleştirmeye istiyebilirdik: verilen herhangi bir sayının herhangi kuvvetten kökünü bulmağa yarıyan bir algoritma kurmak. Böyle bir algoritma kur-

manın daha güç olacağını ummak tabiidir, fakat buna sahip olmak ilgi çekicidir. Hattâ daha ileriye gidebiliriz. Bir a sayısının n nci kuvvetten kökünü bulma

$$x^n - a = 0$$

denklemini çözme (denklemin köklerini bulma) anlamına gelmektedir. Daha da genel olan şu problemi formüle edebiliriz:

n gelişigüzel pozitif bir sayı olmak üzere

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0, \quad (*)$$

şeklindeki herhangi bir denklemin bütün köklerini bulmağa yarıyan bir algoritma kurmak *.

Böyle bir algoritma kurmak daha da güçtür. Gerçekten, denklemler teorisinin esas muhtevası tam bu algoritmayı kurmaktan ibarettir; bunun büyük önemi vardır.

25. SONUÇ ÇIKARABİLME (İSTİDLÂL) PROBLEMİ

Verilen misâller gittikçe daha genel tipten problemleri çözmeğe gittikçe daha güçlü algoritmalar bulmakta matematikçilerin tabii çabasını göstermektedir. Bittabî (*) şeklindeki bütün denklemleri çözme misâli gidilebilen sınırı göstermez. Şayet gittikçe genelleşen algoritmaları bulmağa ait bu isteği sonuna götürmek istersek şu problemi gözönüne almaktan kaçınmamamız gerekir:

Herhangi bir matematik problemin çözülmesine ait algoritma kurmak.

*) Daha kesin olarak, herhangi k tam sayısı için, $1/10^k$ den daha yakınlıkla doğru olan köklerin ondalık yaklaşımını bulmak.

Bu öyle bir genel problemidir ki bütün olarak matematığe küstah bir meydan okuma sayılmıştır. Bu arada «herhangi bir matematik problem»in ne anlam taşıdığının açık olmadığı hususu üzerinde tenkit yürütülebilir. Aynı zamanda böyle bir problemin büyük ilgi çekiciliği şüphe götürmez.

Bu problemin kendine has hikâyesi vardır. Büyük Alman matematikçisi ve felsefecisi Leibnitz (1646-1716) herhangi problemi gözmekte tam şümülü bulunan bir metot tasarladı. Gücü dışında olmakla beraber, onun keşfedileceği zamanın geleceğini ve o zaman matematikçiler arasında herhangi bir muhakemenin kendi kendine kalem ve kâğıtla düzenleneceğini düşündü.

Daha sonra problem, matematiksel lojiğin en ünlü problemlerinin biri, sonuç *çıkarabilme (istidlâl) problemi* halinde bir incelik almıştır. Problemin tam ele alınışı için yerimiz olmadığından, bunun daha ziyade genel hatlarını çizeceğiz.

Bilindiği gibi, matematikteki aksiyomatik metot verilen bir teorideki bütün teoremlerin ispatsız kabul edilen belirli aksiyonlardan şekli adımlarla türetilmesinden ibarettir. Bütün aksiyomatik teorilerin ilki geometri idi, fakat hemen hemen modern matematikte bütün teoriler aksiyomatik olarak kurulmuştur. Matematiksel lojik, matematiksel teorinin herhangi bir önermenin tek şekilde belirli bir formül olarak yazılmasını bize mümkün kılan özel bir «formüller lisanı» kullanmaktadır.

Daha önce asosyatif hesap için (bölüm 11) kullanmış olduğumuz dille, böyle bir formülün özel alfabede, olumsuz, bağlaç, kapsam gibi lojik işlemleri göstermeğe yarıyan semboller, keza harfler ve parantezler gibi, fonksiyon ve değişkenleri göstermeğe yarıyan alışılan matematik semboller ihtiva eden bir kelime olduğunu söyleyebiliriz. Bununla beraber, asosyatif bir hesaba esas benzerlik, bir

asosyatif hesaptaki uygun süstitüsyonlara pek benzer bir tarzda, bir R birinci önermesinden (prömis, öncül) bir S ifadesinin lojik türetilmesinin şekli kelime dönüşümleri halinde yazılması imkânındadır. Bu bize, kendisinden keyfi kompleksiteli herhangi bir lojik istidlâlin kurulabildiği, lojik dedüksiyonun basit etkilerini gösteren uygun dönüşüm sistemi bulunan bir *lojik hesapta* bahsetmiye müsaade eder. Böyle bir dönüşüme misâl bir formüldeki iki ardışık olumsuzun yokedilmesidir; böylece, «ispat edilmemiş», «ispat edilmiş» haline dönüşebilir (bunu bölüm 14 teki misâlde $aa - A$ müsait süstitüsyonu ile mukayese ediniz).

Lojik bir hesapta R ilk önermesinden S önermesinin lojik olarak çıkarılabilmesi sorusu, R yi gösteren kelime-den S yi gösteren kelimeye götüren geçilebilme zincirinin varlığı problemi haline gelir. Şimdi artık sonuç çıkarabilme problemi aşağıdaki gibi formülleştirilebilir:

Bir lojik hesapta herhangi iki A ve S kelimesine (formülüne) ait R den S ye geçilebilme zincirinin mevcut olup olmadığını tayin etmek.

Çözüm, bu tipten (R ve S gelişi güzel) herhangi bir problemin çözülmesine ait bir algoritma olacak diye farzedilmiştir. Böyle bir algoritma, aksiyomatik olarak kurulmuş olan bütün matematik teorilerdeki (veya daha ziyade, *sonlu şekilde aksiyomlaştırılabilir* teorilerdeki) problemlerin çözülmesine ait genel bir metot verir. Böyle bir teoride herhangi bir S ifadesinin muteber oluşu yalnız, bunun bir aksiyom sisteminden çıkarılabildiği, veya aynı şey demek olan, bütün aksiyomların cari olduğunu farzeden R ifadesinden çıkarılabildiği anlamına gelir. Bu halde algoritmanın uygulanışı S önermesinin muteber olup olmadığını tayin eder. Bundan başka, eğer S önermesi cari olsa idi, bu halde lojik hesapta buna karşılık gelen bir geçilebilme zinciri bulabilir ve bundan önermeyi ispat edecek bir istidlâl zinciri elde edebilirdik. Öne sürülen

algoritma, gerçekten formüleleştirilebilmiş ve bu güne kadar çözülmemiş hemen bütün matematik problemlerin çözülmesine ait etkili tek metot olurdu. İşte neden böyle bir «tam şümüllü algoritma» ve bunun eşi «her işe gücü yeter bir makina» kurmanın, bu kadar ümit verici ve aynı zamanda bu kadar güç oluşu anlaşılıyor.

Büyük adamların bu kadar uzun ve dirençli şabalarına rağmen, böyle bir algoritma bulma güçlükleri aşılamaz kalmıştır. Ayrıca, daha sonra çok daha az genel tabiatından bir kaç probleme ait algoritmalar bulmaya çalışmada benzer güçlüklerle karşılaşmıştır. Bunlar arasında Diophant denklemleri üzerine Hilbert problemleri (bölüm 3), keza yukarıda incelenmiş olan daha başkaları vardı.

Böyle algoritmalar kurmakta girişilen bir çok verimsiz teşebbüsün sonucu olarak, işe karışan güçlüklerin temelli olduğu aydınlanmış oldu ve *her sınıftan probleme ait bir algoritma kurmanın mümkün olmadığı* şüphe götürür oldu.

Belirli bir problem sınıfının algoritmik olarak çözülmediği iddiası basitçe henüz hiç bir algoritmanın keşfedilmediği anlamını taşımamaktadır. Gerçekte böyle bir algoritmanın *hiç bir zaman* keşfedilemeyeceği anlamını taşımaktadır, başka sözlerle, böyle bir algoritmanın mevcut olmadığı anlamına gelmektedir. Bu iddia bazı nevi matematik ispatlar üzerine kurulmuş olmak gerekir; bununla beraber, böyle bir ispat, kesin bir «algoritma» tarifine sahip oluncaya kadar hiçbir anlam ifade etmez, çünkü neyin imkânsız olduğunu ispata çalışacağımız aşikâr olmayacaktır. Yeri gelmişken matematik tarihinde çözümleri uzun zaman boşuna aranmış ve çözümlerin elde edilemeyeceği daha sonra ispatlanmış başka problemlerin mevcut bulunmuş olduğunu hatırlamak faydalıdır. Misâlleri, bir açıyı üç eşit parçaya bölme problemi ile Genel beşinci derece denkleminin köklerle çözülmesi problemidir.

Bir açının pergel ve cetvelle iki eşit parçaya bölünmesi metodu her öğrenci tarafından bilinmektedir. Eski yunanlılar pergel ve cetvel kullanarak bir açıyı üç eşit parçaya bölmeğe çalıştılar. Daha sonra ise keyfi bir açının böyle araçlarla üç eşit parçaya bölünmesinin imkânsız olduğu ispatlandı. Keza bir ikinci derece denkleminin aritmetik işlemlerine ait işaretler ile kök işareti kullanan bir formül yardımıyla katsayıları cinsinden yazılabildiği iyice bilinmektedir. Üçüncü ve dördüncü derece denklemi için son derece karışık köklü formüller vardır. Dörtten fazla dereceli denklemlerin köklerle benzer çözümlerine ait araştırma, ondokuzuncu asrın başlangıcına kadar başarısız şekilde getirildi, ki o zaman nihayet aşağıdaki önemli netice kuruldu.

5 ten büyük veya buna eşit herhangi bir n için, genel n nci dereceden denklemin köklerini, katsayıları cinsinden aritmetik işlemler ve kök alma işlemiyle ifade etmek imkânsızdır.

Bu hallerin her ikisinde de imkânsızlığının ispatı sadece, «Pergel ve cetvelle çizim ne demektir?» ve «Bir denklemi köklerle çözmek ne demektir?» sorularına kesin tarifler var olduktan sonra mümkün olmak durumuna düşmüştür. Bu iki tarifi bazı özel algoritmalara, yani bir denklemin köklerle çözülmesine ait (denklemlerin genel olarak çözülmesine ait değil) bir algoritma ile bir açının pergel ve cetvelle üç eşit parçaya bölünmesine ait (keyfi aletlerle üç eşit parçaya bölünmesine ait değil) bir algoritmaya daha kesin bir anlam vermiş olduğunu kaydedelim.

26. «ALGORİTMA» TARİFİNİN FORMÜLLEŞTİRİLMESİ

Yakın zamana kadar «algoritma» kavramının kesin bir anlamı yoktu ve bundan ötürü böyle bir tarifi kuruluması modern matematiğin en modern problemlerinden

biri olmuş oldu. Bir «algoritma» tarifi (veya diğer herhangi bir matematik tarifi) formülleştirilmesinin, «algoritma» kelimesinin anlamının ne olabileceğinin tesbiti gibi matematikçiler arasındaki sadece keyfi bir uygunluk olarak gözönüne alınması gerektiğinin belirtilmesi çok önemlidir. Tarif, müphem olmakla beraber gerçekte doğru olan ve şimdiye kadar bir çok misâle açıklanmış bulunan bu fikirlerin esasını doğru şekilde aksettirmelidir. Bu maksatla algoritmalar kurulmasında gerçek olarak kullanılmış bulunan bütün metotları karakterize etmek için 1930 da başhyan bir düziye incelemeye girişilmişti. Mesele, sadece şeklen değil daha önemlisi esasen tam olacak bir *algoritma* kavramının bir tarifini formülleştirmekten ibaretti. Bu konuda çalışan bir çok kimseler lojik bakımından farklı hareket noktalarından işe başladılar, ve bu yüzden, bir çok tarifler teklif edildi. Bununla beraber, bütün bunların denk olduğu ve aynı kavramı tarif ettiği sonucuna varıldı; bu modern *algoritmanın* tarifi idi. Görünüşte farklı olan bütün bu tariflerin, gerçekte esas itibariyle aynı oluşu hayli manidardır; bu değer taşıyan bir tarif elde ettiğimizi gösterir.

Makina matematiği görüş noktasından, özellikle *makinalarla yapılabilir işlemlerin gözönüne alınışından meydana gelen tarif şekliyle* ilgilendik. Böyle ciddi matematik tarif için makinanın, mümkün olduğu kadar basit bir lojik yapıda olan, fakat matematik incelemelerde kullanılmak üzere yeter derecede kesin olan standart bir şema halinde işlemlerini göstermek zaruridir. Bu iş, ilk defa bir hesap makinasının pek genel fakat pek sade bir kavramını öne süren İngiliz matematikçisi Turing tarafından yapılmıştır. Şunu kaydedelim ki Turing makinası ilkin 1937 de, yani modern hesap makinalarının kurulmasından önce açıklanmıştır *. Turing sadece, bir maki-

*) Eserin sonundaki bibliyografyada [1] e bakınız.

nanın işlemlerini, belirli ihtarları takip eden insan kalkülâtörünün işine eşitleme genel fikri üzerinde durmuştur. Bizim onun fikrini ortaya koymamız şimdi kullanılmakta olan elektronik makinalar genel fikrinde fayda sağlayacaktır.

8. TURING MAKİNASI

Turing makinasının, bölüm 5 ve 6 da anlatılan elektronik makinalarla mukayese edildikte ayıredici özellikleri şunlardır.

1. Turing makinasında bir işlemin basit işlemlere indirgenmesi belirli bir yönde sınır haddine vardırılır. Böylece, meselâ elektronik makinada bir tek işlem olarak gözönüne alınan toplama işlemi daha çok basit işlemlere bölünür. Bu, tabiatıyla makina tarafından yapılan hesaplama adımları sayısını önemli bir miktar artırır, ama aynı zamanda lojik yapıyı sadeleştirir, teorik alıştırmalar için büyük bir imkân sağlar.

2. Turing makinasında hatıra kısmı * her iki yönde sonsuz uzun ve hücrelere bölünmüş bir bant (teyp) olarak düşünülmüştür. Tabii hiç bir makinanın sonsuz hatırası olmaz ve Turing makinası, bir makinanın, hatıra kaplam haddini artırma imkânını aksettiren bir idealleştirilmesidir. Bu idealleştirme daha önce (bölüm 24) otomatik makinalarla, gerçekleştirilmesi mümkün algoritma kavramı arasındaki bağ üzerinde incelememizde doğrulanmıştı.

27. TURING MAKİNALARININ TARİFİ

Şimdi bir Turing makinasının tafsîlâtına girişiyoruz.

*) Özellikle dış hatıra denilen kısım.

1. Makinanın beslendiği bilginin şifrelendiği *dış alfabe* denilen sonlu bir

$$s_1, s_2, \dots, s_k$$

sembol cümlesi vardır. Genelliği bozmadan, dış alfabenin sembolleri arasında bir *boş harfin* meselâ s_1 in mevcut olduğunu farzedebiliriz. Boş harfin bir hatıra hücreesine konuluşu burada daha önce gözüken sembolü oradan çıkarır ve boş bir harf ihtiva eden hücrenin de artık boş hücre olduğunu söyleriz.

Her bir hücre en az bir sembol ihtiva edebilir. Her bilgi parçası boş harfi bulunmuyan ve bant üzerinde ardışık hücrelerde saklı bulunan sonlu bir sembol zinciriyle gösterilebilir. İlk bilgi banda sokulur. Makina o zaman çalışmağa başlar ve devrelerinin herbirinin oluşu sırasında ilk bilgi ara bilgiye dönüşür. Her bir devrenin sonunda bant üzerindeki bütün bilgiler bu merhaledeki ara bilgiyi toplar. İlk bilgi, hatıra hücreleri arasında herhangi bir tarzda dağılmış dış alfabedeki herhangi sonlu bir semboller sistemi (alfabedeki herhangi bir kelime) olabilir. Bununla beraber, hangi $\mathfrak{A}$ ilk bilgisi verildiyse ona bağlı olarak iki hâl mümkündür.

a) Sonlu sayıda bir devirden sonra bir dur emri alarak makina durur ve bant üzerinde bir $\mathfrak{B}$ bilgisinin temsili gözüktür. Bu halde makinanın, $\mathfrak{A}$ ilk bilgisine *uygulanabilir* olduğunu ve bunu $\mathfrak{B}$ sonuç bilgisi haline dönüştürmüş olduğunu söyleriz.

b) Dur emri hiç alınmaz ve makina hiç durmaz. Bu halde makina $\mathfrak{A}$ ilk bilgisine *uygulanamazdır* deriz.

Eğer bir makina verilen bir sınıftan herhangi bir problemi (sabit bir şifreyle) temsil eden bilgiye uygulanabilir ise ve bu bilgiyi, çözümü (aynı şifreyle) temsil eden bilgi haline dönüştürebilirse *bu makina bu verilen problemler sınıfını çözebilir* deriz.

2. Makinanın bölüm 5 ve 6 daki üç adresli ihtar forması yerine Turing makinası (şimdiki bazı makinalar gibi) bir adresli forma kullanır; yani, bir hesaplamanın herhangi bir adımında sadece bir hatıra hücresi rakamları kullanır. (Verilen bir Turing makinası ihtarında bulunan tek bir hatıra hücresine bu adımın *taranmış hücresi* denir.) Meselâ β ve γ daki sayıların toplanmasına ve sonucun δ da saklanmasına ait üç adresli ihtar yerine aşağıdaki üç ihtar dizisi konulabilir:

- a) β nın muhtevasını toplayıcıya geçir.
- b) γ nın muhtevasını toplayıcıya geçir.
- c) Toplayıcının muhtevasını δ ya geçir.

Turing makinasında basit işlemler sistemi ile bir adresli ihtarlar sistemi daha da çok sadeleştirilmiştir. Her bir devrede ihtar taranmış hücrede bulunan s_i sembolünü başka bir s_j sembolüne değiştirir. Şayet $i = j$ ise o takdirde taranmış hücredeki sembol değişmez; şayet $j = 1$ ise, o takdirde taranmış hücredeki sembol sadece ortadan kalkar. Ayrıca, makinanın bir devresinden ondan sonrakine geçilmesinde taranmış hücrenin adresi 1 den fazla değişmez. Yani, bundan sonraki taranacak hücre ya sağa doğru bir hane (espas) ötedeki hücre, ya sola doğru bir hane berideki hücre veya aynı hücredir.

Şayet özel bir hücrenin muhtevası hesaplamada gerekirse, makina doğrusunu buluncaya kadar bütün hücreleri teker teker yokluyarak arar. Bu işlemeyi epeyce uzatır, ama aşağıdaki kolaylığı sağlar: programın ihtarlarında taranmış hücreye ait gelişigüzel bir adres yerine aşağıdaki sembollerle gösterilen üç standart adresle yetiniriz:

- R — daha sonraki hücreyi sağa tara,
- L — daha sonraki hücreyi sola tara,
- S — aynı hücreyi tekrar tara.

3. Hafıza (hatıra) hücresinin ihtiva ettiği sayısal bilgiyi işlemek için, Bölüm 5 ve 6 da anlatılan elektronik makinanın bir aritmetik kısmı vardır, bu kısım şu sonlu sayıda halden herhangi birinde olabilir: toplama hali, çıkarma hali, v.s. Herhangi bir işlemi aritmetik kısımda yapmak için sadece üzerinde işlem yapılacak sayıların değil bu kısmı özel işleme başlatan işaretlerin de geçirilmesine ait yollar kurulmalıdır. Bu Şek. 10 b de açıklanmıştır. Turing makinasında bilginin işlenişi yine sonlu sayıda haller şeklinde yazılabilen bir *Lojik kısmıyla* yapılır. Bu halleri göstermek üzere ortaya koyduğumuz özel semboller

$$q_1, q_2, \dots, q_m$$

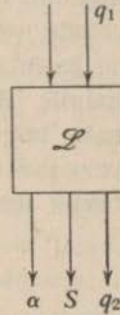
olsun. Kısım iki giriş hattını haizdir: bunlardan biri boyunca taranmış bulunan kare üzerindeki s_1 sembolü gelir, öteki boyunca da akım devresi süresinde kısmın bulunduğu halin q_1 sembolü gelir. O takdirde kısım iki s_1 ve q_1 girişiyle tek şekilde belirli olan bir s_1 «dönüşmüş» işaretini, onun çıkış hattı yoluyla taranmış bulunan hücre haline geçirir. Makinanın her bir devirde ne yapacağını belirliyen ihtarlar şu şekillerden biri halindedir:

$$Rq_l, Lq_l, Sq_l \quad (l = 1, 2, \dots, m),$$

burada ilk sembol taranacak karenin adresini belirler (yukarıya bakınız), ikinci de lojik kısmı kendi özel yerine koyar. $R, L, S, q_1, q_2, \dots, q_m$ sembolleri makinanın iç *alfabesini* meydana getirir.

s_1 çıkışına ilâveten, lojik kısım, daha sonraki devreye başlamak için hangi ihtarın kontrol kısmına konacağını tayin eder. Böylece, lojik kısım, sadece banda alınacak s_1 sembolüne ait bir çıkış hattına değil fakat aynı zamanda daha sonraki ihtar kontrol kısmına geçiren iki çıkış

hattına da malik olmak zorundadır (bak. Şek. 11). s_i, P, q_i üçlü çıkışının sadece akım devresinin s_i, q_i sembollerinin *giriş çiftine* bağlı olduğu önemlidir *. Bu, lojik kısmın, bir üçlü s_i, P, q_i sembolünü her s_i, q_i sembol çiftine birleştiren



Şekil: 11

bir fonksiyon gibi olduğu anlamını taşımaktadır (hepsi böyle $k \cdot m$ çift vardır). Bu fonksiyonu (buna makinanın *lojik fonksiyonu* denir), bir sütunu her bir hal sembolü ve bir satırı dış alfabenin her bir sembolü olmak ve çıkış üçlüsü, giriş çiftinin satır ve sütununun birleştiği yerde bulunmak üzere dik dörtgen bir tablo olarak gösterebiliriz. Bu tabloya makinanın *işleme (fonksiyonel) matrisi* diyeceğiz; böyle bir matrise misâl Şek. 12 de açıklanmıştır **.

	q_1	q_2	q_3	q_4	q_5
$\wedge$	$\wedge R q_4$	$\wedge L q_3$	$\wedge R q_1$	$\wedge S q_5$	$\wedge S q_5$
$ $	$\alpha S q_2$	$\beta S q_1$	$ R q_1$	$ L q_1$	$ S q_5$
α	$\alpha L q_1$	$\alpha R q_2$	$ L q_3$	$\wedge R q_4$	$\alpha S q_5$
β	$\beta L q_1$	$\beta R q_2$	$\wedge L q_3$	$ R q_4$	$\beta S q_5$

Şekil: 12

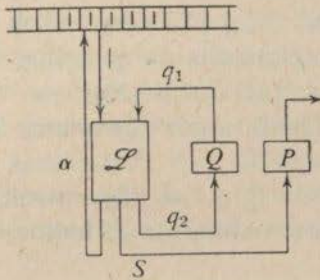
*) P, R, L, S sembollerinden herhangi biri olabilir.

**) Şimdilik dış alfabenin $\wedge, |, \alpha, \beta$ sembollerinden ibaret olduğunu farzedelim, burada $\wedge$ boş harfi göstermektedir.

28. TURING MAKİNASININ İŞLEMESİ

Turing makinasının işleminin bunun işleme matrisi ile tamamen belirlendiği aşikârdır, o şekilde ki aynı matrisli iki Turing makinası yaptıkları şeyler hususunda birbirinden ayırdedilebilirler.

Bir makinanın yapısı ile işlemesi ve çeşitli kısımlarının birbirine etkisi Turing makinalarına ait genel *yapı* diyagramından anlaşılabilir (Şek. 13).



Şekil: 13

Diyagram hatıranın bölümünü içeri ve dışarı aksettirir. Dış hatıra, içinde bilginin saklanacak dış alfabenin sembolleri halinde şifrelendiği sonsuz bant üzerinde hücrelerle gösterilir. İç hatıra, daha sonraki bilginin saklanmasına ait iki hücreden ibarettir: hal sembolü Q hücresinde ve bant hareketi de P hücresinde saklanır. Bir devrenin başında lojik kısım tarafından çıkartılan P , q_1 sembolleri kontrol kısmına nakledildikleri zaman daha sonraki bir devrenin başlangıcına kadar bu iki hücrede saklanır. Kontrol kısmının görevi P sembolü tarafından belirlenen doğrultuda bandı bir hane = espas (veya hiç) hareket ettirmeğe müncer olur. Hal işareti, geri getirme hattı denilen hattı teşkil ederek Q hücrelerinden $\mathcal{L}$ hücre-

sine nakledilebilir, ki bununla $\mathcal{L}$ tarafından önceki devrede çıkartılan sembol $\mathcal{L}$ ye geri getirilir.

Turing makinasının işlemesi aşağıdaki gibidir. Harekete başlamazdan önce, ilk bilgi banda konur ve özel bir hücre (Şek. 13 de sağdan dördüncü stroku ihtiva eden hücre) «taranma durumuna» gelir ve ilk halle bant hareketinin ilk doğrultusunun sembolleri (yani q_1 ve S) Q ve P hücrelerine konur. Bu duruma gelince makina işleme matrisleriyle belirli tek bir tarzda kendi kendine işler. Meselâ Şek. 12 deki işleme matrisi halinde ne olup bittiğine göz gezdirelim.

Birinci devre. $(|, q_1) \rightarrow \alpha Sq_2$: strok yerine α konur, bant hareket etmez makina da q_2 haline değişir.

İkinci devre. $(\alpha, q_2) \rightarrow \alpha Rq_2$: α değişmez, sağda daha sonraki hücre taranma durumuna gelir ve makina q_2 halinde kalır.

Üçüncü devre. $(|, q_2) \rightarrow \beta Sq_1$: strok yerine β konur, bant hareket etmez makina da q_1 haline değişir.

İlâ.

Şek. 12 de son sütundan da görülebildiği gibi, ancak q_5 hali işlemin bir merhalesinde meydana getirilirse makina durur. Bu olursa, taranmış sembol değişmez ve makina aynı hücreyi aramakta devam eder ve aynı (q_5) halinde kalır. Bu hesabın sonuna işaret eden bir *durma-halidir*.

Turing makinasının işletilmesi elle bile yapılabilir. Bunu yapmak için *görünüş (konfigürasyon)* kavramını kullanmak elverişlidir. k nci görünüşle k ninci devrenin başlangıcında bandın bir diyagramıyla, bu devre süresince lojik kısmın halini gösteren taranmış hücrenin aşağısında yazılı sembolü kastediyoruz. Bu şekilde, bu devredeki semboller giriş çiftini açıkça gösterir ve işleme matrisine bakarak $(k+1)$ nci görünüşü belirleyen çıkış üçlüsünü elde ederiz.

Yukardaki misâlde, birinci ve ikinci görünüşler sırayla q_1 ve αq_2 giriş çiftleriyle şunlardır:

q_1									
				α					
q_2									

Birinci görünüşten ikincisine geçiş Şek. 12 de $|q_1$ giriş çiftine karşılık gelen $\alpha S q_2$ üçlüsüyle yapılır.

Bundan böyle işleme matrisini daha sade ve daha elverişli tarzda yazacağız. $s_i P q_i$ çıkış üçlüsünü daima tam şekilde yazmayıp, şayet buna karşılık gelen giriş sembolünün aynı ise s_i veya q_i sembolünü hafzedeceğiz ve S sıfır bant hareketini göstermiyeceğiz. Bu halde bir dur haline karşılık gelen tamamiyle boş sütunlar elde ederiz. Bu takdirde böyle sütunlar Şek. 12 deki işleme matrisini gösteren Şek. 14 de olduğu gibi bu basit gösterişte matris-

	q_1	q_2	q_3	q_4
$\wedge$	$R q_4$	$L q_3$	$R q_1$	!
	αq_2	βq_1	$R q_1$	$L q_1$
α	L	R	L	$\wedge R$
β	L	R	$\wedge L$	$ R$

Şekil: 14

ten atılır, burada «!» dur halini göstermektedir. (Bundan sonra dur halini göstermek üzere daima «!» işaretini kullanacağız.)

Makina q_1 halinde α sembolünü tararsa, bunun, α veya β nin bütün geçişlerini, bunları değiştirmeden ve q_1 halinde kalarak aşp, bir strok veya boş bir hücreyle karşılaşınca kadar, sola doğru bir düzeye harekete

başlıyacağını Şek. 14 te görmek, Şek. 12 de görmekten daha kolaydır. Yalnız, bu halde makina farklı bir duruma girer.

Bundan sonraki bölümde bazı basit sayısal algoritmalar yapan Turing makinalarının kuruluşunu inceliyeceğiz. (Daha önceki bölümde söylenilenlere uygun olarak, bir Turing makinasının kuruluşuyla basitçe, böylece algoritmaya ait standart form teşkil eden kuruluşu kastediyoruz.) Keza Turing makinalarının (işleme matrisleri) kuruluşunu ilgilendiren bazı daha genel hususları da inceleyeceğiz.

9. TURING MAKİNALARINDA ALGORİTMALARIN GERÇEKLENİŞİ

29. ONDALIK GÖSTERİLİŞTE n Yİ $n + 1$ E ÇEVİRME

Aşağıdaki problem çözülecektir:

Ondalık şekilde pozitif bir n tam sayısı verildiğine göre, $n + 1$ sayısının ondalık gösterilişini bulmak.

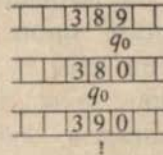
Dış alfabe için Λ boş sembolünü ve on tane 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 rakamını alalım. Makina şu iki halin birinde olabilir: q_0 (işleme hali) ve ! (durma hali). Verilen n sayısı keza $n + 1$ sonuç sayısı, ardışık hücrelerdeki bir düzeye rakamlar vasıtasıyla (hücre başına bir harf) ondalık sistemde yazılabilir. Şayet (anlamları kısaca açıklanacak olan) son sütun ve son satıra bakmazsak işleme matrisi Şek. 15 te verilmiş olur. Başlangıçta makina,

	q_0	q_1
0	1 !	
1	2 !	
2	3 !	
3	4 !	
4	5 !	
5	6 !	
6	7 !	
7	8 !	
8	9 !	
9	0 L	
$\wedge$	1 !	
	L	$\wedge Lq_0$

Şekil: 15

n sayısının en sağdaki rakamını (yani, birler basamağındaki rakamını) taramağa koyulur ve q_0 halinde bulunur. Şayet bu sayı 9 dan küçükse, makina, işleme matrisinde gösterildiği gibi bu rakam yerine bundan sonraki daha yüksek rakamı koyar ve durur. Şayet rakam 9 ise, o takdirde makina onu sıfıra çevirir ve, yine q_0 halinde kalarak bir espas sola hareket eder (bu daha sonraki rakama bir 1 rakamı gerektirmektedir). Şayet sayı k dokuzla sonlanırsa, makina $k + 1$ devreden sonra durur.

Şek. 16, $n = 389$ a ait görünüşleri göstermektedir. Şimdi genişletilmiş Şek. 15 tablosunun anlamını açıklayalım. Bu, q_1 ilâve halini ve dış alfabe'deki bir $|$ ilâve sembolünü haiz olan bir işleme matrisidir. Eğer makina



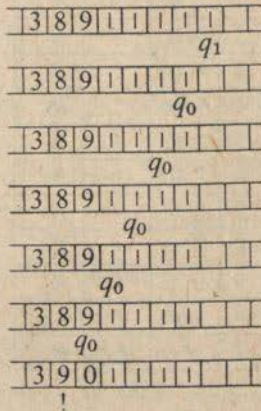
Şekil: 16

başlangıçta q_0 halinde olur ve $|$ sembolü bant üzerinde hiç bir yerde gözükmese, o takdirde makinanın işlemesi tamamen son misâldeki gibi devam edecektir. Bunun böyle olacağı, bu şartlar altında tablonun son satır ve son sütununun hiç bir rol oynamıyacağı keyfiyetinden açıkça anlaşılır.

Makina, yukarda gözönüne alınan problemlerin çözülmesiyle birlikte başka işler de yapar. Bandın bir n sayısının ondalık gösterilişiyle n nin hemen sağında bir kaç strok ihtiva ettiğini farzedelim. İlkbaştan makina en sağdaki stroku taramağa başladığına ve q_1 halinde bulunduğuna göre makinanın bu işleme matrisiyle çalışmasını gözönüne alalım. Birinci devrede (q_1 giriş çifti), strok ortadan kalkar ve makina bir espas sola kayar ve q_0 haline

(ALq_0 çıkış üçlüsü) gelir. Bu halde makina n nin en sağdaki rakamına gelinciye kadar (q_0 halinde oldukça ve bu halde kaldıkça diğer strokları bırakarak) sola kaymaya devam eder. Bu durumda, makina basitçe evvelki algoritmayı yerine getirir; yani, n sayısını $n + 1$ haline dönüştürür ve bu yapılmıca durur.

Kısaca, makina strok sayısını bir indirger ve n yi $n + 1$ haline çevirir. Bu işleme n nin ondalık gösterilişinden $n + 1$ in ondalık gösterilişine *kontrollü geçişi* diyoruz. Şek. 17 beş strok ve $n = 389$ için görünüşleri göstermektedir.



Şekil: 17

30. ONDALIK GÖSTERİLİŞ HALİNE DEĞİŞTİRME

Bir makina (bir algoritma) için aşağıdaki problem sınıfını çözmekte kullanılan işleme matrisini kuracağız: *Ardışık hücrelerde yazılı sonlu bir stroklar cümlesi verildiğine göre (böyle bir stroklar topluluğuna bir stroklar zinciri denir), stroklar sayısını ondalık şekilde yazınız.*

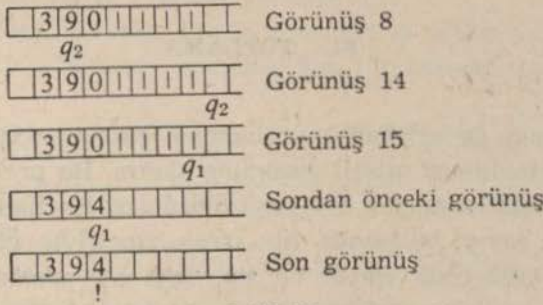
Böyle bir matris Şek. 18 de verilmiştir. Bu tablonun gerçekte gerekli makinayı (algoritmayı) gösterdiğine bir kimseyi inandırmak için, bunu Şek. 15 le mukayese etmek faydalıdır. Şek. 18 de q_0 sütunu, «!» için yeni bir q_2 hali konmuş olmakla Şek. 15 te buna karşılık gelen sütunun aynıdır. q_1 sütunundaki fark Şek. 15 teki matrisin işlemesine bir dayanak teşkil etmez. Bundan ötürü, eğer bant, bir stroklar zincirini takip eden bir n sayısının ondalık gösterilişini ihtiva ederse ve ilk şartlar son misâldeki gibi

	q_0	q_1	q_2
0	1 q_2	!	R
1	2 q_2	!	R
2	3 q_2	!	R
3	4 q_2	!	R
4	5 q_2	!	R
5	6 q_2	!	R
6	7 q_2	!	R
7	8 q_2	!	R
8	9 q_2	!	R
9	0 L	!	R
$\wedge$	1 q_2	!	Lq_1
$\downarrow$	L	$\wedge Lq_0$	R

Şekil: 18

ise (yani, makina q_1 halinde ise ve en sağdaki stroku tararsa), o takdirde makina Şek. 15 tekinin aynı işlemi icra etmekle başlar; yani, en sağdaki stroku siler ve n i $n + 1$ haline değiştirir. Bununla beraber, bu halde Şek. 15 teki makina ! haline gider ve işlem sona ererdi, halbuki Şek. 18 deki makina q_2 haline girer ve işlem devam eder. Özellikle, ilk görünüş Şek. 19 daki gibi olur. Şekil 18 deki q_2 sütunundan işlemin nasıl devam edebildiğini görebiliriz: makina bütün rakamları ve strokları atlıyarak boş bir

hücreye gelinceye kadar sağa kaymağa başlar (Şek. 19, görünüş 14). Aq_2 giriş çifti makinanın q_1 haline gitmesine ve sola kaymasına sebeb olur (Şek. 19, görünüş 15). Şimdi makina en sağdaki stroku tarar ve q_1 halinde bulunur. Bu birincinin benzeri öbür ilmiğe ait ilk şartları koydurur. İkinci ilmikte, diğer strok silinmiş olur ve $n + 1$ sayısı $n + 2$ haline değişir. Şayet başlangıçta k strok varsa, bu halde k ilmikten sonra strokların hepsi silinmiş olur ve n sayısı yerine $n + k$ konur. k nci ilmik sonunda makina yeniden q_1 halinde kalacak ve taranan hücre bir strok



Şekil: 19

değil (çünkü hepsi silinmişti) fakat $n + k$ sayısının en sağdaki rakamını (Şekil 19 daki sondan bir önceki görünüş) ihtiva edecektir. Şek. 18 e bakmakla bunun makinayı durdurduğunu görürüz (Şek. 19, son görünüş).

Söylediklerimizden, şayet bant başlangıçta 0 rakamı ile k strok zincirini ihtiva ederse, o takdirde makina strokların hepsini siler ve 0 ı k nin (strok sayısı) ondalık gösterilişine çevirir. Fakat, q_0 ve q_1 hallerinde makina boş bir hücreye 0 rakamına yaptığıının aynı işlemi yaptığından, hiç sıfır rakamı olmadan ama sadece stroklar zinciriyle başlarsak aynı şey doğru olur (bak. Şek. 18). Bundan ötürü Şek. 18 de verilen tablo gerçekten, herhangi

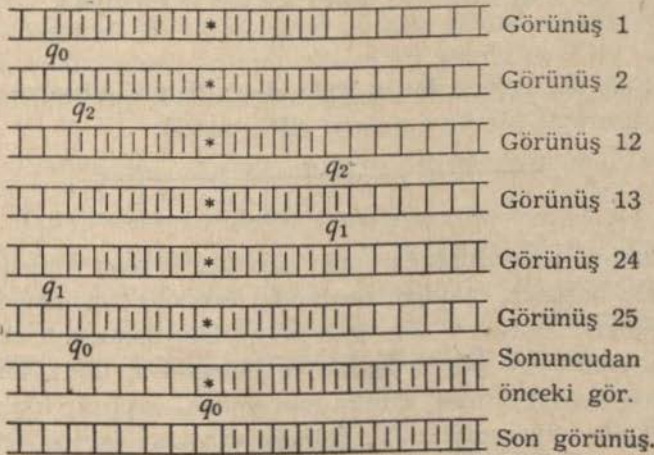
stroklar bir zincir teşkil etmezdi. Şek. 20 bir makinanın iki sayıyı toplayacak işleme matrisini göstermektedir.

	q_0	q_1	q_2
$\wedge$	$\wedge R q_2$	L	R
$*$	R	$R q_0$	q_1
	$\wedge !$	L	R

Şekil: 20

İlk durumlar: Makina q_0 halindedir ve en soldaki stroku tarar (Şek. 21 deki görünüş 1).

Aşağıdaki devreler bütün strokları ve yıldızı atlıyarak (q_2 sütununa bakınız), makina boş bir hücreyi tarıyana



Şekil: 21

kadar sola kayar (görünüş 12). Daha sonra ($\wedge q_2$ giriş çifti) makina boş hücrede bir strok işaretler ve q_1 haline gider (görünüş 13). Makina, bütün strokları ve yıldız işaretini atlıyarak yeniden bir boş hücre tarayacağına kadar (görünüş 24) sola kayar. Makina o zaman ($\wedge q_1$ giriş çifti)

en soldaki stroku taramak üzere sağa kayar ve q_0 haline gider (görünüş 25). Bu ilmeğin neticesi olarak en soldaki strok, yıldız işaretinin sağında bulunan zincirin sağdaki ucuna hareket etmiş olur. Şayet yıldız işaretinin solunda başlangıçta k strok varsa, bu takdirde, k ilmiikten sonra bunların hepsi yıldızın soluna hareket etmiş olacaktır. Son ilmiğin sonunda, solda hiç kalmadığından makina bir strok değil fakat daha ziyade yıldızı tarar ve (son görünüşten sonraki) q_0 halinde bulunur. Daha sonraki devre ($*q_0$ giriş çifti) yıldızı ortadan kaldırır ve makinayı durdurur (son görünüş). Bu halde bant üzerindeki aranan toplamı buluruz.

32. MÜKERRER TOPLAMA VE ÇARPMA

Şek. 20 deki tabloyu öyle değiştirmek istiyoruz ki, bant üzerinde bir n ve m sayı çifti meselâ:

							*			
--	--	--	--	--	--	--	---	--	--	--

verildiğine göre, makina şunlardan ibaret olan sonsuz bir işleme başlasın: yıldızın solunda bulunan m sayısının eklenmesi, sonra m nin $n + m$ toplam sonucuna eklenmesi, sonra m nin $n + 2m$ ye eklenmesi, ilâ... sonsuz şekilde devam etmesi. Bunun için soldaki sayı bir toplama süresinde silinmelidir. Bu, soldaki stroklar yerine onları ortadan kaldıracak yerde başka semboller koymakla yapılabilir. Şek. 22 de içinde α harfinin bu görevi yaptığı bir işleme matrisi görülmektedir. Şek. 20 de birinci satırdaki boş sembol yerine Şek. 22 de birinci satırdaki α gelmiştir ve Şek. 22 de yeni toplanmış α satırındaki ilk üç hane Şek. 20 deki aynen Λ satırındaki gibidir. Ayrıca, makinanın ilk toplamadan sonra durmaması için, Şek.

20 deki dur-hali yerine, işleme devamı yarıyacak ve q_3 diyeceğimiz bazı öteki hallere ait sembol konmalıdır. Bu yeni sembol için tabloda ilâve bir sütun olmalıdır.

Şek. 22 de dur hali yoktur ve bundan ötürü tablonun ifade ettiği işlem sonlu değildir. Okuyucu, buna karşılık

	q_0	q_1	q_2	q_3
	$\alpha R q_2$	L	R	
$\wedge$	R	$R q_0$	q_1	$R q_0$
$*$	q_3	L	R	L
α	R	$R q_0$	q_1	L

Şekil: 22

gelen makinanın, soldaki sayıyla sağdaki sayının sonsuz düzüye toplamasını yapacağına kendini artık kolaylıkla ikna edebilir. Şayet ilk önce yıldızın solunda strok yoksa, yani sağdaki sayı sıfırsa, o takdirde sağda m strok, sonra $2m$, sonra $3m$, ilâ... sonsuz olarak strok husule getirir.

PROBLEM. Çarpım için bir işleme matrisi kurunuz.

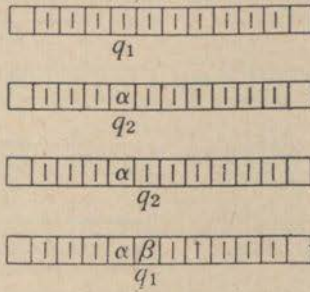
Yol gösterme. Evvelki tabloyu esas olarak alınız ve mükerrer toplama işlemini öyle değiştiriniz ki mükerrer toplama işlemi sonsuz olarak devam etmeyip çarpana eşit sayıda bir ilmikten sonra dursun (her ilmikten sonra, çarpanı gösteren bir cümleden bir strok silinir).

33. EUCLIDE'SEL ALGORİTMA

Şimdi Euclide'sel algoritmaya (iki a ile b sayısının en büyük ortak böleninin bulunması) ait bir Turing makinası gözönüne alacağız. Daha önce bu algoritmayı iki defa, ilki sözlü ihtarlar vasıtasıyla (bölüm 1), ikincisi ise otomatik hesap makinasına ait bir program şeklinde

birinden biri bitinciye kadar, her bir diziden bir birim alıp karşılaştırmak zorunda olan bir insan kalkülâtörün yapabileceği gibi çalışır. Böylece, birinci sayının bir strokunu bir α ya değiştirir, sonra ikinci sayının bir strokunu bir β ya değiştirir; sonra birinci sayıya dönüp onun bir strokunu bir α ya çevirir, ilâ.

İlk dört devrede makina Şek. 23 te gösterilen görünüşleri hasıl eder. Dördüncü devrenin sonunda makina her bir sayıdan bir strok «siler» ve soldaki sayıda, daha sonraki strok için aramağa sola kaymak üzere olur. Bir kaç toplama devresinden sonra bant üzerinde Şek 24 teki



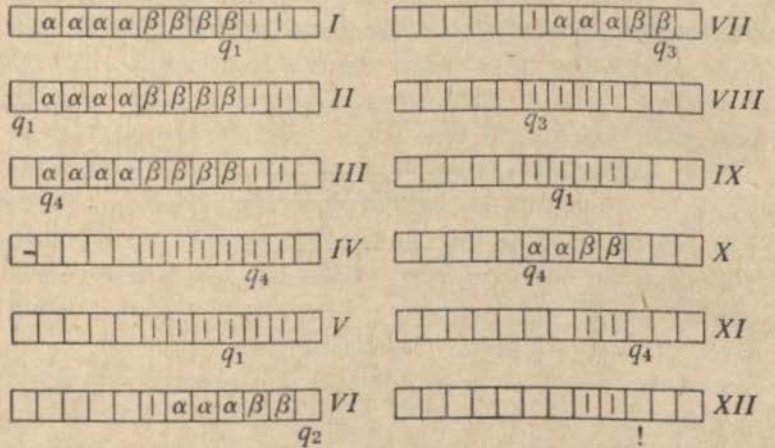
Şekil: 23

I görünüşü ortaya çıkar; birinci sayı son bulmuş fakat ikincisi son bulamamıştır. Solda bir strokun aranması başarısız olup *II* görünüşüne yolaçar; artık mukayese ilmeği sadece q_1 ve q_2 halini kullanmak suretiyle tamamlanmıştır. Bundan sonraki devre *III* görüşünü verir.

Şek. 14 te q_4 sütununda gösterildiği gibi, artık makina, bütün α ları boş sembollere değiştirerek (başka türlü söylersek, onları silerek) ve bütün β ları da stroklar haline değiştirerek sağa kaymağa başlar. Son β yerine bir strok konuluşundan sonra bant, *IV* görünüşünü ve bundan sonraki devreden sonra ise *V* görünüşünü gösterir. Bu şekilde, bir mukayese ilmeği ile bir çıkarma

ilmeğinden sonra, küçük a sayısı silinmiş ve daha büyük b sayısı a ve $b - a$ halinde ayrılmış olur; makina soldaki sayıyı taramakta ve q_1 halinde bulunmaktadır. Bu mukayese devresinin tekrarlanmış olacağı anlamına gelir, ama artık a ve b yerine a ve $b - a$ sayıları için. Zaten bildiğimiz gibi bu Euclide'sel geometrinin sade temelidir.

Şimdi diğer mukayese ilmiğine geçiyoruz. Fakat bu sefer o küçülmeğe başlayan sağdaki sayıyı bitirmekle sona erer. Soldaki sayının üç stroku α lar haline değişmiş



Şekil: 24

olur, sağdaki sayının her iki stroku β lar haline değişmiş olur; makina ise boş bir hücreye gelir (Görünüş VI).

Bundan sonraki devre, diğer çıkarma ilmiğine başlayan VII görünüşünü verir. Makina, bütün β ları silerek ve bütün α ları stroklara çevirerek sola kayar; bu değişmelerin sonuncusundan sonra görünüş VIII i elde ederiz, bunu görünüş IX takip eder. Bu çıkarma ilmeğini sona erdirir ve daha sonraki mukayese ilmeği başlar. İşlem, problem bir birine eşit iki sayıya münce oluncaya kadar devam eder (misâlimizde şimdiden bu durumdayız). Bun-

dan sonra nihai cevaba götürmesi gereken son mukayese ilmiği gelir. Bu görünüş X u verir, son çıkarma görünüş XI i verir, en sonra da XII sonucuna ulaşırız.

34. ALGORİTAMALARIN TERKİBİ

Yeni işleme matrisleri kurulurken, eğer mümkünse evvelce kurulmuş matrisler kullanmak elverişlidir. Yeni gözönüne alınan algoritma, belirli bir anlamda, daha evvel kurulmuş algoritmaların bir terkibi olduğundan bu yapı-

	q_1	q_2	q_3	q_4	p_0	p_1	p_2
0					$1p_2$	!	R
1					$2p_2$	!	R
2					$3p_2$	!	R
3					$4p_2$	!	R
4					$5p_2$	!	R
5					$6p_2$	!	R
6					$7p_2$	!	R
7					$8p_2$	!	R
8					$9p_2$	!	R
9					$0L$	!	R
$\wedge$	Rq_4	Lq_3	Rq_1	p_2	$1p_2$	!	Lp_1
$ $	αq_2	βq_1	Rq_1	Lq_1	L	$\wedge Lp_0$	R
α	L	R	$ L$	$\wedge R$			
β	L	R	$\wedge L$	$ R$			

Şekil: 25

labilir. Bunu bir misâlde açıklayalım. Stroklar zinciriyle gösterilen, verilen bir sayı çiftini, bunların ondalık şekilde yazılmış en büyük ortak böleni haline çevirecek bir algoritma için işleme matrisini kurmak istediğimizi farzedelim. Bu algoritma daha evvelki iki algoritmanın

— önce bölüm 33 tekinin, sonra bölüm 30 dakinin — *terkibinden* (yani ardarda uygulanışından) elde edilebildiğinden bunun işleme matrisi Şek. 25 de görüldüğü gibi Şek. 14 ile Şek. 18 in terkihiyle elde edilebilir. Şe. 14 ün hallerinden ayrı tutmak için önce Şek. 18 in hallerini p_0 , p_1 ve p_2 olarak yazacak ve Şek. 14 te « ! » yerine p_2 alacağız. Bu halde Şek. 25 teki gibi bu değişmiş tabloları birlikte koyacağız. Boş yerler, gözönüne alınan problemde hiç geçmiyecek olan giriş çiftlerini gösterir (herhangi bir çıkış üçlüsü bu yerlerin herbirine konulabilir). Şek. 25, içinde, önce verilen bir stroklar zinciri çiftinin, bunların en büyük ortak bölenlerini gösteren zincir haline çevrildiği, sonra da « ! » da duracak yerde (Şek. 24, görünüş XII), makinanın p_2 haline gittiği bir işlemi ifade etmektedir; işlem devam eder ve sonucu ondalık şekle çevirir.

Bu metodun herhangi sonlu sayıda algoritmanın terkihiine teşmil edilebildiği kolaylıkla görülebilir. Bundan, özellikle sayısal bir algoritma için tabii sayıların stroklar zinciri şeklinde verildiğini farzedebileceğimizi anlarız, çünkü bu hal için bir matris, meselâ $\mathcal{A}$ kurabilirsek, o takdirde, ilk veriler ve sonuç ondalık şekilde olmak üzere hesaplamanın yapılması için, bir matris kurabilir yani şu üç matrisi terkip edebiliriz: ondalık gösterilişi stroklar haline dönüştürmeğe yarıyan matris, $\mathcal{A}$ matrisi, ve strokları geriye ondalık hale dönüştürmeye yarayan matris *.

Algoritmalar terkihiinin diğer metodu aynı algoritmanın önceden belirlenmiş bir şart gerçekleninceye kadar mükerrer uygulanmasından ibarettir. Meselâ, ondalık gösteriliş haline çevirmekte kullanılan algoritma, daha hiç strok bırakılmıyana kadar devam eden n den $n + 1$ e

*) Bu, ikili sistemi kullanan elektronik makinanın işlemesine benzer: ilk ondalık verileri ikili şekle ve ikili sonucu ondalık şekle geçirmeğe yarıyan bazı düzenekleri haiz olmalıdır.

geçiş e ait algoritmanın mükerrer uygulanişından ibarettir. Verilen algoritmanın işleme matrisinden ve hesaplamanın sonuna ait durumdan, mükerrer algoritma için bir matris kurabiliriz. Bununla beraber kullanılan metot, terkip için kullanılan dan daha karışık tır ve bunu teferrüatlı incelemeyeceğiz.

Tahlilini yaptığımız misâllerden, öteki algoritmalar, özellikle sayısal olmyanlar için işleme matrislerinin nasıl kurulacağı ortaya çıkmış olur. Bölüm 14 teki kelime ircası algoritmasının kuruluşunun ana hattını çizeceğiz. Önce aşağıdaki doğru süstitüsyonları icra etmek üzere $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$ matrislerini kuralım:

$$b \rightarrow acc,$$

$$ca \rightarrow accc,$$

$$aa \rightarrow \Lambda,$$

$$cccc \rightarrow \Lambda,$$

(Meselâ $\mathfrak{A}_3$ e karşılık gelen makina $\{a, b, c\}$ alfabesindeki herhangi bir kelimeyi, içinde aa harf çiftinin ilk geçişinin atıldığı diğer kelime haline dönüştürür. aa çifti geçmezse kelimeyi değışmeden bırakır.) Bundan sonra $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4$ ün mükerrer uygulanişlarına ait $\widetilde{\mathfrak{A}}_1, \widetilde{\mathfrak{A}}_2, \widetilde{\mathfrak{A}}_3, \widetilde{\mathfrak{A}}_4$ tablolarını kuralım (Meselâ, $\widetilde{\mathfrak{A}}_3$ e tekabül eden makina, aa nın ilk geçişini başlangıçtaki kelimededen kaldırır, bundan sonra aa nın ilk geçişini yeni kelimededen kaldırır, ilâ, ta ki aa nın hiç bir geçişini haiz olmyan bir kelime husule gelsin). Nihayet, aranan irca algoritması, $\widetilde{\mathfrak{A}}_1, \widetilde{\mathfrak{A}}_2, \widetilde{\mathfrak{A}}_3, \widetilde{\mathfrak{A}}_4$ matrislerinin terki binden ibarettir.

10. ALGORİTMALAR TEORİSİNİN TEMEL HİPOTEZLERİ

35. TEMEL HİPOTEZLER

Son bölümün misâlleri bir Turing makinanın işleyişinin, algoritmanın gerektirdiği hesaplamadan ibaret yavaş bir film gibi olduğu intibasını bırakır. İşleme matrislerinin, meselâ sözlü ihtarlar veya cebirsel formüller gibi çeşitli yollardan verilmiş bilinen öteki algoritmalar için bulunabilip bulunamayacağını bilmek isterdik. İncelememizin bu merhalesinde bunun yapılabildiği açıkça gözükmemektedir. Gerçekten durum bu mudur? Turing makinası ve Turing işleme matrisi ne kadar geneldir? Bütün algoritmaların işleme matrisleri olarak gösterilebildiğini söyleyebilir miyiz?

Modern algoritma teorisi bu sorulara aşağıdaki hipotezle cevap vermektedir.

Algoritma teorisinin temel hipotezi:

Bütün algoritmalar işleme matrisleri halinde verilebilir ve buna karşılık gelen Turing makinalarıyla icra edilebilir.

İki soru ortaya çıkar:

1. Algoritma teorisine dair hipotezin taşıdığı anlam nedir?
2. Hipotez için temel nedir?

Temel hipotez bir yandan «bütün algoritmalar» yani genel algoritma kavramından bahsetmektedir ki bunun kesin bir matematiksel kavram olmadığını bir çok defa belirtmiştik. Öte yandan, kesin matematiksel bir kavram olan Turing işleme matrisinden bahsetmektedir. Bu, genel fakat belirsiz «bütün algoritmalar» kavramının daha özel fakat tamamen kesin Turing işleme matrisiyle aydınlatıldığı anlamına gelmektedir. Böylece, algoritma teorisi, incelemelerinin konusunun, mümkün bütün Turing işleme matrisleri (Turing makinaları) olduğunu ifade etmektedir. O takdirde bazı problem sınıfını çözmeğe yarıyan bir algoritmanın mevcut olup olmayışı gibi soruları ileri sürmenin anlamı olur. Şimdi bunun, gerekli özellikleri haiz olan bir Turing makinasının (işleme matrisinin) mevcut olup olmayışı anlamına geldiğini anlıyoruz.

Böylece, temel hipotez, modern algoritma teorisinin temel tarifinin kabulünü haklı göstermektedir, ki bunun kullanılışı, belirsiz bir algoritma tarifinin, bir Turing makinasının işleme matrisinden ibaret kesin kavramla aynı olduğunun anlaşılmasını mümkün kılar.

Bu önemli hipotezin dayanağı nedir? Bunu bir matematik teoremini ispat ettiğimiz gibi ispat edemeyiz, çünkü bu, kesin olarak tarif edilmemiş bulunan ve bundan ötürü matematiksel incelemenin özel bir konusu olmayan, genel algoritma kavramına dair bir ifadedir.

Hipotezin muteber oluşuna inancımız esas itibariyle denemeye dayanmaktadır. Matematik tarihinde binlerce yıl boyunca gelişmiş bulunan bütün bilinen algoritmalar Turing işleme matrisleri şeklinde yazılabilir. Fakat hipotez, tam geçmişte bulunmuş olan algoritmalar hakkında bir ifade değildir. Bu, geleceği ilgilendiren bir kehanetten de çok farklı bir tabiata sahiptir; herhangi bir ihtarın algoritma şeklinde verildiği her seferinde bu, şekilleri veya elemanter işlemler ne olursa olsun bunları bir Turing

işleme matrisi olarak vermenin mümkün olacağını bildirmektedir. Bu anlamda temel hipotez, fiziksel bir hipotezle, meselâ, geçmişteki geniş denemelerin ağırlığının gelecek olayların kehaneti için iyi bir temel teşkil ettiğine hüküm verdiği enerjinin korunumu prensibiyle mukayese edilebilir.

36. TARİHİ HATIRLATMALAR

Son bölümde bilinen algoritmalarından daha karışık algoritmalar kurmanın iki yolunu zikrettik: terkip ve iterasyon. Uzun bir liste verebilirdik. Bununla beraber, bilinen metotların hepsi, o şekildedir ki ilk algoritmalar işleme matrisi halinde verilir verilmez, bileşik algoritmalar da işleme matrisleri halinde verilebilir. Özel olarak, algoritmalar terkibi için, yeni bir matris kurmağa yarıyan, verilen işleme matrislerinin nasıl kullanıldığını gördük. Bundan başka, bölüm 26 da zikredilmiş bulunan şunu hatırlatıyoruz: sahih bir *algoritma* kavramına ait şiddetli bir zorunluk ortaya çıktığında, bir çok matematikçiler, matematik incelemeler için özel bir konu olacak olan yeteri kadar kesin ve aynı zamanda bütün tasavvur edilmiş algoritmaları göstermekte yeter derecede genel, algoritmalar için standart bir şekil bulmağa koyuldular. Turing işleme matrisiyle birlikte, başka bir çok metotlar teklif edildi. Meselâ, A. A. Markov (bu kitapta bölüm 14 teki irca algoritması vasıtasıyla kısaca anlatılan) *normal algoritmaya* ulaşmıştır, Gödel ve Kleene de *rekürsif (ric'i) algoritmaya (rekürsif fonksiyona)* ulaşmıştır. Bütün bunların denk olduğu sonucuna varılır. Bu bir raslantı olarak düşünülemez; bu temel hipotez için, üstelik bir âmindir.

Algoritma teorisinin kendisinde, temel hipotez kullanılmaz. Yani, teoride teoremlerin ispatlanmasında temel

hipoteze referans gösterilmez. Böylece, hipotezi bilmiyen veya kabul etmiyen bir kimse teoride şekli güçlüklerle karşılaşmazdı. Bununla beraber, bu kimse *algoritma teorisi* dediğimizi sadece Turing işleme matrisleri teorisi olarak, yani *özel tipte algoritma teorisi olarak* gözönüne alırdı.

Yazar şahsen, temel hipotezin muteber oluşuna inanmakta ve bunun sonucu modern algoritma teorisine, basitçe gelişigüzel seçilmiş algoritma sınıfından yani «Turing algoritmaları» ndan ibaret bir teoriden ziyade *genel* bir teori olarak değer biçmektedir.

11. GENEL TURING MAKİNASI

Şimdiye kadar değişik algoritmaların, herbiri kendi işleme matrisiyle çalışan, değişik Turing makinaları tarafından yapıldığı görüş noktasında karar kıldık. Bununla beraber, *herhangi* bir algoritmayı yapmağa güçlü, yani, belirli bir anlamda herhangi bir Turing makinasının işini yapmağa güçlü *genel* bir Turing makinası kurabiliriz.

37. TAKLİT (BENZERİNİ YAPMA) ALGORİTMASI

Bunun nasıl yapıldığını daha iyi anlamak için şöyle düşünelim: makinanın bandı başlangıçta bir A bilgisini ihtiva etsin ve biz bir kimseye, makinanın bu verileri nasıl işlediğini göstermek istiyelim. Eğer bu kimse Turing makinalarının işlemesine alışık, bu takdirde ona A ilk bilgisini ve makinanın işleme matrisini vermek yetecektir. Euclide'sel algoritma için yaptığımız gibi görüşleri kaydetmekle o, kendisi için hesaplamayı tekrar kurabilir. Fakat bu demektir ki şayet ona makinanın işleme matrisi verilirse, o herhangi bir Turing makinasının işini yapabilir. Bir makinanın işleme matrisinden taklidinin tam yolu tabii olarak *taklit algoritması* denilen, Turing makinaları hakkında hiç bir şey bilmiyen biri tarafından icra edilebilen, tam bilgi dizisi halinde bizzat verilebilir. Şayet o, herhangi bir makinanın işleme matrisine ve bandın herhangi ilk görünüşüne malikse, o, makinanın işlemesini tam olarak taklit edebilir ve aynı sonucu elde edebilir.

Mufassal taklit algoritması aşağıdaki ihtarlar listesiyle verilir:

Ihtar 1. Altında bir sembolü bulunan bant üzerindeki (tek) hücreyi tarayınız.

Ihtar 2. İşleme matrisinde taranmış hücre *altında* gözüken sembol tarafından başa getirilen sütunu bulunuz.

Ihtar 3. Taranmış hücre *içinde* gözüken sembol tarafından başa getirilen satırı bulunuz bu satır ve sütunun kesiştiği yerde gözüken semboller üçlüsünü kaydediniz.

Ihtar 4. Taranmış hücredeki harf yerine üçlünün ilk sembolünü yerleştiriniz.

Ihtar 5. Eğer üçlünün ikinci sembolü « ! » ise durunuz.

Ihtar 6. Üçlünün ikinci sembolü *S* ise, taranmış hücre altındaki sembol yerine üçlünün üçüncü sembolünü koyunuz.

Ihtar 7. Üçlünün ikinci sembolü *L* ise, taranmış hücre altındaki sembolü silip üçlünün üçüncü sembolünü solda daha sonraki hücre altına yazınız.

Ihtar 8. Üçlünün ikinci sembolü *R* ise, taranmış hücre altındaki harfi silip üçlünün üçüncü sembolünü sağda daha sonraki hücre altına yazınız.

Ihtar 9. Ihtar 1 e başlayınız.

38. GENEL TURING MAKİNASI

Bizim farazi insan kalkülâtör yerine bir Turing makinası kullanabiliriz; bu diğer herhangi bir Turing makinasının işini yapabilecek *genel* bir makina olacaktır. Başka türlü söylersek, yukarda dokuz sözlü ihtar halinde yazılı taklit algoritması bir Turing işleme matrisi (genel matris) şeklinde verebilir. Kendisi temel hipotezin daha

ziyade bir doğrulanması olan bu keyfiyetin tam ispatını buraya almak oldukça yer tutar. Genel makinanın şeklini ilgilendiren bazı genel açıklamalarla yetineceğiz.

Taklit algoritmasının ilk bilgisi işleme matrisini ve taklit makinasının ilk görünüşü ihtiva etmektedir. Algoritma ilk görünüşü, taklit edilen makina tarafından elde edilebilecek olan sonucu gösteren son görünüş haline dönüştürür. Genel bir makina aynı şeyi yapmalıdır. Aşağıdaki iki güçlüğü yenmek zorundayız.

1. Taklit edilen makinanın işleme matrisini ve ilk görünüşünü genel makinanın bandı üzerine doğrudan doğruya koyamayız, zira burada, bütün Turing makinalarında olduğu gibi bütün bilgiler bant üzerinde tek-boyutlu olarak (yani, bir tek satırda) sıralanmıştır. Fakat verdiğimiz işleme matrisleri bir kaç satırı bulunan «iki boyutlu» matris halindedir; aynı şey içindeki hal sembollerinin bandın altında gözüktüğü görünüşler için de doğrudur.

2. Genel makina (bütün Turing makinaları gibi) sabit bir dış alfabe kullanır. Bu alfabeyle, herhangi sayıda harfleri bulunan alfabeleri sırayla kullanabilen mümkün bütün matris ve görünüşleri göstermeğe güçlü olmamız gerekir.

Bundan ötürü önce, Turing makinalarının gerektirdiği şeylere yani bilginin bir boyutlu oluşu ve alfabenin sonsuz oluşuna uyan bir işleme matrisleri ve görünüşler yazma metodu tasarlamamız gerekir. k satır ve m sütunlu iki boyutlu bir işleme matrisi yerine, mk ardışık beşli sembolü bulunan bir dizi yazarız, bu dizide bir beşlinin ilk sembolü matrisin satır elemanına, ikincisi sütun elemanına karşılık gelmektedir, son üçü ise söz konusu satır ve sütunun keşiştiği yerde gözüken üçlüyü tamamlar.

Meselâ Şek. 12 nin tablosu yerine,

$$\underbrace{\Lambda q_1 \Lambda R q_4 \Lambda q_2 \Lambda L q_3 \dots}_{(Q)}$$

yazarız. İlk tablo aşikâr olarak bu satır tarafından tek bir tarzda tekrar teşkil edilebilir. Benzer şekilde, hal sembolünü taranmış bir hücrenin altından ziyade sağına yazmakla bir görünüşü gösteririz. Böylece Şek. 24 teki IV görünüşü yazılmak istenirse

| | | | q_4 |

şeklinde yazılır.

39. ŞİFRELI SEMBOLLER GURUBU

Verilen bir işleme matrisi ve görünüşü karakterize etmek için iç ve dış alfabede hangi harfleri kullanırsak kulanalım farketmez. Meselâ, Şek. 12 deki tabloda β yerine b harfini koysa idik, makineyi incelememize bunun bir etkisi dokunmazdı. Önemli olanı sadece farklı eşyanın farklı sembollerle gösterilişi ve hal sembolleri ile bant hareketi sembollerini dış alfabeden ayırdetmiye güçlü oluşumuzdur.

Bu şeyleri hesaba katarak, bir Ω satırında herbir harf yerine sıfır ve birlerden ibaret bir seri (*şifreli bir gurup*) koyacağız, öyle ki verilen bir harfin her ortaya çıkışı yerine aynı bir şifreli gurup konulmuş olacak. Sonuç olarak Ω , 0 lar ve 1 lerden ibaret bir Ω' satırına çevrilir. Ω nın Ω' ile teşkil edilmesi (örtülmesi) için şifreleme metodu aşağıdaki şartları gerçeklemelidir.

(1) Ω' , tam bir tarzda kendisinin çeşitli guruplarına ayrılabilir.

(2) Şifreli guruplar arasında kaydırmalara ait (L , R ve S), hal sembolleriyle dış alfabe sembollerini ayırdetmek üzere bir kural (bir algoritma) bulunmalıdır.

Bu iki şart aşağıdaki şifreleme metoduyla uygunluk teşkil eder.

1. Şifreli guruplar için

$$100 \dots 01$$

şeklinde $3 + k + m$ kelime alırız (iki 1 arasında 0 lar zinciri).

Bir Ω' satırı basitçe, arada 1 ler arasında kesilmek suretiyle şifreli guruplarına bölünebilir.

2. Şifre şöyledir:

<i>Sembol</i>	<i>Şifreli gurup</i>
<i>L</i>	101
<i>S</i>	1001
<i>R</i>	10001
dış alfabe	$\left\{ \begin{array}{lll} s_1 & 100001 & 4 \text{ sıfır} \\ s_2 & 10000001 & 6 \text{ sıfır} \\ \cdot & \cdot & \cdot \\ s_k & 100 \dots 001 & 2(k+1) \text{ sıfır} \end{array} \right\}$
hal sembol-leri	$\left\{ \begin{array}{lll} q_1 & 1000001 & 5 \text{ sıfır} \\ q_2 & 100000001 & 7 \text{ sıfır} \\ \cdot & \cdot & \cdot \\ q_m & 100 \dots 001 & 2(m+1)+1 \text{ sıfır} \end{array} \right\}$
	$\left. \begin{array}{l} 2 \text{ den büyük} \\ \text{tek sayıda} \\ \text{sıfır} \end{array} \right\}$ $\left. \begin{array}{l} 3 \text{ ten büyük} \\ \text{çift sayıda} \\ \text{sıfır} \end{array} \right\}$

Bu şifrede $s_1 = \Lambda$ olmak üzere, Ω'

$$\begin{array}{l} 100001 \ 1000001 \ 100001 \ 10001 \ 1000000000001 \ 100001 \\ 100000001 \ 100001 \ 101 \ 10000000001 \dots \end{array}$$

olur.

Böyle işleme matrisi veya gösterişlerden şifrelenmiş 0 lar ve 1 ler satırı, sırayla şifreli işleme matrisleri veya

şifreli görünüşler adını alacaktır. Bu şifreyle, verilen bir işleme matrisi veya görünüş kendi ilk şeklinde kolayca tekrar teşkil edilebilir (0 ve 1 yerine, herhangi başka iki sembol, meselâ *a* ve *b* kullanabilirdik).

40. GENEL TURING MAKİNASI İÇİN ALGORİTMA

Taklit algoritmasına ait 1—9 ihtarlarını, bunların sözlü şekillerinden, şifreli işleme matrisiyle şifreli ilk görünüşü şifreli son görünüş haline çeviren bir algoritma haline dönüştürebiliriz. İşte bir kaç misâl:

Ihtar 1. Şifreli gösterişte, tek sayıda (3 ten büyük) sıfırlı (biricik) şifreli gurubun hemen solunda gözüken şifreli gurubu tarayınız.

Ihtar 2 ve 3. Şifreli matriste ihtar 1 de zikredilmiş guruplar çifti ile denk olan komşu guruplar çiftini bulunuz.

Ihtar 6. Şayet incelenmiş bulunan şifreli guruplar üçlüsünün ikinci kısmı 1001 gurubu ise, o takdirde şifreli görünüşte (3 ten büyük) tek sayıda sıfırı haiz şifreli gurup yerine üçlünün üçüncü gurubunu koyunuz.

Bu algoritmanın ileri bir tahliliyle, şifreli bir gurup üzerine yapılan herhangi bir işlemi (bir sembolü öbürüne çevirerek ve sola veya sağa doğru bir hane kaydırarak) Turing makinalarının standart işleyiş zincirine indirgiyebiliriz. Şunu kaydetmeli ki 0 ve 1 ile birlikte öbür harfler genel makinanın dış alfabesinde gözükecektir, meselâ, şifreli işleme matrisini şifreli görünüşten ayırmağa yarıyan harf ile 0 lar ve 1 lerin incelenmesi sırasında geçici «hatırlatıcılar» rolü oynamaya yarıyan harfler gibi (bölüm 33 te Euclide'sel algoritmaya ait programa bakınız).

Bu tahlilin sonucunda, taklit algoritması bir Turing makinasının işleme matrisi olarak ifade edilebilir. O tak-

dirde bu genel bir makinanın matrisi olur. Şayet bir A makinası bir problemi çözerse, A nın şifreli işleme matrisiyle şifreli ilk görünüşü verildiğine göre genel makina da bunu çözer.

Genel bir makinanın mevcudiyeti verildiğine göre, herhangi bir işleme matrisi (veya şifreli işleme matrisi) iki şekilde gözükebilir.

1. İşleme matrisi buna karşılık olan problemi çözmekte kullanılan özel bir Turing makinasının lojik kısmını anlatır (bu başlangıçta kabul ettiğimiz görüş noktasıdır).
2. Tablo, genel makina tarafından, buna karşılık gelen problemi çözmede kullanılacak bir *program* anlatmaktadır.

Sonuç olarak şunu kaydedelim ki modern elektronik makinalar genel makina gibidir, bunda, verilen bir probleme ait ilk verilere ilâveten, bunu çözmeye yarıyan programı da hatıraya koyarız. Elektronik makinalar da hatıranın bir iç ve bir dış kısma bölünmesiyle karakterize edilir, dış hatıra, meselâ manyetik bir bant veya kasnak olabilir. Fakat bir Turing makinasının dış hatırası sonsuzdur, halbuki, tabii herhangi bir gerçek makinanın dış hatırası sonludur.

Gerçek makinalarla Turing makinası arasındaki bu esaslı farktan, bittabi kaçınılamaz. Aynı zamanda, herhangi gerçek makinanın dış hatırasının, makinanın kuruluşunu değiştirmeden, meselâ ilâve bant üzerine bağlamakla, keyfi olarak artırılabilmesini gerçekleştirmek önemlidir.

12. ALGORİTMA İLE ÇÖZÜLEMİYEN PROBLEMLER

41. CHURCH TEOREMİ

Belirsiz algoritma kavramından kesin Turing makinası kavramına geçip, şimdi verilen bir problem sınıfının bir algoritma ile çözümlüp çözülemeyeceği önemli sorusunu kesinleştirelim. Şimdi bu problemin şu anlama geldiğini anlıyoruz: Verilen problem sınıfını çözmekte kullanılan bir Turing makinası mevcut mudur? («Bir Turing makinası verilen bir problem sınıfını çözer» cümlesinin anlamı için Bölüm 8 e bakınız.)

Algoritma teorisinin «hayır» cevabını verdiği problemler vardır. Bunlardan bir ilki Amerikan matematikçi A. Church tarafından 1936 da keşfedildi (*American Journal of Mathematics*, Vol. 58 de, A. Church, «An Unsolvble Problem of Elementary Number of Theory» ye bakınız). Bu problem sonuç çıkarabilme ile meşgul olur (bölüm 25 e bakınız).

CHURCH TEOREMİ. *Sonuç çıkarabilme problemi algoritma yoluyla çözülemez.*

Bu sadece, buna karşılık olan algoritmayı kurmakta daha önceki bütün teşebbüslerin başarısızlığını açıklamaz, fakat böyle teşebbüslerin haliyle boşa gitmiş olduğunu gösterir.

42. KENDİ KENDİNE HESAPLANABİLME PROBLEMİ

Algoritmalar teorisindeki imkânsızlık ispatları, matematiğin öbür dallarındaki imkânsızlık ispatlarının (meselâ bir açının pergel ve cetvelle üç eşit parçaya bölünüşü veya bir karenin kenarı ile köşegeni için ortak bir ölçek bulunması) aynı olan matematiksel sıhhatle karakterize edilir. *Kendi kendine hesap edilebilme problemi* için böyle bir ispatın ana hatlarını çizeceğiz.

1. Makina bu görünüşe uygulanabilir; yani, bu görünüşten başlayarak, sonlu sayıda adımdan sonra makina bir dur emrine ulaşır.

2. Makina bu görünüşe uygulanamaz; yani, ne kadar uzun süre çalışırsa çalışsın, asla dur emrine erişmez.

Aşağıdaki genel problem ortaya çıkar.

Uygulanabilme problemi. Herhangi bir makinanın işleme matrisi ve ona ait bir görünüş verildiğine göre, makinanın görünüşe uygulanabilip uygulanamadığını tayin ediniz. Bu, algoritmalar kurulmasında tipik bir problem-dir, çünkü bunu çözmek isteyen bir kimse, herhangi bir makina ve herhangi bir görünüş için, makinanın görünüşe uygulanabilip uygulanamadığı sorusuna cevap veren genel bir metot (bir *algoritma* veya *makina*) bulmak zorundadır.

Şimdi bir Turing makinasının bandında, makinanın alfabesi cinsinden yazılı, makinanın kendi şifreli işleme matrisi gözüktüğünü farzedelim. Şayet makina bu görünüşe uygulanabilirse, buna *kendi kendine uygulanabilir* deriz; aksi halde *kendi kendine uygulanamaz* diye adlandırırız.

Kendi kendine hesaplanabilme problemi. Herhangi bir şifreli işleme matrisi verildiğine göre buna karşılık

olan makinanın kendi kendine uygulanabilir olup olmadığını tayin ediniz.

TEOREM. *Kendi kendine hesap edilebilme problemi algoritmik olarak çözülemez.* Besbelli ki bu teorem uygulanabilirlik probleminin algoritmik olarak çözülemezliğini ifade etmektedir.

İspat. Aksine olarak kendi kendine hesaplanabilme problemini çözen bir makinanın mevcut olduğunu farzedelim. A , her kendi kendine hesaplanabilir işleme matrisi (kendi kendine hesap edilebilirliği gösteren) bir σ sembolüne, her kendi kendine hesaplanamaz işleme matrisini kendi kendine hesaplanamazlığı gösteren) farklı bir τ sembolüne dönüştürür. A dan aşağıda zikredilen özellikleri bulunan diğer B makinasını kurabiliriz. Evvelki gibi bu, bütün kendi kendine hesaplanamaz matrisleri τ haline dönüştürür ama bu, kendi kendine hesaplanabilir matrislere uygulanamaz. Böyle bir makina A nın işleme matrisini değiştirmek suretiyle öyle kurulabilir ki σ sembolü gözükünce, duracak yerde, makina silmeden daimi surette σ sembolünü tarar.

Böylece, B , kendi kendine hesaplanamaz bütün matrislere uygulanabilir ve kendi kendine hesaplanabilir bütün matrislere uygulanamaz. Fakat bu bizi bir çelişmeye götürür:

1. B makinasının kendi kendine hesaplanabilir olduğunu farzedelim. O takdirde kendi şifreli B' işleme matrisine uygulanabilir. Fakat B sadece kendi kendine hesaplanamaz makinaların şifreli işleme matrislerine uygulanabildiğinden, bu B nin kendi kendine hesaplanamaz olduğu anlamına gelir.

2. B nin kendi kendine hesaplanamaz olduğunu farzedelim. O takdirde bu B' ne uygulanamaz, ki bu B nin

kendi kendine hesaplanabilir olması gerektiği anlamına gelir. Bu çelişme teoremi ispat eder.

Yukarki teoreme dayanan Turing makinaları teorisinde ortaya çıkan algoritmik çözülemezlik halleri ile çeşitli diğer genel problemler tesis edebilir.

43. ÖRTME PROBLEMLERİ; ÇEVİRİLEBİLME PROBLEMİ

Aşağıda söylediklerimizden ibaret ispat metoduna genişçe yer verildi. Bir $\{a_i\}$ problem sınıfının her a_i problemine, bir $\{b_i\}$ sınıfının bir $f(a_i)$ problemini eşlik ettirelim, öyle ki $f(a_i)$ probleminin bir çözümünden a_i probleminin bir çözümü türetilibilsin. Bu halde $\{b_i\}$ problem sınıfının $\{a_i\}$ problem sınıfını *örttüğünü* söylemek tabiidir. Bu şartlar altında, şayet $\{b_i\}$ problem sınıfını çözmeğe yarıyan bir algoritma mevcutsa, bunun $\{a_i\}$ problem sınıfını çözmeğe yarıyan bir algoritma haline çevrildiği de aşikârdır. Bununla beraber şayet, algoritmik olarak çözülemez $\{a_i\}$ problem sınıfı tesis edilmişse, bundan $\{b_i\}$ sınıfının aynı şekilde algoritmik olarak çözülemediği çıkar. Böyle bir durum çoğu defa, algoritmik çözülemezlik kurulmasında kullanılır. Özel olarak, verilen bir $\mathfrak{B}$ problemi için bir u problemi bulunur şöyle ki $\mathfrak{B}$, u yu örter fakat u algoritmik olarak çözülemez. Bu metodu kullanarak bir çözülemezlik ispatına dair bir misâl veriniz.

Çevrilebilirlik problemi. Verilen M Turing makinası ve buna ait iki K_1, K_2 görünüşü için, şayet K_1, M ye ait ilk görünüş olarak alındığına göre, K_2 işlem sırasında hiç, ya son görünüş veya bir ara görünüş olarak ortaya çıkar mı?

M Turing makinasından aşağıdaki özeliği bulunan yeni bir M^* Turing makinası kurabiliriz.

K , M ye ait herhangi bir görünüş olsun.

1. Şayet M , K ye uygulanamazsa, o takdirde M^* , K ye uygulanamaz.

2. Şayet M , K ye uygulanabilirse, o takdirde M^* de K ye uygulanabilir, ve son görünüş, taranmış hücrede τ dan ibarettir ve τ nun M ye ait dış alfabede herhangi bir sembol olduğu bütün öteki hücrelerde boş kalır.

Herhangi bir M makinası için, M ye ait uygulanabilirlik problemi P_u ve M^* ye ait çevrilebilirlik problemi $f(P_u)$ olsun. Şayet $f(P_u)$ çözülebilirse, bu halde, her K görünüşü için, M^* nin bunu K^* ye dönüştürüp dönüştürmediğini tayin eden bir makina mevcuttur; fakat M ancak ve ancak K ya uygulanabilirse M^* , K yi K^* ye dönüştürür (M^* nin kuruluşuna bakınız), bu, $f(P_u)$ yi çözen makinanın P_u yi de çözeceği anlamına gelir. Böylece, çevrilebilirlik problemi, $\{f(P_u)\}$ yi ihtiva ettiğinden uygulanabilirlik problemi $\{P_u\}$ yi örter. Bundan ve uygulanabilirlik probleminin algoritmik olarak çözülemezliğinden, çevrilebilirlik probleminin de algoritmik olarak çözülemezliği çıkar.

Ayrıca, şayet K_2 son görünüşlere münhasır kalırsa çevrilebilirlik problemi çözülemez kalır.

44. TARİHİ AÇIKLAMALAR

Algoritmik çözülemezlik ilkin matematiksel lojikteki problemler (sonuç çıkarabilme problemi) ve algoritmalar teorisindeki problemler (meselâ, kendi kendine hesap edilebilme problemi) için tesis edilmişti. Daha sonra ise, matematiğin birçok başka dallarındaki benzer fakat daha az genel problemlere yöneltildi. Bunların en önemlileri, kelime probleminin değişik çeşitlerine getirilen cebirsel problemler dizisidir.

Asosyatif hesaplar için kelime problemi ilkin 1914 gibi erken bir tarihte Norveç matematikçisi Thue tarafından formülleştirildi; o, özel tipten bazı asosyatif hesaplardaki algoritmaları verdi. O vakitten beri herhangi bir çift için kullanılabilen genel bir algoritma kurmakta bir çok çabalar harcanmıştır. 1946 ve 1947 de Sovyet matematikçisi A. A. Markov ve Amerikan matematikçisi Emil Post birbirlerinden bağımsız olarak, kelime probleminin algoritmik olarak çözülmediği asosyatif hesap misalleri kurmuşlardır. Böylece *evleviyetle* keyfi hesaplarda kelime denkliliğini tayin edecek bir algoritma yoktur. O vakit A. A. Markov ve öğrencilerinin bu sonucu üzerine geniş asosyatif hesaplar sınıfı için algoritmalar bulunmasının imkânsızlığı kurulmuş oldu.

Bundan sonraki bölümde herhangi keyfi asosyatif hesapta kelime denkliliğinin tayinine ait bir algoritma mevcut olmadığına ilişkin teoremin ispatlarından birini vereceğiz.

1955 te P. S. Novikov gurup teorisindeki özdeşlik probleminin algoritmik çözülemesizliğini ispat etmekle matematik dünyasında büyük bir canlılık yaratmıştır. Bu problem, özel bir hesaplar sınıfına, yani, içinde alfabenin her a harfi için yine alfabenin (muhtemelen a ya eşit) bir α harfinin

$$a\alpha - \Lambda$$

uygun sübstitüsyonlar listesinde gözükmek üzere mevcut olduğu sınıfa ait kelime denkliliği probleminin yerini tutar. Bölüm 14 te olduğu gibi, bir harfi elemanter bir dönüşüm, bir kelimeyi de, kelimedeki harflere karşılık gelen elemanter dönüşümlerin *çarpımı* olarak yorumlarsak, bu halin anlamı aydınlanmış olur. Bu halde, boş harf özdeşlik dönüşümüne (bütün elemanları bulundukları yerde

bırakan dönüşme) karşılık gelir. O takdirde $aa - \Delta$ uygun süstitüsyonunun mevcudiyeti, her a elemanter dönüşümü için bir α elemanter dönüşümünün mevcut olduğunu öyle ki α dan sonra a nın uygulanışının özdeşlik dönüşümü olduğunu ifade etmektedir. Daha ileriye gitmeden, *dönüşüm gurupları* adı verilen böyle bir dönüşüm cümlesinin incelenişinin büyük pratik ve teorik faydası olduğunu, gurup kavramının, modern matematiğin en önemli bir kavramı bulunduğunu kaydedeceğiz.

Markov ve Post'un en önemli sonuçlarının, guruplar teorisindeki özdeşlik problemi hakkında açıklanacak hiç bir sonuca henüz elvermediğini yukarda söylenenlerden anlamak yerinde olur. Bunun sebebi, Post ve Markov tarafından kurulan hesapların yukarda verilen gurup aksiyomunu gerçekleştirmemiş oluşudur. Bundan ötürü, bu problem için bir algoritma kurulması için hâlâ ümit beslenmekteydi, ve bu husustaki çabalar, Novikov'un sonuçlarını yayınlamasına kadar devam etti. Novikov, gurup aksiyomunu gerçekliyen bir hesap misâli verdi ki bunun için hiç bir algoritma mevcut değildi. Bu yüzden, bütün guruplar için genel bir aksiyom, insana imkânsız gibi gelmektedir.

Markov ve Post tarafından kurulan misâller pek karışık idi ve uygun süstitüsyonlardan yüzlercesini ihtiva etmekteydi. Daha basit misâl bulma meselesi bir çok uluslar tarafından yeni yeni çözülmüştür. Meselâ, G. S. Tsentin, sadece yedi uygun süstitüsyon ihtiva eden ve buna ait kelime probleminin algoritma ile çözülemediği, bölüm 12 deki misâlin hesabını kurdu.

Algoritma ile çözülemez problemlerin keşfi, bir algoritma kurmağa çalışan bir matematikçinin, algoritmanın asla mevcut olmadığı imkânını da düşünmesi gerekli bulunduğu bir durumun ortaya çıkmasına sebep olur. Şu halde, istenen algoritma kurmağa harcanan çabalara

paralel olarak, bu algoritmanın mevcut olmayışını ispat etmekte de çaba harcanmalıdır. Böylece böyle bir problemin çözülmesi ya

- 1) algoritmanın bulunuşu
- veya
- 2) bunun mevcut olmayışının ispatı olabilir.

Bölüm 3 te Diophant denklemleri üzerine Hilbert problemini anlattık. Yarım asır boyunca bu problem için bir algoritma kurmağa boş yere çaba harcanmıştır. Daha yeni yeni çabalar zıt yönde, yani böyle bir algoritmanın mevcut olmadığını ispata harcanmaktadır. Bununla beraber henüz kesin sonuç mevcut değildir ve bu zamana kadar elde edilmiş bulunan kısmi sonuçlar, Hilbert probleminin algoritmik olarak çözülemeyeceği zannını vermektedir.

13. GENEL KELİME PROBLEMİNE AİT BİR ALGORİTMANIN MEVCUT OLMAYIŞI

Bu bölümde, herhangi bir keyfî hesapta kelime denklîği tayinine ait bir algoritmanın mevcut olmadığı ispatlanmıştır. Bu ispat iki kısımda yapılmıştır.

Birinci kısımda $P \rightarrow Q$ direkt sübstitüsyonlu π asosyatif hesapları gözönüne alınmıştır (bölüm 11 e bakınız). Simetrisiz denilebilen böyle hesaplar için, çevrilebilirlik problemini gözönüne alalım. Oklar, bir önceki kelimenin doğru sübstitüsyonlardan birinin uygulanışıyla bir sonrakine dönüşebildiğini göstermek üzere, şayet bir

$$R \longrightarrow R_1 \longrightarrow R_2 \longrightarrow \dots \longrightarrow R_i \longrightarrow S$$

geçilebilme zinciri mevcutsa, bir R kelimesi bir S kelimesine çevrilebilir denilir. Herhangi bir simetrisiz hesapta kelime çevrilebilirliğini tayin için hiç bir algoritma mevcut olmadığı tesis edilmiştir. π hesabından, her bir $P \rightarrow Q$ direkt sübstitüsyonu yerine $P - Q$ endirekt sübstitüsyonu yerleştirmekle elde edilen bir π' hesabı türetebiliriz.

Aşikâr ki, bir R kelimesi bir π hesabında bir S kelimesine çevrilebilir ise o takdirde bu kelimeler π' de birbirine denktir. Denklik tesisinde herhangi bir direkt $P \rightarrow Q$ uygun sübstitüsyonun $Q \rightarrow P$ tersi de uygulanabildiğinden bu iddianın karşıtı genel olarak doğru değildir. Demek ki simetrisiz hesaplar için elde edilen sonuç, kendiliğinden denklik problemine teşmil edilemez. İkinci

kısımda bu zorluk aşılmış ve denklik problemine ait çözümleme teoremi ispat edilmiş olur.

45. ÇEVİRİLEBİLİRLİĞİN TAYİNİNE AİT BİR ALGORİTMANIN MEVCUT OLMAYIŞI

TEOREM 1. *Her R ile S kelime çifti için, R nin S ye çevrilip çevrilemediğini tayin eden bir algoritma yoktur.*

Teorem 1 in ispatı simetrisiz hesaplar için çevrilebilirlik problemini, Turing makinaları için çevrilebilirlik problemine indirerek yapılır. Mademki ikincisi algoritma ile çözülemez o halde birincisi de çözülemezdir.

Bir Turing makinasının herhangi bir görünüşünü gözönüne alalım. Bu görünüşün aşağıdaki hücrelerini aktif diye adlandıracağız:

- a) taranmış hücre;
- b) boş sembolen başka bir sembol ihtiva eden hücreler;
- c) a veya b tipinden hücreler arasında bulunan hücreler.

Bütün aktif hücreler cümlesi bant üzerinde bunun aktif kısmı denilen sürekli bir küme teşkil eder. Şek. 26 da aktif kısımları işaretlenmiş bulunan bazı görünüşler gösterilmiştir. Şek. 26 a daki taranmış hücre, bandın aktif kısmının bir iç hücresidir; yani aktif kısmın ne en sağdaki ne de en soldaki hücresidir. Şek. 26 b; 26 c ve 26 d deki sırayla *sol*, *sağ* ve *birlik* diye adlandıracağımız gibi görünürlere zıt olan bir görünüşe bir iç görünüş denir.

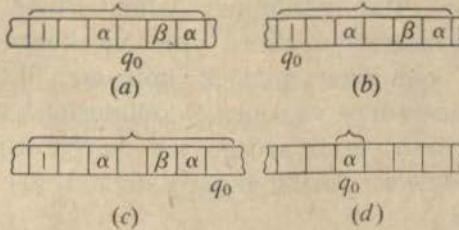
Bir makinanın dış alfabesi

$$s_1, s_2, \dots, s_m$$

ve iç alfabesi de

$$q_1, q_2, \dots, q_k$$

olsun. Bandın aktif kısmının sınırlarını göstermek üzere bir de (yukardaki alfabelerin hiç birine katılmıyan) h harfi ithâl edelim. O takdirde her görünüş, R , bölüm



Şekil: 26

38 de kurulmuş bulunan kelime olmak üzere, bir hRh kelimesi şeklinde gösterilebilir. Bu kelimelere G -kelimeleri (görünüş kelimeleri) diyeceğiz. Meselâ, Şek. 26 daki görünüşlerden

$$\begin{aligned} h \mid \Lambda \alpha \Lambda q_0 \beta \alpha h; & \quad h \mid q_0 \quad \alpha \Lambda \beta \alpha h; \\ h \mid \Lambda \alpha \Lambda \beta \alpha \Lambda q_0 h; & \quad h \alpha q_0 h \end{aligned}$$

kelimelerini elde ederiz.

M makinasına aşağıdaki şekilde tarif edilen bir π_M hesabı eşlik ettirelim.

1. π_M alfabesi, h harfiyle M nin alfabesinin bütün harflerinden ibarettir. Şunu kaydedelim ki her G -kelimesi π_M hesabındaki bir kelime olduğu halde, π_M deki her kelime bir G -kelimesi değildir. Meselâ, q_1 harfi $hs_1q_1s_1h$ kelimesinde iki defa gözükmemekte olduğu halde bunun bir G -kelimesinde gözükmesi imkânsız olur.

2. π_M deki (doğru) sübstitüsyonlar öyle bir tarzda kurulmuştur ki, makinadaki ihtarların yerine getirilmesine karşılık olan G -kelimelerindeki dönüşümlerin uygunluğunu sağlar. Bandı harekete getirmiyen

$$sq \longrightarrow s'Sq' \quad (1)$$

şeklinde bir ihtar gözönüne alalım. Böyle bir ihtarın yerine getirilmesinden sonra yine öncekinin aynı hücrelerin aktif olduğunu görmek kolaydır. İhtarın yerine getirilmesinden önce ve sonra G -kelimesini karşılaştırarak sq harf çiftinin yerine sadece $s'q'$ harf çiftinin geldiğini görürüz. Böylece, Turing makinasındaki (1) ihtarına π_M hesabındaki

$$sq \longrightarrow s'q'$$

doğru sübstitüsyonunu eşlik ettirmiş oluyoruz.

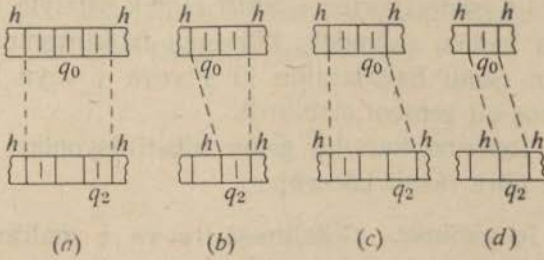
Şayet bir ihtar bandı harekete getirirse, o takdirde bandın aktif kısmı, görünüş tiplerine (iç, sol v.s.) ve kayma yönüne (sol, sağ) bağlı olarak değişebilir. Böylece, ihtara karşılık gelen tek bir sübstitüsyon değil (1) ihtarı halindeki gibi) daha ziyade bir çok sübstitüsyondan ibaret bir cümle elde ederiz.

MİSÂL. Toplamaya ait işleme matrisinden çıkan

$$| q_0 \longrightarrow \Lambda Rq_2$$

iharı Şek. 27 de gösterilen dönüşüm görünüşlerini ortaya çıkartır.

Şek. 27 a da, bandın aynı kısmı aktif kalır. Şek. 27 b ve 27 c de, aktif kısım sırayla önce daralır sonra genişler. Şek. 27 d de, aktif kısım kayar fakat aynı büyüklükte kalır.



Şekil: 27

Buna karşılık gelen G -kelimeleri için Tablo 3 te gösterilen dönüşümleri elde ederiz.

TABLO 3

İlk G -kelimesi	Dönüşmüş G -kelimesi
$h \mid q_0 \mid h$	$h \mid \Lambda \mid q_2 h$
$h \mid q_0 \mid h$	$h \mid q_2 h$
$h \mid q_0 h$	$h \mid \Lambda \Lambda q_2 h$
$h \mid q_0 h$	$h \mid \Lambda q_2 h$

Sağdaki sütunda bulunan kelimelerin, aynı doğru süstitüsüyonu uygulamakla, soldaki buna karşılık gelen kelimelerden türetilemediğini görmek kolaydır.

Şimdi

$$sq \longrightarrow s'Rq' \quad (2)$$

şeklindeki ihtarlar karşılık gelen doğru süstitüsyonlar sisteminin nasıl kurulduğunu göstereceğiz ($sq \rightarrow s'Lq'$ şeklindeki ihtarlar hali benzer şekilde incelenir).

Aşağıdaki notasyonu ithal ediyoruz: şayet taranmış hücrenin hemen sağındaki hücre aktif ise, bunun muhte-

vasını σ ile gösteriyoruz, τ , aktif olmak şartıyla taranmış hücrenin hemen sağındaki hücrenin muhtevasını göstermektedir. Şunu hatırlatalım ki σ veya τ veya her ikisi birden boş bir sembol olabilir.

(2) ihtarına karşılık gelen süstitüsyonları görünüş tiplerine göre tasnif edeceğiz:

1. *İç görünüş.* G -kelimesi (σ ve τ makinanın dış alfabesindeki herhangi harfler olmak üzere) bir $\sigma s q \tau$ kelimesinin bir geçişini ihtiva eder. Böyle her kelimeye

$$\sigma s q \tau \longrightarrow \sigma s' \tau q'$$

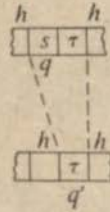
süstitüsyonunu eşlik ettiriyoruz.

2. *Sol görünüş.* G -kelimesi bir $h s q \tau$ kelimesinin bir geçişini ihtiva etmektedir. Bunu bir

$$s' \neq \Lambda \text{ ise } h s q \tau \longrightarrow h s' \tau q',$$

$$s' = \Lambda \text{ ise } h s q \tau \longrightarrow h \tau q'$$

süstitüsyonuna eşlik ettiriyoruz. Son süstitüsyon s yi ihtiva eden süstitüsyonun aktif olmıyan hale geldiğini aksettirmektedir (bakınız Şek. 28).

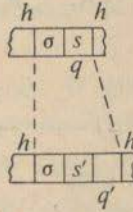


Şekil: 28

3. *Sağ görünüş.* G -görünüşü bir $\sigma s q h$ kelimesinin bir geçişini ihtiva etmektedir, buna

$$\sigma s q h \longrightarrow \sigma s' \Lambda q' h$$

geçişini eşlik ettiriyoruz ki bandın aktif kısmının genişlediğini ifade eder (bakınız Şek. 29).



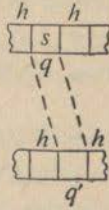
Şekil: 29

4. *Birlik görünüşü.* G -kelimesi $hsqh$ şeklindedir; ona

$$s' \neq \Lambda \text{ ise } hsqh \longrightarrow hs' \Lambda q'h$$

$$s' = \Lambda \text{ ise } hsqh \longrightarrow h \Lambda q'h$$

süstitüsyonunu eşlik ettirelim. Son süstitüsyon bandın aktif kısmının kaydığını belli etmektedir (bakınız Şek. 30).



Şekil: 30

Bu (2) şeklindeki makina ihtarlarından ortaya çıkan π_M deki bütün doğru süstitüsyonların sayılmasını sonuçlandırır.

MİSÂL. Toplamaya ait Σ Turing makinası için (bölüm 31 e bakınız), buna karşılık olan π_Σ hesabını kurunuz.

π_Σ nın alfabesi

$$|, \Lambda, *, h$$

dir. $\Lambda q_2 \rightarrow |Sq_1$ ihtarı

$$\Lambda q_2 \longrightarrow |q_1$$

süstitüsyonuna karşılık gelir.

$|q_0 \rightarrow \Lambda Rq_2$ ihtarı şu süstitüsyonlara karşılık gelmektedir:

$$\begin{aligned} \text{İç} \quad & \left\{ \begin{array}{l} |q_0| \rightarrow |\Lambda|q_2, \\ |q_0\Lambda \rightarrow |\Lambda\Lambda q_2, \\ |q_0^* \rightarrow |\Lambda^*q_2, \\ \Lambda|q_0| \rightarrow \Lambda\Lambda|q_2, \\ \Lambda|q_0\Lambda \rightarrow \Lambda\Lambda\Lambda q_2, \\ \Lambda|q_0^* \rightarrow \Lambda\Lambda^*q_2, \\ *|q_0| \rightarrow *\Lambda|q_2, \\ *|q_0\Lambda \rightarrow *\Lambda\Lambda q_2, \\ *|q_0^* \rightarrow *\Lambda^*q_2; \end{array} \right. \\ \text{Sol} \quad & \left\{ \begin{array}{l} h|q_0| \rightarrow h|q_2, \\ h|q_0\Lambda \rightarrow h\Lambda q_2, \\ h|q_0^* \rightarrow h^*q_2; \end{array} \right. \\ \text{Sağ} \quad & \left\{ \begin{array}{l} |q_0h \rightarrow |\Lambda\Lambda q_2h, \\ \Lambda q_0h \rightarrow \Lambda\Lambda\Lambda q_2h, \\ *|q_0h \rightarrow *\Lambda\Lambda q_2h; \end{array} \right. \\ \text{Birlik} \quad & \{ h|q_0h \rightarrow h\Lambda q_2h. \end{aligned}$$

Başka ihtarlara karşılık gelen süstitüsyonlar tıpkı böyle sayılabilir.

Şimdi verilen bir M makinasından kurulan π_M hesabının aşağıdaki özelliklerini kaydedelim:

Iddia 1. M makinasına ait her G -kelimesi π_M deki bir kelimedir.

Iddia 2. Şayet $\tilde{A}$ bir A görünüşüne karşılık gelen G -kelimesi ise, o takdirde π_M de en fazla bir süstitüsyon buna uygulanabilir. Bu süstitüsyon $\tilde{A}$ yı, makinanın A ya dönüştürdüğü B görünüşüne karşılık gelen $\tilde{B}$ G -kelimesine dönüştürür.

Iddia 3. Şayet A , M makinasına ait bir son görünüşü ise, o takdirde, hiç bir süstitüsyon $\tilde{A}$ ya uygulanamaz.

Bu iddialardan, doğrudan doğruya bir A görünüşünün M makinasında bir B görünüşüne çevrilebilir olup olmadığı sorusunun, π_M hesabında $\tilde{A}$ G -kelimesinin $\tilde{B}$ G -kelimesine çevrilebilir çevrilemediği sorusuna denk olduğu çıkar. Başka türlü deyişle, simetrisiz hesaplara ait çevrilebilirlik problemi Turing makinalarına ait çevrilebilirlik problemini örter. Bu, Teorem 1 i ispatlar.

B sadece M makinasına ait son görünüşleri gösterdiğine göre, aşağıdaki teoremi elde ederiz.

TEOREM 2. *Simetrisiz kayfî bir hesap ve ikincisi irca edilmiş olan keyfî bir $\tilde{A}$ ve $\tilde{B}$ çifti için, $\tilde{A}$ nın $\tilde{B}$ ye çevrilip çevrilemediğini tayine yarıyan bir algoritma mevcut değildir.*

46. KELİME DENKLİĞİ PROBLEMİNİN ÇÖZÜLEMEZLİĞİ

R ile S , π_M hesabında iki G -kelimesi olsun. Şayet R , doğru süstitüsyonlarla S ye çevrilebilirse, o takdirde R ,

π_M de *evleviyette* S ye denktir. Eğer direkt (doğru) süstitüsyonlar yerine buna karşılık olan endirekt süstitüsyonlar kinsa acaba daha ileri bir denklik ortaya çıkar mı? Bu sorunun cevabı aşağıdaki yardımcı teoremden verilmiştir.

YARDIMCI TEOREM. *Eğer S bir son G -kelimesi ve R , π_M de (endirekt süstitüsyonları kabul ederek) S ye denk ise, o takdirde R , sadece π_M nin direkt süstitüsyonlarını kullanmakla S ye çevrilebilir.*

Asosyatif hesaplarda kelime denklği probleminin algoritmik olarak çözülemeliği doğrudan doğruya bu yardımcı teoremden çıkar. Zira, şayet kelimelerin denk olup olmadığını tayine yarıyan bir algoritma mevcut olsa idi, simetrisiz hesaplardaki çevrilebilirlik problemini çözerdi. Fakat Teorem 2 ye göre, böyle bir algoritma yoktur.

Yardımcı teoremin ispatı. Eğer $R \sim S$ ise, o takdirde R den S ye giden bir geçilebilir zinciri mevcuttur:

$$R = R_1 - R_2 - R_3 - \dots - R_{k-1} - R_k = S. \quad (3)$$

R_i ile R_{i+1} , bu zincirde iki komşu kelime olsun. Eğer R_i den R_{i+1} e geçiş $P \rightarrow Q$ şeklindeki direkt bir süstitüsyonun uygulanışıyla yapılmışsa, $R_i \rightarrow R_{i+1}$ yazarız; eğer verilen direkt süstitüsyonlardan birinin tersini uygulamakla yapılmışsa (sayfa 123 e bakınız), $R_i \leftarrow R_{i+1}$ yazarız. Şimdi (3) zincirindeki üç ardışık kelimenin aşağıdaki imkânlarını gözönüne alalım:

$$R_{j-1} \leftarrow R_j \rightarrow R_{j+1}, \quad (4)$$

$$R_{j-1} \rightarrow R_j \leftarrow R_{j+1}. \quad (5)$$

2 idiasından ötürü, R_i kelimesine sadece bir süstitüs-yon uygulanabildiğinden, R_{i-1} ile R_{i+1} kelimeleri (4) halinde çakışmalıdır. Demek ki, böyle bir kelime üçlüsü bir geçilebilme zincirinden, basitçe elemanlarından ikisini (meselâ R_{i-1} ile R_i yi) silip çıkarmakla yokedilebilir. (5) halinde, bununla beraber, R_{i-1} ve R_{i+1} birbirinden farklı olabilir. Turing makinası bakımından bu, makina-nın görünüşünün önceki görünüşü tek şekilde tayin etmediği anlamına gelmektedir.

Şimdi (3) zincirine dönelim. Mademki R_k bir son görünüştür, yalnız $R_{k-1} \rightarrow R_k$ elde ederiz (iddia 3 e bakınız). Eğer bütün oklar sağa doğru yönelmişse, yardımcı teorem ispat edilmiş olur. Sola doğru yönelen okların bulunduğunu farzedelim, meselâ bu, sonuncusu yani R_i den R_{i-1} e olanı bulunsun. O takdirde bize R den S ye giden daha kısa bir geçilebilme zinciri veren, kendi-sinden iki kelime silinebilen bir

$$R_{j-1} \leftarrow R_j \rightarrow R_{j+1}$$

üçlüsü mevcuttur. Bu yoketme işlemine devam ederek, sonunda herhangi daha kısası yapılamıyan bir zincire ulaşırız; o takdirde bu sadece sağa yönelen oklar ihtiva etmeğe zorunludur. Böylece, uygun direkt süstitüsyon-larla R , S ye çevrilebilir.

SONUÇLAYICI AÇIKLAMALAR

Netice itibariyle bir kaç genel gözlem yapalım.

1. İlk önce, bazı sınıf problemlerin algoritmik olarak çözülemezliğine ilişkin teoremler bir umutsuzluğa sebep olmamalıdır. Böylece bir teorem sadece, *bütün* problem *sınıfını* elealmada güçlü bir algoritma mevcut olmadığını ispat eder. Bu demek değildir ki sınıfın ihtiva ettiği özel problemler arasında algoritmik olarak çözülebilen hiç biri mevcut değildir. Meselâ, bölüm 42 de ispatlanmış teorem, özel bir makina için prensip itibariyle, kendisinin hesap edilebilir olup olmadığını tesis etmenin imkânsız olduğu anlamına alınmamalıdır. Bu sadece, problem sınıfının çok geniş olduğu; sınıfın *bütün* problemlerini çözen tek bir algoritmanın mevcut olmadığı anlamına gelmektedir. Böyle durumlarda, verilen tipten problemlerin daha genel alt sınıfını çözmeye yarıyan daha daha genel algoritmalar kurmağa çalışılmalıdır.

2. İkinci olarak, algoritmik çözülemezlik teoremleri, matematiğin algoritmalar kurulmasına irca edilmediğini gösterir. Dar matematik alanlarına nispet edilse bile (meselâ, sonlu geneleştirilmiş guruplar teorisi), herhangi bir neviden otomaton (sonlu sayıda halleri ve sonlu hatırası bulunan Turing makinası) ile çözülemiyen geniş problemler vardır. Bundan, tamamiyle ilmin yaratıcı çalışması sayesinde, makinalar hakkındaki ifadelerin saçmalığı anlaşılır.

3. Aynı zamanda şunu bilmek gerekir ki algoritmaların uygulama alanı pek geniştir ve matematikteki

hesaplama işlemiyle kısıtlanmamıştır. Alışlageldiği gibi zor ve karışık gibi düşünülen bir çok işlemler için aslında oldukça basit olan algoritmalar kurma mümkündür; bu işlemlerin yapılmasıyla ilgili pratik güçlükler, algoritmaların çoğu defa icra edilecek muazzam sayıda işlem (her bir işlemin kendisi basit olsa bile) gerektirmesi keyfiyetinden ortaya çıkmaktadır. Bu düşünce, özel olarak gali-biyetin, çoğu defa en uygun hareketi tayin etmede geniş sayıda imkânlar gözetmekteki kabiliyete bağlı olduğu oyunlara (özellikle satranca) uygulanır.

Yüksek hız hesaplama makinalarının imaliyle birlikte pratik algoritmaların sayısı önemli derecede genişlemiştir.

4. Nihayet, her gerçek makinanın ancak yaklaşık bir Turing makinası modeli olarak gözönüne alınabildiğini bir kere daha zikredelim. Yani, her gerçek makina sınırlı bir dış hatırayı haizdir, halbuki Turing makinası sonsuz bir bandı haizdir. Tabii, fizik bakımından, sonsuz bir hatıra kurmak imkânsızdır, fakat, daha önce ortaya getirilen ölçekle mukayese edildiğinde makina boyutlarındaki önemli bir artma, sadece makbul bir şey değil aynı zamanda pek muhtemeldir. Böylece, dış hatıranın boyutlarını artırma ve hesaplama hızını artırma, hesap otomatlarının gelişmesinde çok daha büyük ilerlemeler getirecektir.

BİBLİYOGRAFYA

1. Turing, A. M. «On Computable Numbers, with an Application to the Entscheidungsproblem», *Proceedings of the London Mathematical Society*, Seri 2, Cilt 42 (1936-37), sayfa 230-65.
2. Kleene, S. C., *Introduction to Metamathematics*. Princeton, N. J.: Van Nostrand, 1952.
3. Davis, Martin, *Computability and Unsolvability*. New York: McGraw-Hill, 1958.
4. Rozsa, Péter, *Rekursive Funktionen*. Berlin: Akademie Verlag, 1957.
5. *Advances in Computers*, Cilt 1 (Alt, Franz L., ed.) New York: Academic Press, 1960.
6. Beth, E. W., *Foundations of Mathematics*. Groningen: North-Holland Publishing Co., 1959.
7. McCracken, D. D., *Digital Computer Programming*. New York: John Wiley, 1957.

Yayınlanmış olan eserler:

- 1 — Eşitsizlikler (160 Krş.)
- 2 — Çokrenk Problemleri (180 Krş.)
- 3 — Sayılar Teorisinden Bazı Problemler (500 Krş.)
- 4 — Tesadüfi Hareketler (250 Krş.)
- 5 — Geometrik İspatlarda Hatalar (180 Krş.)
- 6 — Mekaniğin Matematiğe Bazı Tatbikleri (180 Krş.)
- 7 — Yalnız Pergel Kullanarak Yapılan
Geometrik Çizimler (250 Krş.)
- 8 — Denklemlerin Tam Sayılarla Çözülmesi
(Diophant Denklemleri) (220 Krş.)
- 9 — Gruplar Teorisine Giriş (375 Krş.)
- 10 — Eşitsizliklere Giriş (425 Krş.)
- 11 — Matematiksel İndüksiyon Metodu (220 Krş.)
- 12 — Sayılar Teorisine Giriş (550 Krş.)
- 13 — Geometrik Eşitsizlikler (450 Krş.)
- 14 — Yalnız Cetvel Kullanarak Yapılan
Geometrik Çizimler (250 Krş.)
- 15 — İhtimaller Hesabına Giriş (550 Krş.)
- 16 — Rasyonel ve İrrasyonel Sayılar (600 Krş.)
- 18 — İndirgemeli Diziler (200 Krş.)
- 19 — Geometri (Cilt I) (500 Krş.)
- 20 — Geometri (Cilt II) (750 Krş.)
- 21 — Geometri (Cilt III) (400 Krş.)
- 22 — Algoritmalar ve Otomatik Hesap Makinaları (400 Krş.)

Yayınlanmak üzere olan eserler:

- 17 — Gayri Euklid Geometriler

Matematik Derneğinin adresi:

Fen Fakültesi, Matematik Enstitüsü

Vezneciler - İstanbul

Fiati 400 Krş.