SAKARYA ÜNİVERSİTESİ

# Ayrık İşlemsel Yapılar

## Hafta 1

**Doç.Dr. Nilüfer YURTAY**

# Kombinasyonel Problemler ve Teknikler

## 1.1 Giriş

Ayrık matematik veya ayrık işlemsel yapılar, olası durumları sonlu olan problemin çözümü ile uğraşan matematik alanıdır. Bu problemler varlık belirleme, sayma ve optimizasyon olmak üzere üç ana kategoriye ayrılabilir. Bazı durumlarda çözümün var olup almadığı açık değildir. Bu bir varlık belirleme problemidir (existence). Bazı durumlarda ise çözümün olduğu bilinir ancak, bunların kaç tane olduğunu bilmek isteriz. Bu ise bir sayma (counting) problemidir. En iyi olan çözümün istendiği durum ise optimizasyon problemi olarak düşünülebilir. Her bir tip için çeşitli örnekler verilebilir.

Dört evli çift , her pazar akşamı iki sahada karışık ve çift olarak tenis maçı yapıyorlar. İki saat oyun süresi içinde her yarım saatte bir eşlerini ve rakiplerini değiştiriyorlar. Her bir adamın her bir kadınla en az bir kere birlikte ve karşısında oynadığı ve her bir diğer adama karşı en az bir kez oynadığı bir fikstür var mı ? Bu problem bir varlık belirleme problemidir.

Altı kişilik bir yatırım kulübü her yıl başkan ve sayman pozisyonlarını dönerli olarak değiştirmek istiyorlar. Aynı insanların , aynı pozisyonlara gelmesi için kaç yıl geçmelidir ? Buradaki problem ise sayma problemi olarak düşünülebilir.

Bir iş yerindeki üç çalışan Ali, Ayşe ve Ahmet sırasıyla 10 milyon, 12 milyon, 15 milyon saat ücreti alıyorlar. Patronun bu insanlara vereceği 3 ayrı iş var. Tabloda her bir insanın bu işleri ne kadar sürede yapacağı gösterilmiştir. Patron toplam olarak en az ödeme yapacak şekilde her birine hangi işi vermelidir ? Açıkça görülmektedir ki bu bir optimizasyon problemidir.

| | Ali | Ayşe | Ahmet |
|---|---|---|---|
| İş 1 | 7,5 | 6 | 6,5 |
| İş 2 | 8 | 8,5 | 7 |
| İş 3 | 5 | 6,5 | 5,5 |

Çoğunlukla , kombinasyonel bir problemde , problemi çözmek için adım-adım işlem blokları içeren procedurden bir algoritma geliştiririz. Bir çok algoritma bilgisayar programı olarak gerçeklemeye çok uygun olduğundan, ayrık matematiğin önemi de artmıştır. Buna rağmen , güçlü bir bilgisayarda olsa , kombinasyonel problemin çözümünde olası tüm durumlar programlamak çoğunlukla olanaksızdır.

Çok daha özenli yöntemler geliştirmemizi gerektirmektedir. Bu bölümde daha karmaşık örneklerle kombinasyonel problemleri ele alacağız ve onların nasıl çözülebileceklerinin bazı analizlerini yapacağız.

## 1.2 Proje Tamamlama İçin Zamanlama

Büyük bir mağaza bayram indirimi için 8 sayfalık bir ilan göndermeyi planlıyor. Bu ilanın bayramdan en az 10 gün önce postalanması gereklidir. Ancak bir çok işin yapılması ve öncelikle de bazı kararların alınması gerekiyor. Kısım müdürlerinin stoktaki hangi malların indirime gireceğini seçmesi, satın alıcıların da yine hangi malların indirim için alınacağını seçmesi gerekiyor. Yönetim kurulunun bunların içinden hangilerinin ilana gireceği ve satış fiyatlarını belirlemesi gerekiyor. Reklam bölümü seçilen malların isimlerini hazırlayacak ve bir yazarda onların reklam yazılarını hazırlayacaktır. Daha sonra resimler ve yazılar bir araya getirilerek ilan hazırlanacaktır. Postalama sistemi ise, satışa girecek mallara bağlı olarak çeşitli kaynaklardan derlenecek, etiketler basılacak, ilan basıldıktan sonra etiketler yapıştırılarak, posta koduna göre sıralanarak postaneye gönderilecektir. Tüm bu işlerin zamana ihtiyacı vardır. Bunun için ise sadece 30 gün olduğunu kabul edelim. Her bir işin ne kadar süre alacağını tablo olarak verirsek ;

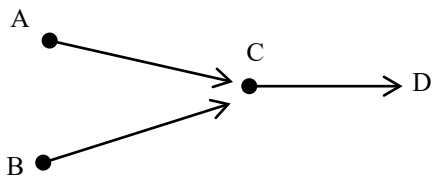
| İş | Gün |
|---|---|
| Mal Seçimi (Kısım Mud.) | 3 |
| Mal Seçimi (Satın Alma) | 2 |
| İlan için mal seçimi ve fiyat belirleme | 2 |
| İlan Hazırlığı (Resim) | 4 |
| İlan Hazırlığı (Yazı) | 3 |
| **İlan Tasarımı** | **2** |
| Posta Listesi Hazırlama | 3 |
| Etiket Basımı | 1 |
| İlan Basımı | 5 |
| Etiket Yapıştırma | 2 |
| İlan Postalama | 10 |

Görüldüğü gibi ilk bakışta bu işin yetiştirilmesi için gerekli toplam zaman 37 gündür. Ancak bazı işleri aynı anda gerçeklemek olası olduğu gibi bazı işler de önceki işlerin yapılmasına bağlıdır. Örneğin kısım müdürleri ve satın almacılar birbirinden bağımsız olarak işlerini yapabilirler. Ama posta listesi hangi malların ilana konacağına seçiminden sonra hazırlanabilecektir. Buna göre hangi işin hangisine bağlı olduğunu belirlemek için işlere A,B,C.....K gibi etiketler verelim ve bir başka tablo hazırlayalım.

| İş | Önceki İşler |
|---|---|
| A Mal Seçimi (Kısım Mud.) | Yok |
| B Mal Seçimi (Satın Alma) | Yok |
| C İlana Mal Seçimi Ve Fiyat Saptama | A,B |
| D Resim | C |
| E Yazı | C |
| F İlan Tasarımı | D,E |
| G Posta Listesi | C |
| H Etiket Basımı | G |
| I İlan Basımı | F |
| J Etiket Yapıştır | H,I |
| K İlan Postalama | J |

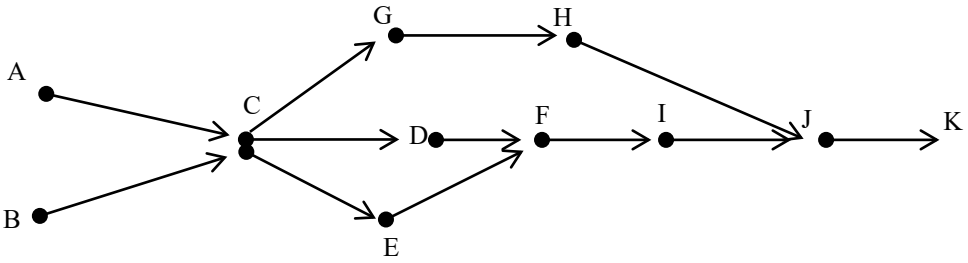
Bütün elemanların mümkün olan en kısa zamanda işlerine başlamış olduklarını varsayalım. Yine de işin yetişip yetişmeyeceğine henüz karar veremiyoruz. Bu işin 30 gün veya daha kısa sürede tamamlanabilme olasılığı var mı? Görüldüğü gibi burada problem bir varlık problemidir. Bu sorunun yanıtı için bazı analizleri yapmamız gerekmektedir.

Bazı durumlarda bilgiyi grafik olarak göstermek çok daha anlaşılabilir kılacaktır. Her bir görevi bir nokta ile gösterelim ve iki görev arasında eğer biri diğerinin yapılması için gerekiyorsa bir okla iki noktayı birleştirelim. Örneğin A ve B işleri C den önce yapılmalı ve D işi için de C önce yapılmalıdır. Buna göre graf şekil 1.1.'deki gibi olacaktır.



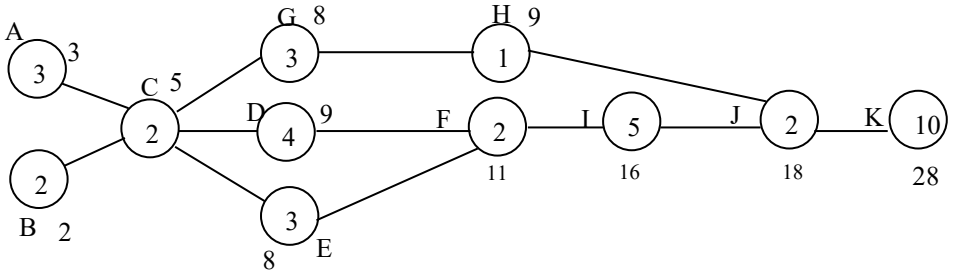

**Şekil 1.1**

Benzer düşünceyle yukarıda anlatılan problemin graf gösterimi şekil 1.2'deki gibidir.




**Şekil 1.2**

Burada tüm okların sağdan sola doğru olduğunu bildiğimize göre okları çizmeyebiliriz. Ayrıca her düğümü küçük bir daire ile gösterip içine o iş için gerekli süreyi de yazabiliriz. Bu durumda şekil 1.3'ü elde ederiz.




**Şekil 1.3**

Diyagramın her düğümünün dışına da o işin tamamlanması için gereken maksimum süreyi yazalım. Örneğin C düğümünün dışına 5 yazmalıyız. Çünkü A işi 3 gün B işi 2 gün olduğundan, A ve B bitmeden C tamamlanamayacaktır. Benzer biçimde F için 11 gün gerekmektedir. D işi 9 , E işi 8 gün gerektirdiğinden fazla olanı alıyoruz. Sonuçta K işi için toplam 28 gün bulduğumuza göre iş zamanında yetişecek demektir.

### 1.2.1 Kritik Yol Analizi

Açıkladığımız bu yöntem PERT (Program Evalution and Review Technique) olarak adlandırılır. Bu yöntem birkaç büyük projenin programlanması zamanlanması için çok kullanılan bir yöntemdir.

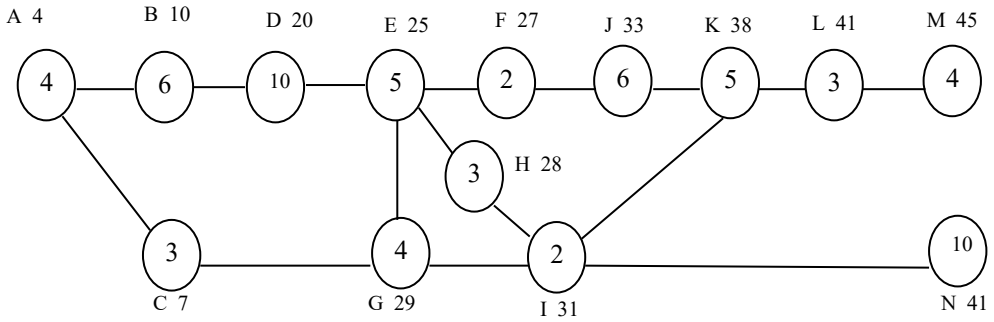
Diyagramdan daha farklı bilgileri de çıkarmak mümkündür. Şimdi işin tamamlandığı K düğümünden geriye doğru incelemeye başlarsak ; K 28 günde tamamlanıyor , buna etken J 'nin 18 günde

bitmesidir. J ' ye baktığımızda H ve I işlerine bağlıdır. H, 9 ve J, 16 günde bittiğine göre I çok daha önemli olmaktadır. Bu düşünce ile geriye başlangıç düğümlerine doğru analize devam edersek diyagramda sonuçta A düğümüne varırız. A-C-D-F-I-J-K yolu " Kritik Yol " olarak adlandırılır. Kritik yol toplam proje zamanını belirlediği için önemlidir. Eğer proje süresini kısaltmak istiyorsak kritik yol üzerindeki işleri kısaltmaya çalışmalıyız. Örneğin G işini 3 günden 2 güne indirsek G kritik yolunda olmadığı için toplam süreye etki etmeyecektir. Ama I süresini 1 gün kısaltsak toplam süre 27 güne olacaktır. Bu arada kritik yol üzerindeki işlerin süresini kısaltmak kritik yolun değişmesine neden olabilir.

**Örnek 1.1**

Aşağıdaki tabloda yapılacak işler , süreleri ve diğerlerine bağlılıkları verilmiştir. Kritik yol ve işin bitiş süresi nedir ?

| İş | Süre | Önceki Adımlar |
|---|---|---|
| A | 4 | Yok |
| B | 6 | A |
| C | 3 | A |
| D | 10 | B |
| E | 5 | D |
| F | 2 | E |
| G | 4 | C,E |
| H | 3 | E |
| I | 2 | G,H |
| J | 6 | F |
| K | 5 | I,J |
| L | 3 | K |
| M | 4 | L |
| N | 10 | I |

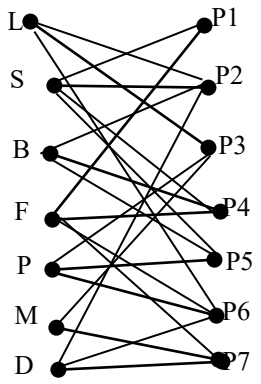



Kritik Yol : A-B-D-E-F-J-K-L-M Süre 45 gün

## 1.3 Eşleme Problemi

Türk Hava Yollarının pazartesi sabahları Ankara'dan 7 ayrı ülkeye uçuşu vardır. THY 7 ayrı pilotunu uçmak istedikleri şehirlere göre tarife yapmak istemektedir. Pilotların isteklerine göre aşağıdaki liste hazırlanmıştır.

Libya : P2 , P3 , P6
Suriye : P1 , P2 , P4 , P5
Brezilya: P2 , P4 , P5
Fransa : P1 , P4 , P6 , P7
Peru : P3 , P5 , P6
Macaristan : P3 , P7
Danimarka : P2 , P6 , P7




Bu listeye göre görevlendirme mümkünse tüm pilotların isteklerine göre yapılacak , değilse mümkün olduğunca çok eşleme yapılacaktır. Bu problem bir optimizasyon problemi olarak düşünülebilir. Uçuşlarla pilotlar , pilotların isteğine göre mümkün olduğunca fazla çakışma olacak şekilde eşlenecektir.

Eşleme işlemine doğrudan başladığımızı düşünelim. Her uçuşa bir pilotu eşlemek için olası tüm yolları listeleyebiliriz ve olası her yol için kaç tane görevlendirmenin pilotların isteğine uyduğunu sayabiliriz. Örnek olarak bir alanı eşlemeyi ele alalım.

| | | İstek |
|---|---|---|
| L | P1 | Hayır |
| S | P2 | Evet |
| B | P3 | Hayır |
| F | P4 | Evet |
| P | P5 | Evet |
| M | P6 | Hayır |
| D | P7 | Evet |

Bu eşlemede 4 pilot istediği yere uçabilmektedir. Başka bir eşleme çok daha iyi bir sonuç verebilecektir. Eğer uçuş listesini aynı sırada sabit tutarsak, alanı eşlemede yapılan iş , pilotların sıralamasını değiştirmektir. Örneğin P2 , P2 , P3 , P4 , P5 , P6 , P7 bir başka eşleme olacaktır. Bu problemi çözmek için ortaya çıkan sorular şunlar :

1) Olası sıralama sayısı kaç tanedir ?
2) Tüm olası sıralamayı herhangi birini unutmadan nasıl üretebiliriz?

İkinci soru biraz daha özel bir soru olup daha ileride cevabı tartışacağız. Ancak ilk sorunun cevabını bulabiliriz. Dikkat edersek bu bir sayma problemidir. Burada sayma işlemi sırasında kullanacağınız bazı temel prensip ve teoremleri ele alacağız.

## 1.4 Sayma Teknikleri

Çoğu kombinasyonel problem sayma gerektirmektedir. Problemde ele alınması gereken obje sayısı genellikle çok fazla olduğundan, geçerli obje kümesini listelemeden bunların sayısını bulmamız istenir.

### Pascal Üçgeni ve Binom Teoremi

Bazı problemlerde, verilen bir kümenin belli bir sayıda eleman içeren alt kümelerinin sayısı istenir.

$$C(n,r)=\frac{n!}{r!(n-r)!}$$

n, kümedeki eleman sayısı, r alt kümedeki eleman sayısıdır. Örneğin sesli harflerin {a,e,i,o,u} olarak tanımlanan bir kümesinin iki elemanlı alt kümelerinin sayısı $C(5,2)\frac{5!}{2!3!}=\frac{5.4.3!}{2.1.3!}=10$ olarak bulunabilir.

**Teorem 1.1**

**r ve n, $1\leq r\leq n$ olmak üzere tamsayılar ise C(n,r)=C(n-1,r-1)+C(n-1,r) dir.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| n=0 | | | | C(0,0) | | | | |
| n=1 | | | | C(1,0) | | C(1,1) | | |
| n=2 | | | C(2,0) | | C(2,1) | | C(2,2) | |
| n=3 | | C(3,0) | | C(3,1) | | C(3,2) | | C(3,3) |
| n=4 | C(4,0) | | C(4,1) | | C(4,2) | | C(4,3) | C(4,4) |

üçgeni Pascal üçgeni olarak bilinir. Bu üçgeni yakından incelersek C(n,0)=C(n,n)=1 olduğundan her satırın ilk ve son elemanları 1'dir. Ayrıca Teorem 1.1'e göre, her satırdaki ilk ve son olmayan elemanların dışındaki elemanlar, bir üst satırda kendine en yakın elemanların toplamına eşittir. Örneğin C(4,2)=C(3,1)+C(3,2) olacaktır. Sonuç olarak Pascal Üçgeni aşağıdaki gibi gösterilebilir.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | | | | |
| | | | 1 | | 1 | | | |
| | | 1 | | 2 | | 1 | | |
| | 1 | | 3 | | 3 | | 1 | |
| 1 | | 4 | | 6 | | 4 | | 1 |

**Teorem 1.2**

**r ve n, 1≤r≤n olmak üzere tamsayılar ise C(n,r)=C(n,n-r) dir.**

C(n,r) sayıları Binom sabitleri olarak adlandırılırlar. $(x+y)^n$ in açılımında bu sabitler $x^{n-r}y^r$ nin katsayılarıdır. Buna göre $(x+y)^n$ in katsayıları Pascal üçgeninde n.satırın katsayılarıdır.

Örneğin $(x+y)^3=(x+y)(x+y)^2=x^3+3x^2y+3xy^2+y^3$ de olduğu gibi.

**Pigeonhole Prensibi**

Eğer p eleman q adet yere yerleştirilecek ve $p>kq$ $(k>0)$ ise, q adet yerden bazılarında en az k+1 eleman olacaktır.

**Örnek 1.2**

15 evli çift içinden kaç kişi seçilmelidir ki, seçilenler içinde en az bir evli çift olsun ? Sorunun cevabı 16 kişidir.

**Çarpma Prensibi**

Bir prosedürün ardışık k adımdan oluştuğunu varsayalım. Birinci adım $n_1$ farklı yol, ikinci adım $n_2$ farklı yolla yapılabilsin. Tüm prosedür, $n_1.n_2.....n_i.....n_k$ adet farklı yolla yapılabilir.

## Örnek 1.3

Bir Japon arabası 6 farklı renkte, 3 farklı motorla, otomatik veya manuel vitesle araba üretilebiliyor. Kaç farklı model araba olabilir ? k=3 n1= 6 n2=3 ve n3=2 olduğuna göre 6.3.2=36 farklı model olacaktır. Şimdi uçuş problemine geri dönelim. Birinci uçuşu ele alalım. 7 pilot içinden birini seçebiliriz. Yani 7 olasılık vardır. Bunların içinden bir tanesini seçip , ikinci uçuşa geçtiğimizde 6 farklı seçeneğiniz kalacaktır. Bu şekilde devam edersek olası eşlemelerin sayısı 7.6.5.4.3.2.1 olacaktır. Demek ki n adet uçuş ve n adet pilot varsa eşleme sayısı n!=n.(n-1)(n-2)...3.2.1 adettir. Bu işleme permütasyon adını vermekteyiz. n adet objenin farklı biçimde sıralanması işlemine , permütasyon diyoruz. n objenin içinden r objenin tekrarlanmadan seçilebilme sayısı n objenin r adet permütasyonu olup ,

P(n,r) = n !/(n-r) ! olacaktır.

Örneğin 7 uçuştan 2 si iptal edilmesi durumunda olası görevlendirme sayısı ;

P(7,5) = 7.6.5.4.3 olacaktır.

Problemin çözümünün gerçekleştirilebilmesini tekrar ele alırsak , 7 uçuş için 7 pilot 7! = 5040 farklı biçimde görevlendirilebilir. Eğer bilgisayar kullanıyorsak , önerilen çözüm yöntemi , yani permütasyonları oluşturup değerlendirme işlemi için bir algoritma geliştirilip programlanabilir. Burada 7 uçuş ve 7 pilot örneğine bakarsak gerçeğe göre çok küçük sayılar olup , örneğin Chicago havaalanına günde 1100 uçuş yapılmaktadır.Sadece 20 uçuş ve 20 pilott alsak 20!= $2.4.10^{18}$ olup bilgisayar gerekmektedir. Varsayalım ki bilgisayar saniyede $10^6$ atama yapabilsin ve kaç pilotun isteğine uygun olduğunu kontrol edebilsin. Tüm olası çözümler için bilgisayarın çalışma süresi ;

$2.4.10^{18}/10^6=2.4.10^{12}$s= $4.\ 10^{10}$ dak= $6.7.10^8$ saat=$2.8.10^7$gün =$7.6.10^4$ yıl

Görülüyor ki bilgisayar kullansak bile çok daha akıllı bir hesaplama yöntemi bulmalıyız. İleriki bölümlerde eşleme problemini çok daha etkin yöntemle çözeceğiz.

**Toplama Prensibi**

Eleman sayıları $n_1,n_2,...,n_k$ olan k adet küme olsun. Eğer bu kümelerin elemanları ayık ise, yani hiçbir kümenin başka küme ile ortak elemanı yoksa, bu kümelerin birleşimleri ile oluşan kümenin eleman sayısı $n_1+n_2+...+n_k$ dır.

## Örnek 1.4

A={1,2,3}, B={4,5,6,7} olsun. A ve B kümelerinin elemanları ortak değil, yani $A \cap B = \varnothing$ dir. O halde $|A \cup B| = |A| + |B| = 7$ dir.

## Örnek 1.5

1 ile 100 arasında çift veya 5 ile biten kaç sayı olduğunu araştıralım. 1-100 arasında 50 çift sayı vardır. 5 ile biten tüm sayılar tek sayı olup bunlar 15,25,35,45,55,65,75,85,95 olup 10 tanedir. O halde istenen yanıt 50+10=60 olacaktır.

## 1.5 Knapsack Problemi (Sırt Çantası )

Bir uzay mekiği bir uzay istasyonuna gönderilecektir. Bilim adamlarınca tasarlanan denemeler için 1400 Kg'lık bir yükleme sınırı vardır. Araştırmacılar deneyimlerine göre başvurmuşlar ve her deney için de yanlarına almaları gereken cihazların ağırlıklarını belirlemişlerdir. Daha sonra başvurular değerlendirilmiş ve her bir deneyin önemine göre 1 den 100 e kadar puan verilmiştir. Buna göre her bir deneyin gerektirdiği cihaz ağırlıkları ve önem puanları şöyledir;

| Deney No | Ağırlık (Kg) | Puan |
|---|---|---|
| 1 | 72 | 6 |
| 2 | 528 | 10 |
| 3 | 376 | 7 |
| 4 | 406 | 9 |
| 5 | 208 | 9 |
| 6 | 14 | 7 |
| 7 | 184 | 3 |
| 8 | 130 | 9 |
| 9 | 50 | 4 |
| 10 | 340 | 7 |
| 11 | 160 | 8 |
| 12 | 44 | 5 |

Deneylerin seçiminde, toplam puanını mümkün olduğunca fazla olması, ancak toplam ağırlığın 1400 kg geçmemesi istenmektedir. Bunun nasıl gerçekleşeceği çok açık değildir. Örneğin hemen liste başından başlayarak seçsek 1,2,3 ve 4 toplam 1382 olacak. 5. deney 208 kg olduğundan alınamayacak, 6. deney 14 kg olduğu için alınacaktır. Bu şekilde bir seçim yaparsak toplam puan ;

6+10+7+9+7 = 39 olmaktadır.

Şimdi soru bundan daha iyi bir çözüm olup olamadığıdır. Bir yol listeyi önem puanlarına göre sıralayıp, seçimi buna göre yapmak olabilir. Bir ihtimal bu öncekine göre çok daha iyi bir seçim olabilir. Sıralamada aynı puanda olan deneyden daha hafif olanı önce alırsak , sonuç liste ;

| Deney No | Puan | Ağırlık | Alınsın mı ? | Toplam Ağırlık |
|---|---|---|---|---|
| 2 | 10 | 528 | Evet | 528 |
| 8 | 9 | 130 | Evet | 658 |
| 5 | 9 | 208 | Evet | 866 |
| 4 | 9 | 406 | Evet | 1272 |
| 11 | 8 | 160 | Hayır | 1272 |
| 6 | 7 | 14 | Evet | 1286 |
| 10 | 7 | 340 | Hayır | 1286 |
| 3 | 7 | 376 | Hayır | 1286 |
| 1 | 6 | 72 | Evet | 1358 |
| 12 | 5 | 44 | Hayır | 1358 |
| 9 | 4 | 50 | Hayır | 1358 |
| 7 | 3 | 184 | Hayır | 1358 |

olacak ve toplam puan 10+9+9+9+7+6 =50 olacaktır. Bu bir öncekinden çok daha iyi bir çözüm oldu. Bir başka fikir deneyleri ağırlıklarına göre artan sırada listeleyip seçimi buna göre yapma olabilir. Bu

durumda 6,12,9,1,8,11,7,5 ve 10. deneyler seçilecek ve toplam puan da 58 olacaktır. Fakat bunların hiçbiri optimum seçim değildir.

Problemle biraz daha uğraşırsak deneyleri ekleyip çıkartarak 58 puandan daha iyi bir sonuç elde edebiliriz. Fakat yine de bulduğumuz sonucun en iyisi olduğunu söyleyemeyiz. Görüldüğü gibi bu da bir başka optimizasyon problemidir. Bir önceki eşleme problemine benzer olarak , tüm olasılıkları deneme metoduna dönelim. Deneyleriniz 1 den 12 ye kadar kodlandığına göre problemi çok daha kompakt biçimde ele alabiliriz. Burada bize gerekli olan küme teorisi olacaktır. Küme teorisi hakkında ön bilgimiz olduğunu varsayarsak tüm deneyleri içeren kümeyi U={1,2,3,4,5,6,7,8,9,10,11,12} ile gösterebiliriz. Bu kümeden örneğin 1,2,3,4 ve 6 deneylerini seçmemiz bir T alt kümesi oluşturmak demektir. T={1,2,3,4,6} T kümesi , problemin çözümünde ele aldığımız ilk seçimdir. U'nun bazı alt kümeleri , toplam ağırlık 1400 kg'yi geçtiği için kabul edilmeyecektir. Örneğin {2,3,4,10} alt kümelerinin toplam ağırlığı 1650 kg'dir. O halde basit olarak U kümesinin tüm alt kümeleri için toplam ağırlığı hesaplayabiliriz. 1400 kg'yi geçmeyenlerin toplam puanlarını hesaplayıp , sonuçta hangi alt kümenin ( ya da alt kümelerinin) max puanı olduğunu buluruz. Buna göre iki soru akla gelecektir ;

1. Kaç tane at küme vardır ?
2. Herhangi birini unutmadan bu alt kümeleri nasıl listeleyebiliriz ?

n elemanlı bir kümenin $2^n$ adet alt kümesi vardır. U kümesinin 12 elemanı olduğuna göre $2^{12}$ = 4096 dır. Görüldüğü gibi alt küme sayısı ancak bilgisayarla hesaplamaya uygundur.

Eşleme probleminin aksine bu problemin çözümü için bilinen etkin bir yol yoktur. Etkin yoldan ne kastedildiği ise karmaşıklık teorisinin tartıştığımızda daha açık anlayacağız.

## 1.6 Algoritma Karmaşıklığı

Önceki bölümlerde ele alacağımız problemleri çözmek için çeşitli algoritmalar geliştirdik. Ayrıca bunları bilgisayarda çözdürmek için gereken süreyi de ele aldık. Bazı çözümler için bulduğumuz sürelerin hiçte kabul edilebilir olmadığını gördük. Bu durumda algoritmaların kurulmasını ve etkinliğini biraz daha detaylı ele almamız gerekmektedir.

Sırt çantası örneğinde gördük ki 12 deney için yaklaşık 4096 alt küme sınanmaktadır. Eğer deney sayısı 20 olsaydı , bu rakam 1000000 olacaktı. 250 kez daha fazla. Burada problem, n deney sayısı ile ölçülebilir. Her tür problem için çözümün dayandığı bilgi miktarına ölçü olabilecek bir n sayısı

bulunabilir. Problemin çözümü için geliştirilecek algoritmalarda n aynı büyüklüğü göstermek üzere , buna bağlı olarak karşılaştırılabilir.

Ayrıca , problemin çözümünde yapılacak olan hesaplama yükünü de ölçmek isteriz.Muhakkak ki bu da n sayısına bağlıdır. Bir algoritmada n büyüdükçe bu yükün hızla artmaması istenilebilir. Bir algoritmik çözümün büyüklüğünü ölçmek için bir birime ihtiyacımız vardır. Uzay mekiği probleminde n deney için $2^n$ alt küme sınanmakta idi. Alt küme sınanmasından kasıt , küme elemanlarının ağırlıklarının toplanması, 1400 kg yi aşıp aşmadığını sınanması , aşmıyorsa toplam puanın hesaplanması ve bir önceki en iyi çözümle karşılaştırılmasıdır. Bir alt küme için gereken işlem yükü aynı zamanda o kümedeki eleman sayısına da bağlıdır.

İlk olarak algoritmanın değerlendirilmesinde kullanılan alışılagelmiş yöntemi ele alalım. Bu yöntemde toplama , çıkartma , çarpma , bölme ve karşılaştırma gibi elementer işlemlerin toplam sayısı hesaplanır. Örneğin k adet
$a_1$, $a_2$, $a_3$..... $a_k$ sayısını toplamak istersek k-1 adet toplama yaparız ; $a_1+ a_2$, $(a_1+ a_2)+ a_3$ .....$(a_1+$ .....$+a_{k-1})+a_k$

Elemanter işlemlerin toplam sayısını algoritmanın karmaşıklığı (complexity of algorithm) adını veriyoruz. Bu yolla karmaşıklık ölçmenin iki dezavantajı vardır.

1. Bu yöntem esas olarak algoritmanın gerçeklenmesi için geçecek toplam süreyi hesaplamaya yöneliktir. Burada her elamanter işlemin aynı sürede yapıldığı varsayılmaktadır. Ancak, bilgisayarlar bellekleri ile sınırlıdırlar. Algoritmanın bellek gereksinimi bilgisayarda mevcut bellekten fazla olabilir. Ya da daha yavaş ek bellek kullanımı gerekebilir ve işlem yavaşlayabilir. Saklama olayı gözönüne alınması gereken bir değerdir.

2. Elementer işlemler aynı sürede uygulanmazlar .Örneğin bölme işlemi, toplamadan uzundur.

Ayrıca elementer işlemin süresi, üzerinde işlem yapılan elemanların büyüklüğüne de bağlıdır. Büyük sayılar çok daha uzun süre olurlar.Toplu bir değişkene değer atamanın da bir süre alması gibi işlemler de bizim hesaplamamıza katılmamaktadır. Bu eleştirilere rağmen, algoritmanın karmaşıklığının ölçülmesinde önerilen yöntem kullanılabilir. Özel bir bilgisayarın iç işlemleri ile uğraşmamak ve basitlik için bu yöntemi kullanacağız.

## Örnek 1.6

Bazı algoritmaları ve karmaşıklıklarını ele alacağız. İlk olarak m bir reel sayı ve n bir tam sayı olmak üzere $m^n$ in hesaplanmasına bakalım. $X^2=X*X$ , $X^3=(X^2)*X$...,$X^n=X^{n-1}*X$ olduğuna göre toplam n-1 çarpma gerekecektir.Bu işlem için bir algoritma aşağıdaki gibidir.

***Algoritma***

Verilen bir m reel sayısı ve pozitif n sayısı için $P=X^n$ 'i hesaplayan algoritma

*Adım1 (KOŞULLAMA)*

*P=X ve k=1*

*Adım 2 (SONRAKİ KUVVET)*

*While k< n*

*(a) P ← P.X*

*(b) k←k+1*

*EndWhile*

*Adım 3: ( $P=X^n$ olur)*

*Print P.*

Dikkat edilirse 2.adımda n-1 çarpma n-1 toplama ve adet karşılaştırma ( k=n olduğunda 2.adımdan çıkılıyor ) yapılıyor.Buna göre $X^n$ in hesaplanması için toplam (n-1)+(n-1)+n=3n-2 adet elemanter işlem yapılmaktadır.

Genellikle , bu işlem sayısında tam bir değer değil de bir mertebe söylememiz bizim için yeterlidir. Bu örnek için "işlem sayısı 3n civarındadır". Hatta "işlem sayısı n in bir sabitle çarpımı kadardır" bile diyebiliriz. Çoğunlukla n in büyümesi durumunda işlem sayısının hızla büyüyüp büyümediği bizi daha çok ilgilendirir.

## Örnek 1.7

Şimdi de n.dereceden polinomun hesaplanmasını ele alalım

$$P(X)=a_n x^n +a_{n-1}x^{n-1} +.....+ a_1 x +a_a$$

$a_n \neq 0$ ve $a_i$ (i=0,1,.....n) sabit

**Algoritma**

*P(X)=$a_n x^n$ +.....+$a_0$, n pozitif tam sayı ,x,$a_0$,..... $a_n$ reel sayılar*

*Adım 1: (Koşullama)*

*S= $a_0$ , k=1*

*Adım 2 : (Bir sonraki terimi ekle)*

*While k $\leq$ n*

*(a) S$\leftarrow$S+$a_k x^k$*

*(b) **k**$\leftarrow$k+1*

***EndWhile***

*Adım 3: Print s*

2.adımda k $\leq$ n kontrolünü k=1,2.....n+1 için n+1 kere yapıyoruz .Herhangi bir k $\leq$ n için 1 karşılaştırma, 2 toplama ,1 çarpma ve (3k-2) işlem $x^k$ hesabı için yapıyoruz. O halde 3k-2+4=3k+2 işlem yapılmaktadır. k=1,2,...n için 5+8+11+.....+3n+2=$(3n^2 +7n)/2$ işlem yapılıyor. k=n+1için yapılan karşılaştırmayı da eklersek $( (3n^2 +7n)+1) /2$ işlem yapılmaktadır . Görüldüğü gibi algoritmanın karmaşıklılığı bir polinomdur , yani $1.5n^2+3.5n+0.5$ dir. Burada polinomun derecesi karmaşıklıkta bizim için çok dalış önemlidir. Bu örnekte $n^2$ karmaşıklılık etkin olacaktır. Genellikle karmaşıklılığı $n^k$ ′nın bir polinomu olan Algoritma $n^k$ karmaşıklılığında kabul edilir.

Örnek 1.8 (Horner Yöntemi)

Şimdi polinom hesaplama için çok daha etkin bir algoritmayı ele alalım.

$a_3x^3 +a_2x^2 +a_1 x+a_0 = x (a_3 x^2+ a_2 x+ a_1)+a_0 = x(x(a_3 x + a_2)+ a_1) +a_0= x(x(a_3 (x) + a_2) + a_1)+ a_0$

**Algoritma**

| | |
|---|---|
| *Adım 1* | *S= $a_n$ ve k=1* |
| *Adım 2* | *While k $\leq$n* |
| | *(a) S $\leftarrow$ XS + $a_{n-k}$* |
| | *(b) k $\leftarrow$ k+1* |
| | **EndWhile** |
| *Adım 3* | *Print s.* |

Adım 2 de her k $\leq$n için 1 karşılaştırma, 1çarpma, 2 toplama, 1 çıkarma yapılmaktadır. n için 5n+1 işlem yapılmaktadır. Önceki algoritma ile karşılaştırırsak, örneğin n=10 için 51 işleme karşı 186 işlem olacaktır. n'in büyük değeri için fark çok daha artacaktır. Horner yönteminin karmaşıklılığı n olduğu için ilkine göre çok daha üstün bir algoritmadır.

Şimdiye kadar ele aldığımız algoritmalar, karmaşıklılığını tam olarak hesaplayabildiğimiz için yeterince basit algoritmalardır. Ancak, çoğunlukla tam işlem sayısı ,sadece n değerine bağlı değildir. Örneğin n adet sayıyı sıralama algoritması sayıların başlangıçtaki durumuna bağlı olarak daha az yada çok adım gerektirebilir.Böyle durumlarda işlem sayısını üstün halde hesaplarız ve gerçek işlem sayısının buna eşit yada daha az olduğunu söyleriz.

**Alt Küme Üreten Algoritma**

Uzay mekiği problemimizde gereken alt küme hesaplama için bir algoritma geliştirelim. n elemanlı ve elemanları $x_1,x_2,.....,x_n$ olan bir S kümesi ele alalım. Bu kümeyi 0 ve 1'lerden oluşmuş bir küme şeklinde modellersek,alt kümeleri de aynı biçimde üretebiliriz.S kümesindeki 0 elemanı k.sırada ise $x_k \notin S$ ve 1. sıradaki 1 değerinde $x_J \in S$ anlamındadır.Örneğin n=3 ise $2^3$ =8 alt küme 3 elemanlı ikili katar şöyle temsil edilecektir. S:{x1,x2,x3}

000 {$\emptyset$}

001 {$x_3$}

010 { $x_2$}

011 { $x_2,x_3$ }

100 { $x_1$}

101 { $x_1,x_3$}

110 { $x_1,x_2$}

111 { $x_1,x_2$ ,$x_3$}

**Bir sonraki alt katar algoritması**

*Verilen n pozitif tam sayısı için 0ve 1 den oluşan $a_1.a_2 \ldots\ldots a_n$ katarını bir sonraki alt kümeye karşılıklı duran katarı hesaplayan algoritma.*

*Adım 1 (koşullama)*

*k=n*

*Adım 2 (en sağdaki sıfırı bul)*

*While $k \geq 1$ and $a_k = 1$*

*$k \leftarrow k - 1$*

*End While*

*Adım 3 (Sıfır var ise ,sonraki katarı oluştur.)*

*If $k \geq 1$*

*Adım 3.1 (En sağdaki sıfırı bir ile değiştir.)*

*$a_k \leftarrow 1$*

*Adım 3.2 (Arkasındaki birleri sıfır yap)*

*For j = k+1 To n*

*$a_j = 0$*

*End For*

*Adım 3.2 (Çıkış)*

*Print $a_1. a_2 \ldots\ldots a_n$*

*Other wise*

*Adım 3.3 (Alt küme yok)*

*Print ' Katarın tüm elemanları birdir.'*

*End If*

Şimdi de bu algoritmayı sırt çantası problemine nasıl uygulayacağımızı düşünelim. $W_i$ , i . deneyin ağırlığı olsun. Bu durumda $a_1$. $a_2$ .... $a_{12}$ katarı ile temsil ettiğimiz alt küme için $a_1W_1 + a_2 W_2 + \ldots + a_{12}W_{12} \leq$ 1400 karşılaştırmasını yapacağız. Burada $a_i$ =1 i. deney var, $a_i$ =0 i. deney yok anlamında yorumlanacaktır. Bu hesap için karşılaştırma ve indeks hesapları hariç n çarpma ve n-1 toplama olamak üzere 2n-1 işlem vardır. n deney için $2^n$ alt küme vardır. Buna göre bu yöntemin karmaşıklığı C bir sabit olmak üzere Cn $2^n$ olacaktır. Aşağıdaki tabloda saniyede $10^6$ işlem yapabilen bir bilgisayarda n çeşitli değerleri için topla işlem süresi ve bir karşılaştırma için 1000 $n^2$ 'nin alacağı süre verilmiştir.

| n | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| $n.\ 2^n$ | 0.001s | 21 s | 9 s | 1.4 yıl | 1785 yıl |
| $1000n^2$ | 0.1 s | 0.4 s | 0.9 s | 1.6 s | 2.5 s |

Genel olarak , bir algoritmada eğer n sayısı birden büyük bir sayının kuvveti olarak geliyorsa, n 'in bir polinoma göre çok daha hızlı artacaktır. n ! ise çok daha hızlı artacaktır. Aşağıdaki tablo çeşitli n değerleri için farklı n fonksiyonlarında yine saniyede $10^6$ işlem yapan bilgisayarın toplam sürelerini vermektedir.

| n | 10 | 30 | 60 |
|---|---|---|---|
| $\log_2 n$ | 0.0000033 s | 0.0000049 s | 0.0000059 s |
| $n^{1/2}$ | 0.0000032 s | 0.0000055 s | 0.0000077 s |
| n | 0.00001s | 0.00003 s | 0.00006 s |
| $n^2$ | 0.0001 s | 0.0009 s | 0.0036 s |
| $n^2 + 10n$ | 0.0002 s | 0.0012 s | 0.0042 s |
| $n^{10}$ | 2.8 saat | 19 yıl | 19174 yıl |
| $2^n$ | 0.001 s | 18 ay | 36559 yıl |
| n! | 3.6 s | $8.4\ 10^{18}$ yıl | $2.610^{68}$ yıl |