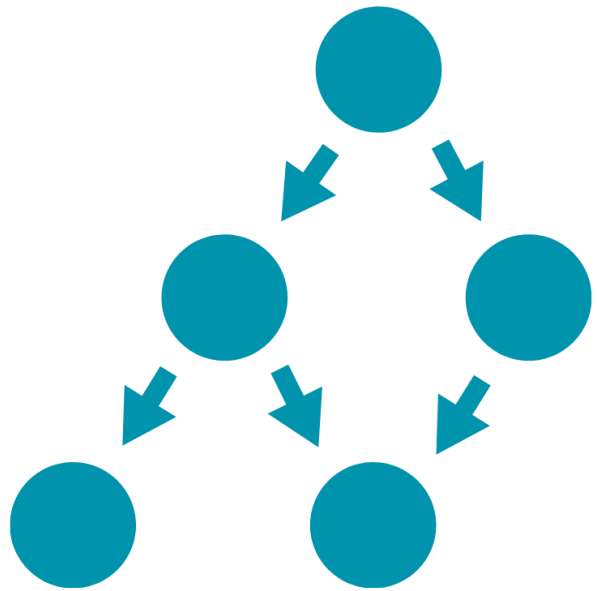


VERİ YAPILARILARI VE ALGORİTMALAR

Algorithm Analysis and Asymptotic Notations

Algoritma

- Karmaşık matematiksel hesapların belirli bir düzenek tarafından yapılmasını sağlayan Turing makinesinin özel bir durumunu ifade eden hesaplamalı bir modeldir.

1. Girişi olmalıdır.
2. Çıkışı olmalıdır.
3. Her adımı tanımlı olacak.
4. Sınırlı sayıda adımda olacak.
5. Etkililik (zamansal/uzaysal analiz)

- Doğruluk
- Tekrar edilebilirlik
- Sürdürülebilirlik

- İşlevsellik
- Sağlamlık
- Modülerlik

- Kullanıcı-dostu
- Basit
- Genişleyebilir

- Güvenilir
- Ölçeklenebilir

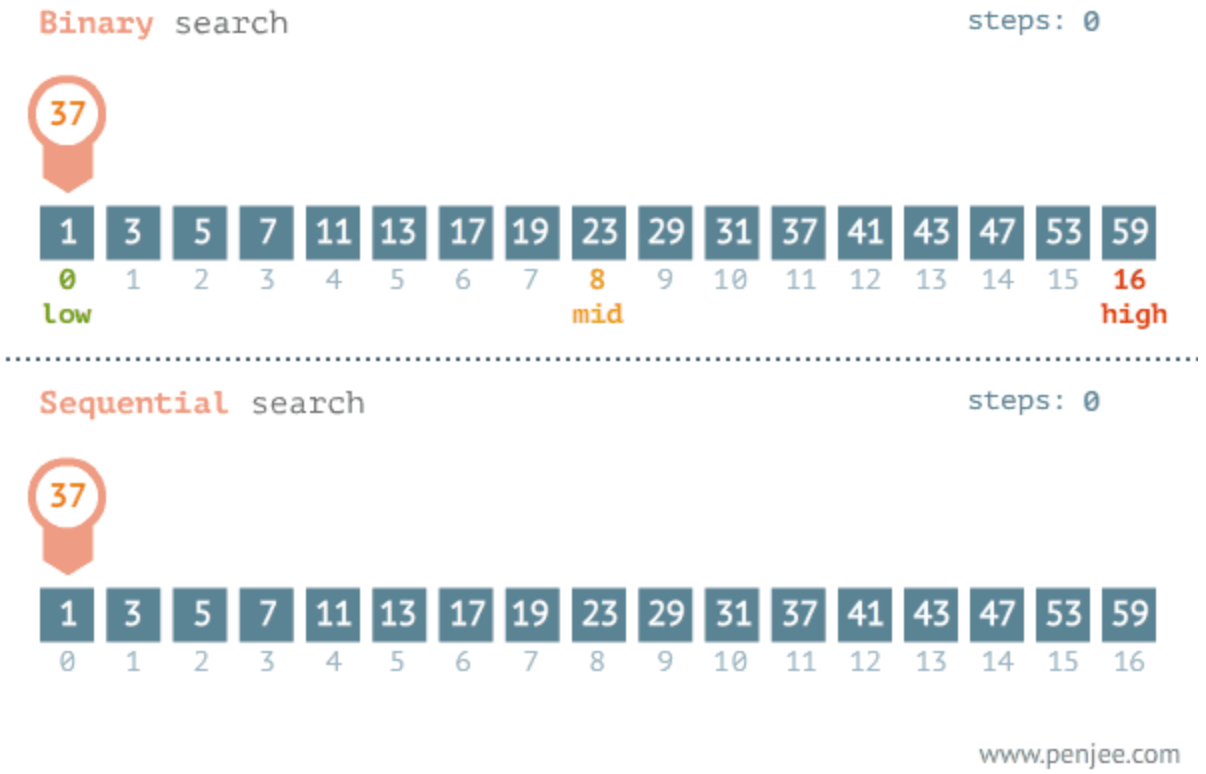
Zaman Karmaşıklığı
(Time Complexity)

$T(n)$

- Algoritmanın girdisi
  - Dizi (boyutu)
  - Polinom (polinom derecesi)
  - Matris (eleman sayısı)
  - İkilik veri (bit sayısı)
  - Graf (kenar ve düğüm sayısı)

- Algoritmanın dayandığı paradigma
  - Backtracking, Branch and bound, Brute-force search, Divide and conquer, Dynamic programming, Greedy algorithm, Prune and search

T(n)


Binary search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
low
mid
high
Sequential search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
www.penjee.com

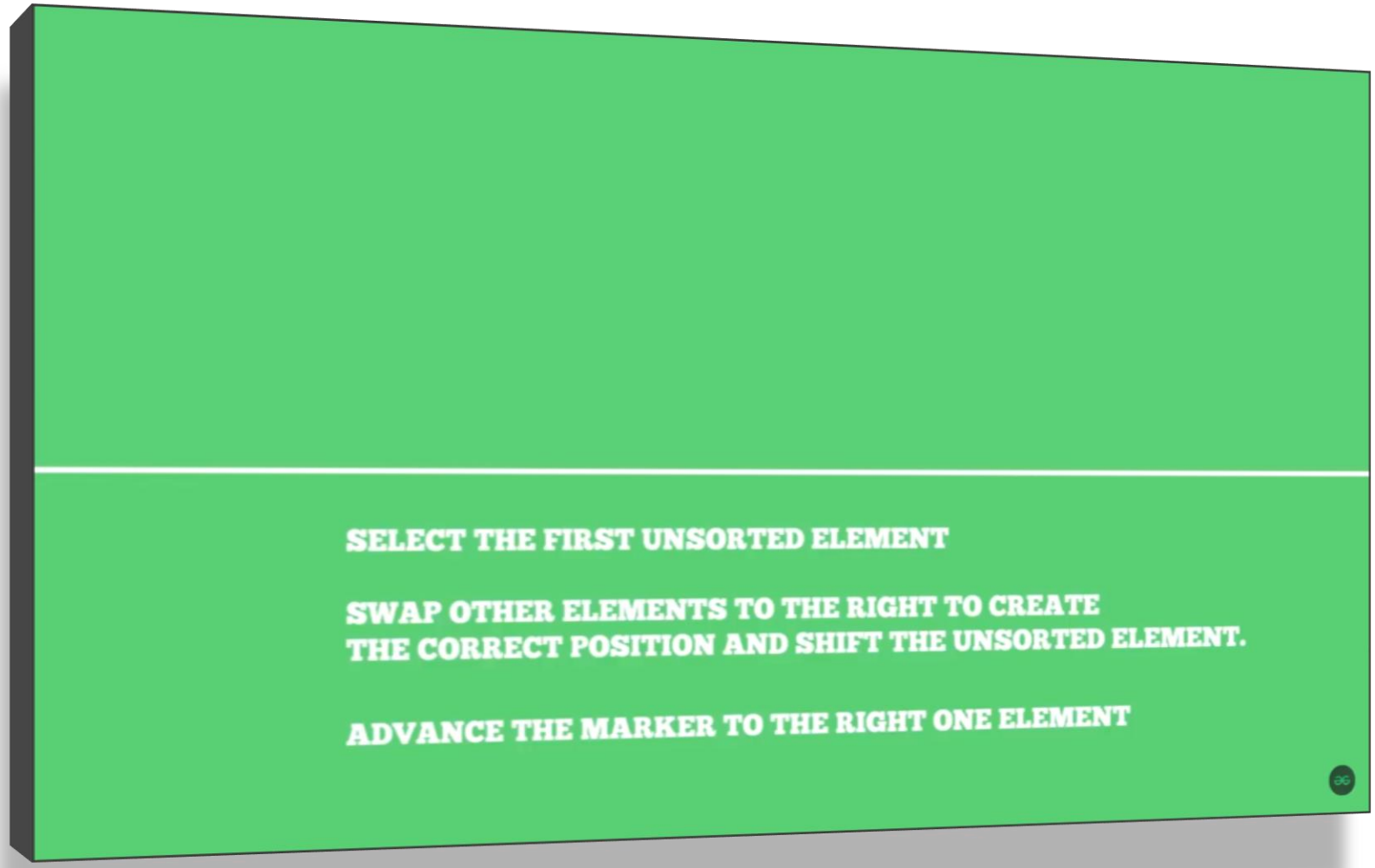
Zaman Karmaşıklığı
(Time Complexity)

T(n)

- Çalışma süresi.
- Süre zamanı değil; işlem sayısını ifade eder.
- Sabit işlem süresi.
- Giriş boyutunun büyümesi.
- Asimptotik notasyonlar.
  - Time complexity (n)
  - Space cost (n)
  - Space complexity (n)

Bir Algoritmanın Yürütme Zamanı Analizi

(Running time analysis)



https://www.geeksforgeeks.org/insertion-sort/

Bir Algoritmanın Yürütme Zamanı Analizi

(Running time analysis)

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j \leftarrow 2$ **to** $length[A]$	c_1	n
2 **do** $key \leftarrow A[j]$	c_2	$n-1$
3 ▷ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] \leftarrow key$	c_8	$n-1$

Bir Algoritmanın Çalışma Zamanı Analizi

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1)$$
$$+ c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right)$$
$$+ c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1)$$
$$= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n$$
$$- (c_2 + c_4 + c_5 + c_8) .$$

Bir döngünün çalışma zamanı analizi

Döngü

	Maliyet	**Defa**
`i = 1;`	c1	1
`sum = 0;`	c2	1
`while (i <= n) {`	c3	n+1
`    i = i + 1;`	c4	n
`    sum = sum + i;`	c5	n
`}`		

Total Cost = c1 + c2 + (n+1)*c3 + n*c4 + n*c5

Bu algoritmanın büyüme hızı **n** değeri ile orantılıdır.

Bir döngünün çalışma zamanı analizi

```
int kareToplami(int N)
{
    int i, toplam = 0;
    for (i = 0; i < N; i++)
    {

        toplam += i * i;
    }
    return toplam;
}
```

Bir döngünün çalışma zamanı analizi

```
double OrtalamaHesapla(double[] A)
{
    double ortalama = 0, toplam = 0;
    for (int i = 0; i < A.Length; i++)
    {

        toplam += A[i];
    }
    ortalama = toplam / A.Length;
    return ortalama;

}
```

Koşul ifadesi içeren bir döngünün çalışma zamanı analizi

Eğer koşul ifadesi içeren bir kod parçacığı var ise

	Maliyet	**Defa**
`if (n < 0)`	c1	1
`  absval = -n`	c2	1
`else`		
`  absval = n;`	c3	1

Toplam Maliyet <= c1 + max(c2,c3)

Ardışık programların çalışma zamanı analizi

```
void kosulBloklari(int n)
{
    int toplam = 0;
    if (n % 2 == 0)
        for (int i = 1; i <= n; i++)
            toplam += i * i;

    else
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                toplam += i;
    Console.WriteLine(toplam);
}
```

Koşul ifadesi içeren bir döngünün çalışma zamanı analizi

```
double EnKucukBul(double[] A)
{
    double enKucuk = A[0];
    for (int i = 1; i < A.Length; i++)
    {

        if (A[i] < enKucuk)
            enKucuk = A[i];
    }
    return enKucuk;
}
```

İç içe döngünün çalışma zamanı analizi

İçiçe döngüler

	Maliyet	Defa
`i=1;`	c1	1
`sum = 0;`	c2	1
`while (i <= n) {`	c3	n+1
`  j=1;`	c4	n
`  while (j <= n) {`	c5	n*(n+1)
`  sum = sum + i;`	c6	n*n
`  j = j + 1;`	c7	n*n
`  }`		
`  i = i +1;`	c8	n
`}`		

Toplam Maliyet = c1 + c2 + (n+1)*c3 + n*c4 + n*(n+1)*c5+n*n*c6+n*n*c7+n*c8

Bu algoritmanın çalışma zamanı n^2 orantılıdır.

İç içe döngünün çalışma zamanı analizi

```
int matrisElemanlarininToplami(int[,] A)
{
    int toplam = 0;
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            toplam += A[i, j];
        }
    }
    return toplam;
}
```

İç içe döngünün çalışma zamanı analizi

```
int[,] toplamMatris(int[,] A, int[,] B)
{
    int n = A.GetLength(0);
    int m = A.GetLength(1);
    int[,] C = new int[n,m];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            C[i, j] = A[i, j] + B[i, j];
        }
    }
    return C;
}
```

Ardışık programların çalışma zamanı analizi

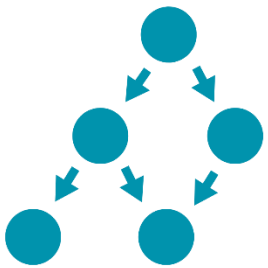
```
void ardisikProgramlama(int n)
{
    int toplam = 0;
    for (int i = 1; i <= n; i++)
        toplam++;

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            toplam++;
    Console.WriteLine(toplam);
}
```

Döngü sayısının ardışık artmadığı durumları değerlendirme

```
void hesapla(int k)
{
    int sayac = 0;
    while (k>1)
    {
        sayac++;
        k /= 2;
    }
    Console.WriteLine(sayac);
}
```

Özyinelemeli (rekürsif) bir metodun çalışma zamanı analizi

```
int faktoriyel(int n)
{
    if (n <= 1)
        return 1;
    else
        return n * faktoriyel(n - 1);
}
```



Veri Yapıları ve Algoritmalar

ZAFER CÖMERT
Öğretim Üyesi