1. **Hafta** Algoritma çeşitleri ve çözme tekniklerinden bahsedilmiş
2. **Hafta**nın konusu **ANALİZ**

Analiz :
Bütünü oluşturan parçalardan önemli olanların ayrıştırılması

Algortima Analizi :
Algoritmanın verimliliğini inceleyeceğiz

1. **Zaman Analizi ☑**
   Zaman karmaşıklığı (Zamanın analiz edilmesi) Algoritmanın ne kadar hızlı çalıştığını ifade eder.

2. **Bellek Analizi**
   Bellek türü burada önemli hale geliyor.
   a. Önbellek
   b. RAM
   c. Sabitdisk

İşlemcinin algoritma için çalıştığı süre; yani temel işlemler alınacak

Temel işlemler

Hız artar ↓

/	Bölme	
*	Çarpma	
+	Toplama	☞Eşit Kabul Edilir
-	Çıkartma	

Sıkça Yapılan Hatalar

• Karmaşıklığı bulmak için sadece döngüleri saymakla yetinmeyin.
– 2 içi içe döngünün 1 den N^2 kadar döndüğünü düşünürsek karmaşıklık $O(N^4)$ olur.

• $O(2N^2)$ veya $O(N^2+N)$ gibi ifadeler kullanmayın.
– Sadece baskın terim kullanılır.
– Öndeki sabitler kaldırılır.

• İç içe döngüler karmaşıklığı direk etkilerken art arda gelen döngüler karmaşıklığı etkilemez.

POLİNOM DEĞERLENDİRMESİ

X^n çalışma zamanı analizi

$x,\ (x)x,\ (x^2)x\ ,\ (x^3)x\ \ldots\ (x^{n-1})x\ =\ x^n$

1. **adım** Koşullama
   $p = x, k = 1$

2. **adım** Bir sonrakini terim
   while $k < n$
   $p = p.x$
   $k = k + 1$
   end while

3. **adım** Print p

Sonuç değil adım sayısını bulmaya çalışıyoruz

X^3 için algoritmayı çalıştıralım;

while $1<3$
$p = x.x$
$k = 1+1$
end while

while $2<3$
$p = (x^2).x$
$k = 2+1$
end while

while $3<3$
**döngü biter*
p'yi yazar

X^3 için algoritma analizi

3	*Karşılaştırma*
2	*Çarpma*
2	*Toplama*

+

X^n için Analizi Gerçekleştirme

$$\left.\begin{array}{ll} n & \text{Karşılaştırma} \\ n-1 & \text{Çarpma} \\ n-1 & \text{Toplama} \end{array}\right\rangle \quad T(n) = 3n-2$$

3n-2
Adım sayısı | yapılan işlem sayısı | temel işlemler

X^n için n=100 ise

$T(n) = 3n - 2$

$T(100) = 3.100 - 2 = 298$

$\Rightarrow 298.10^{-6}\,sn$

Her işlem 1µsn=10^{-6} saniye kabul edilir

$\boxed{T(n) = 3n-2}$

n. dereceden polinomun hesaplanması

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots a_3 x^3 + a_2 x^2 + a_1 x^1 + a_0 x^0$$

1. **adım** Koşullama
   $s = a_0, k = 1$

2. **adım** Bir sonrakini terim
   while $k \le n$
   $s = s + a_k x^k$
   $k = k + 1$
   end while

3. **adım** Print p

Algoritma analizi

1 *Karşılaştırma*
2 *Toplama*
1 *Çarpma*
+ 1 x^k

$= (3k-2)+4$
$= 3k+2$

x^n için
$T(n) = 3n - 2$

Son karşılaştırma yapılır fakat işlemler yapılmaz o yüzden son karşılaştırma +1 olarak değerlendirilir

n=3 için analiz;

	1≤3	2≤3	3≤3	4≤3	→ **4** karşılaştırma yapılacak
+	2	2	2	0	→ **6** toplama
*	1	1	1	0	→ **3** çarpma
x^k	1	1	1	0	→ **3** x^k hesabı

Analizi Genelleştirme

$n+1$ *Karşılaştırma*
$2n$ *Toplama*
n *Çarpma*

$4n+1$

$\sum_{k=1}^{n} 3k+2$

$$\sum_{k=1}^{n} 3k+2 = 3\sum_{k=1}^{n} k + \sum_{k=1}^{n} 2 = 3\cdot\frac{n(n+1)}{2} + 2n \Big\rangle \quad \frac{3}{2}n^2 + \frac{7}{2}n + \underset{\text{son karşılaştırma}}{1}$$

$T(n) = 1,5n^2 + 3,5n + 1$

n=100 için;

$T(n) = 1,5n^2 + 3,5n + 1$

$T(100) = 1,5\cdot(100)^2 + 3,5\cdot 100 + 1 = 15351 \Rightarrow 15351\cdot 10^{-6}\, sn$

n. dereceden polinomun hesaplanması
Algoritma daha iyileştirilemez mi?

$$f(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0$$
$$f(x) = x \cdot (x \cdot \underbrace{(\underbrace{x \cdot (a_3)}_{çarp} + a_2)}_{ekle} + a_1) + a_0$$

1. **adım** Koşullama
   $s = x \cdot s + a_{n-k}$

2. **adım** Bir sonrakini terim
   while $k \le n$
   $s = x \cdot s + a_{n-k}$
   $k = k + 1$
   end while

3. **adım** Print p

Her iterasyonda;

1	*Karşılaştırma*
1	*Çarpma*
2	*Toplama*
+ 1	*Çıkartma*
5	*İşlem*

Genelleme

$$\sum_{k=1}^{n} 5 = 5n$$

$$T(n) = 5n + \underbrace{1}_{son\ karşılaştırma}$$

n=100 için;
$T(n) = 5n + 1$
$T(100) = 5 \cdot 100 + 1 = 501 \Rightarrow 501 \cdot 10^{-6}\, sn$

Ortalama 30 kat Daha hızlı

$\boxed{T(n) = 5n + 1}$

Insertion sort (Araya ekleme) Algoritması

for $i \rightarrow 1$ **to** $length(A)$
    $j \leftarrow i$
    while $j > 0$ and $A[j-1] > A[j]$
        $swap(A[j], A[j-1])$
        $j \leftarrow j-1$
    end while
end for

k adet yer değiştirdiğini düşünürsek;
k.c adım gerekir (1.eleman için)
$c \cdot 1 + c \cdot 2 + c \cdot 3 + \cdots + c \cdot (n-1)$

En kötü & Ortalama durum analizi

$$T(n) = \frac{n \cdot (n-1)}{2} \cdot c = c \cdot \frac{n^2 - n}{2} = c \cdot \frac{n^2}{2} - c \cdot \frac{n}{2}$$

En kötü & ortalama durum

$$T(n) = c \cdot \frac{n^2 - n}{2}$$

En iyi durum analizi
Dizi zaten sıralıdır
Döngü (n-1) defa çalışır
c zamanda atama yapılır

$$T(n) = c \cdot (n-1) = c \cdot n - c$$

En iyi durum

$$T(n) = c(n-1)$$

İkili Arama (Binary Search) Algoritması

Dizinin sıralı olduğu kabul ediliyor

12	25	49	54	66	102	205	302

66 sayısını aradığımızı düşünelim !

Dizi sıralanmış olduğundan, dizinin ortasında bulunan sayı ile aranan sayıyı karşılaştırarak arama boyutunu yarıya düşürülür ve bu şekilde devam edilir.

~~12~~	~~25~~	~~49~~	54	66	102	205	302

Seçtiğinde sol taraf iptal (rastgele) sağ tarafta aranmaya devam edilir.

~~12~~	~~25~~	~~49~~	~~54~~	66	102	~~205~~	~~302~~

~~12~~	~~25~~	~~49~~	~~54~~	66	~~102~~	~~205~~	~~302~~

En iyi durum analizi

c sabit zamanda $T(n)=c$

En iyi durum
$T(n)=c$

En kötü durum analizi

Başta veya sonda olması durumu

n elemanlı bir dizi (Genelleme)

1.adım	n **eleman**	toplam karşılaştırma sayısı	**1**
2.adım	$\frac{n}{2}$ **eleman**	toplam karşılaştırma sayısı	**2**
3.adım	$\frac{n}{4}$ **eleman**	toplam karşılaştırma sayısı	**3**
$\vdots$	$\vdots$		$\vdots$
j.adım	$\frac{n}{2^{j-1}}$ **eleman**	toplam karşılaştırma sayısı	**j**

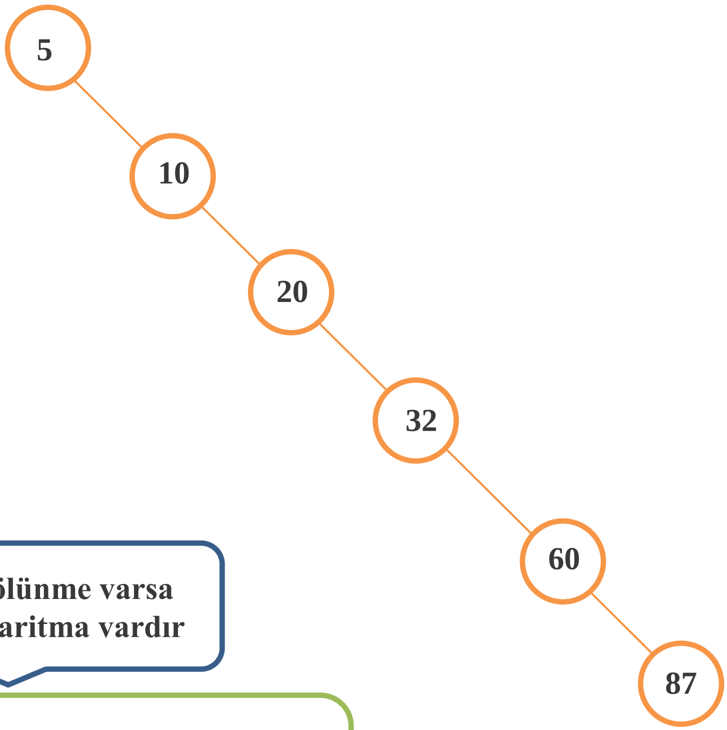
$$\frac{n}{2^{j-1}}=1 \quad \Rightarrow \quad 2^{j-1} \quad \Rightarrow \quad \log_2 n = j-1 \quad \Rightarrow \quad j=\log n+1$$

$$T(n)=\log n+1$$

En kötü durum
$T(n)=\log n+1$

İkili arama ağacı (Binary Search tree)

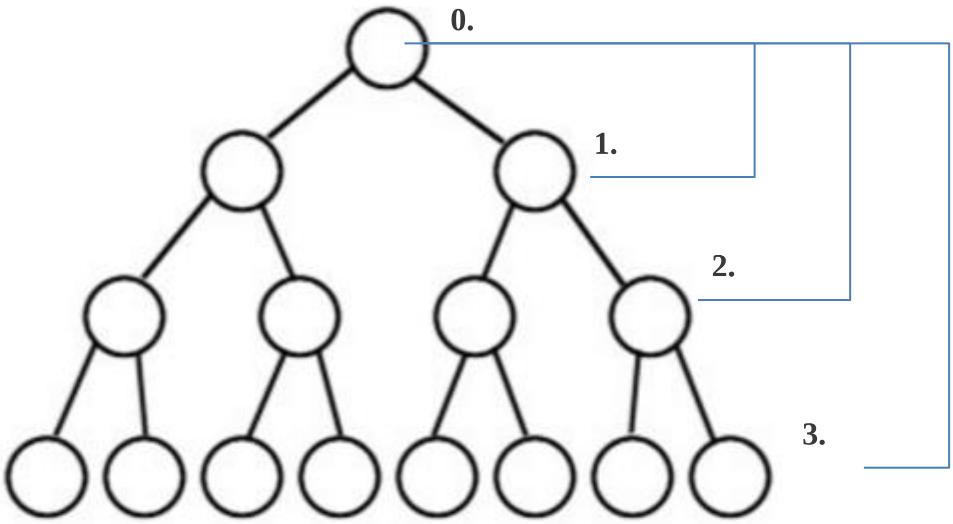
5	10	20	32	60	87



En kötü durum
İkili arama ağacı için en kötü durum LİSTE
$T(n) = h$

Bölünme varsa logaritma vardır

En iyi durum
Tam dolu ağaç
$T(n) = \log(n+1) - 1$



h=yükseklik

n=düğüm sayısı

$n = 2^{h+1} - 1$

$n + 1 = 2^{h+1}$

$h = \log(n+1) - 1$