

Algoritmalar

Analiz

Tasarım

Soru: Performanstan daha önemli ne var?
Güvenlik, kullanıcı-dostluğu, doğruluk,
modülerlik, programcı zamanı, basitlik

Sıralama Problemi

Girdi: Sayı dizisi $\langle a_1, a_2, \dots, a_n \rangle$

Gökti: Sayı dizisi $\langle a'_1, a'_2, \dots, a'_n \rangle$

Öyleki $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Insertion (Araya Yerlestirme) Sıralaması

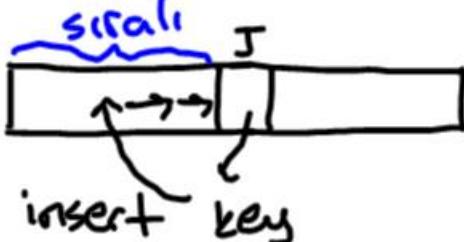
Insertion_Sort(A, n) // A[1...n] i sıralar

for j=2 to n
key = A[j]
i=j-1
while i>0 and A[i]>key
 A[i+1]=A[i]
 i=i-1
A[i+1]=key

pseudocode
sözde kod
yalancı kod

indentation: girişi, hizalama

Değişmez: A:



Örnek: A:	8	2	4	9	3	6
J=2	2	8	4	9	3	6
J=3	2	4	8	9	3	6
J=4	2	4	8	9	3	6
J=5	2	3	4	8	9	6
J=6	2	3	4	6	8	9

```

1: for j = 2 to A.length do
2:   key = A[j]
3:   i = j - 1
4:   while i > 0 and A[i] > key do
5:     A[i + 1] = A[i]
6:     i = i - 1
7:   end while
8:   A[i + 1] = key
9: end for

```

Girişim Zamanı

1. Girdinin boyutuna bağlı: 100 ile 10^9 girdi boyutuna göre parametrize edilecek
2. Girdinin kendisine bağlı (örn: girdi sıralı ise)

2 4 5 8 10 12 17 en hızlı (iyi)

Ters Sıralı Olsaydı

12 10 8 7 4 3 2 en yavaş (kötü)

Analiz Geçitleri

1. En kötü durum (worst-case) genellikle
 $T(n)$ = n boyutlu girdi için algoritmanın maximum çalışma zamanı
2. Ortalama durum (average case) bazen
 $T(n)$ = n boyutlu girdi için algoritmanın beklenen çalışma zamanı
3. En iyi durum (best case) nadiren

Asimetotik Analiz

$n \rightarrow \infty$ iken $T(n)$ in büyümeye bakarız.

Asimetotik Notasyon

Θ notasyonu: düşük dereceli terimleri sil
Büyük katsayıyı ihmal et.

Örnekler: a. $T(n) = 3n^3 + 90n^2 - 50n + 60$

$$T(n) = \Theta(n^3)$$

b. $T(n) = 3000n^2 + 100n \log n$

$$T(n) = \Theta(n^2)$$

b algoritması a algoritmasından daha hızlıdır.

$n=10$ için a daha hızlı olabilir.

Ama biz asimetotik analize bakıyoruz, yani
bizim için $n \rightarrow \infty$ iken $T(n)$ nin büyümesi
öneMLİ

Araçla Yerleştirme Algoritmasının Analizi

En iyi durum: $\xrightarrow{\text{sabit zaman}}$

$$T(n) = \sum_{j=2}^n \Theta(1) = \Theta\left(\sum_{j=2}^n 1\right)$$

$$= \Theta(n-1) = \Theta(n)$$

```

1: for j = 2 to A.length do
2:   key = A[j] \Theta(1)
3:   i = j - 1 \Theta(1)
4:   while i > 0 and A[i] > key do
5:     A[i + 1] = A[i]
6:     i = i - 1
7:   end while
8:   A[i + 1] = key \Theta(1)
9: end for

```

En kötü durum:

$$T(n) = \sum_{j=2}^n \Theta(j) = \Theta\left(\sum_{j=2}^n j\right)$$

$$= \Theta\left(\frac{n(n+1)}{2} - 1\right)$$

$$= \Theta\left(\frac{n^2}{2} + \frac{n}{2} - 1\right)$$

$$= \Theta(n^2)$$

```

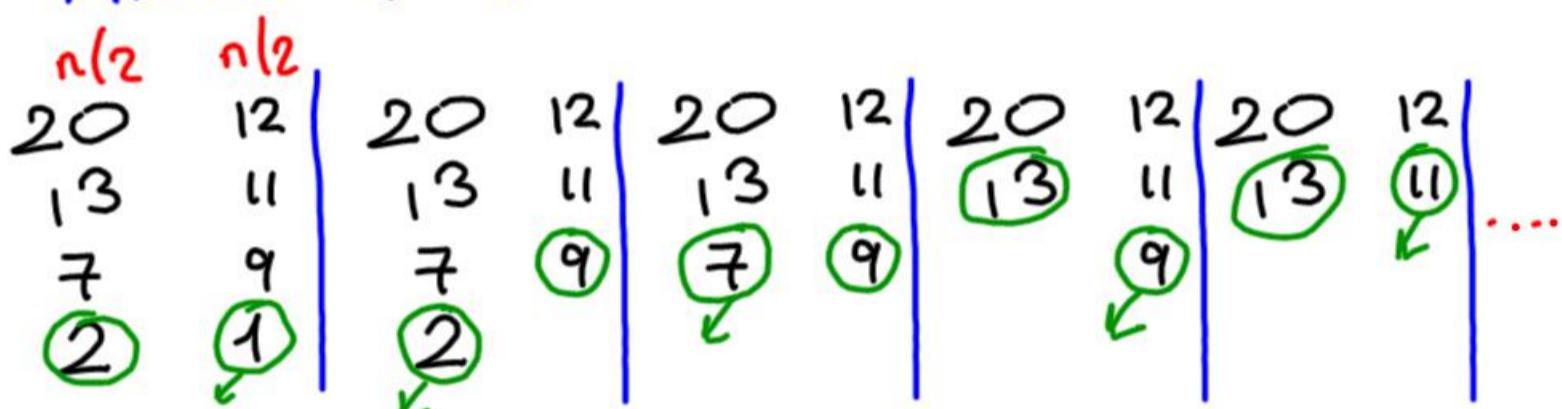
1: for j = 2 to A.length do
2:   key = A[j]
3:   i = j - 1
4:   while i > 0 and A[i] > key do
5:     A[i + 1] = A[i]
6:     i = i - 1
7:   end while
8:   A[i + 1] = key
9: end for

```

Birleştirme Sıralaması (Merge Sort)

$T(n)$	Merge-Sort $A[1 \dots n]$	reküratif özünelemeli
$\Theta(1)$	if $n=1$ return	ceiling
$T(n/2)$	1. Merge-Sort $A[1 \dots \lceil n/2 \rceil]$	
$T(n/2)$	2. Merge-Sort $A[\lceil n/2 \rceil + 1 \dots n]$	
$\Theta(n)$	3. 2 sıralı diziyi ^{alt} birleştir.	

Anahtar işlem: Birleştir



B: 1, 2, 7, 9, 11 ...

Girişim Zamanı: $n(1+1) = \Theta(n)$

Çünkü n eleman birleşiyor.

A	1 2 3 4 5 6 7 8 9 = n
	10 7 8 15 7 3 2 4

10 7 8 | 15

:

10 7 | 8

:

10 | 7

:

7 10 | 8

↓

7 8 10 | 15

7 8 10 | 15

1 5 7 8 10 | 2 3 4 7

1 2 3 4 5 7 7 8 10

Birleştirme Sıralaması Çalışma Zamanı

$$T(n) = 2T(n/2) + \Theta(n) + \Theta(1)$$

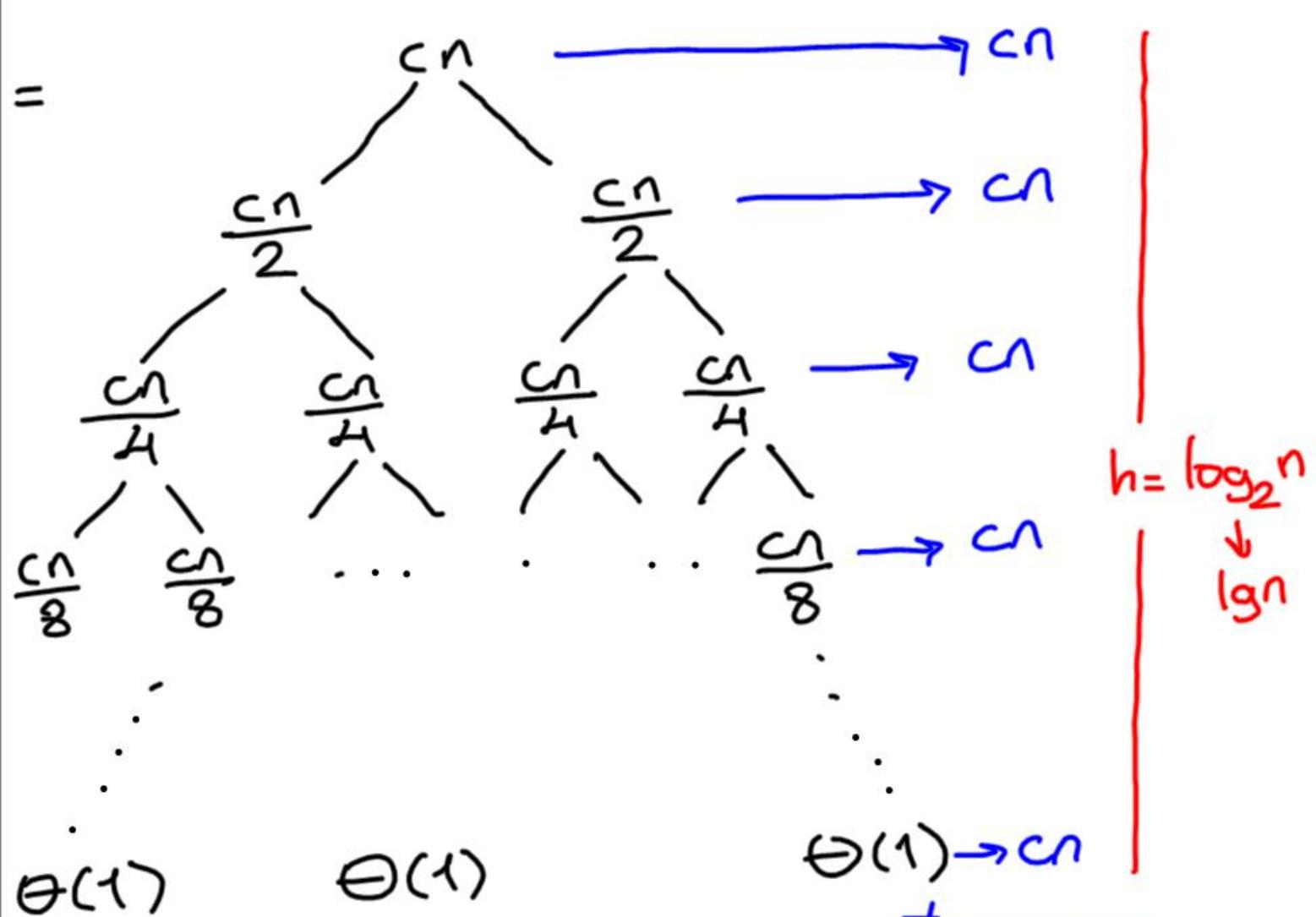
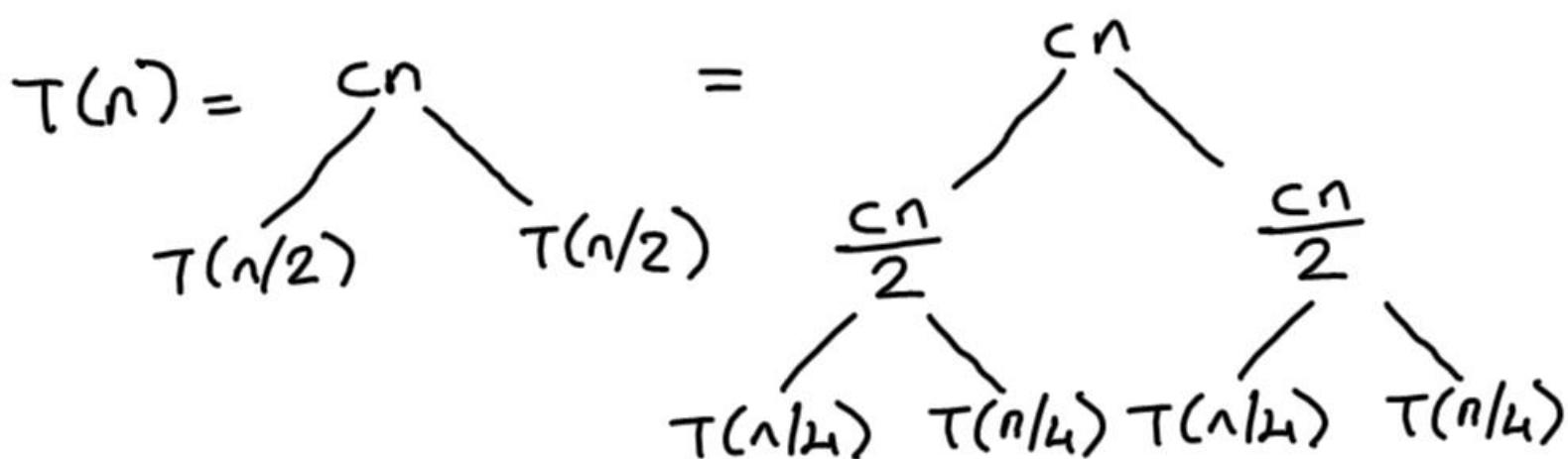
$T(n) = 2T(n/2) + \Theta(n) \rightarrow$ yineleme bağıntısı recurrence

Yineleme Ağacı

$$T(n) = 2T(n/2) + cn \quad c \text{ sabit bir sayı}$$

$$T(n/2) = 2T(n/4) + \frac{cn}{2}$$

$$T(n/4) = 2T(n/8) + \frac{cn}{4}$$



64
32
16
8
4
2
1

$$T(n) = cn \log n$$

$$T(n) = \Theta(n \log n)$$

birleştirme sıralamasının çalışma zamanı

Soru: Aşağıdaki fonksiyonlar, asimptotik büyümelerine göre küçükten büyüğe doğru sıralayın.

$$n, \sqrt{n}, n^2, 2^n, \lg n, n^n, n^3, n \lg n, n!$$

$\xleftarrow{\text{iki}}$ $\xrightarrow{\text{kötü}}$

$$\lg n < \sqrt{n} < n < n \lg n < n^2 < n^3 < 2^n < n! < n^n$$

Asimptotik Notasyonlar

$$O, \Omega, \Theta$$

Büyük O notasyonu (üst sınır)

Tüm $n \geq n_0$ değerleri için, $c > 0$ ve $n_0 > 0$ sabitleri ile $O(f(n)) \leq c g(n)$ ise $f(n) = O(g(n))$ dir.

Örnek: $2n^2 = O(n^3)$ $c=1$ $n_0=2$

$$2n^2 \leq n^3$$

$n=1$ için sağlanamaz

$n > n_0=2$ için sağlanası yeterli

$2n^3 \neq O(n^2)$ olabilir mi?

$$2n^3 \leq 10000n^2 \quad c=10000$$

$$n_0=100$$

tüm $n \geq 100$ için sağlanır mı?

Yanlış

büyük omega

Ω

notasyonu (alt sınır)

Tüm $n \geq n_0$ değerleri için, $c > 0$ ve $n_0 > 0$ sabitleri ile $0 \leq c g(n) \leq f(n)$ ise $f(n) = \Omega(g(n))$ dir.

Örnek: $3n = \Omega(\sqrt{n})$

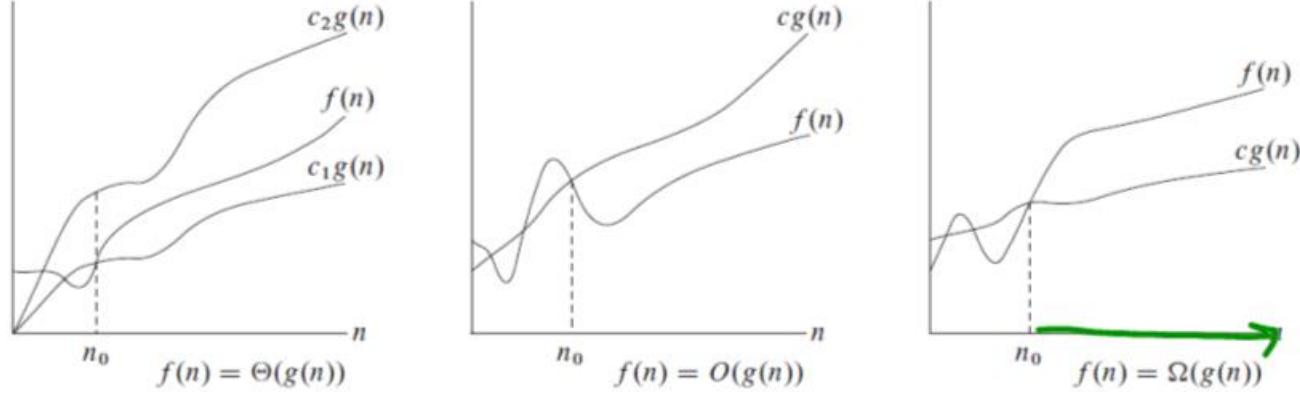
Θ notasyonu (sıkı sınır)

$f(n) = \Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

$$\frac{1}{2}n^2 - 2n = O(n^2) = O(n^3)$$

$$\frac{1}{2}n^2 - 2n = \Omega(n^2) = \Omega(n)$$

$$\frac{1}{2}n^2 - 2n = \Theta(n^2)$$

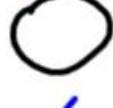


Analoji

üst sınır

alt sınır

sıkı sınır



\leq

\geq

$=$

$$2n^2 = O(n^3)$$

$$\sqrt{n} = \Omega(\lg n)$$

Yineleme Bağıntıları

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n)$$

$$T(n) = 3T(n/3) + \Theta(n)$$

$$T(n) = 4T(n/2) + \Theta(1)$$

$$T(n) = T(n-1) + \Theta(n)$$

$$T(n) = T(n/2) + T(n/3) + \Theta(\sqrt{n})$$

Yineleme Çözüm Yöntemleri

1. Yerine koyma metodu ✗
2. Master metodu ✓
3. Yineleme ağacı metodu ✓

$$\begin{aligned}T(n) &= 2T(n/3) + \Theta(n) \\T(n) &= 3T(n/4) + \Theta(\sqrt{n}) \\T(n) &= T(n-1) + \Theta(1) \\T(n) &= T(n/2) + T(n/3) + n\end{aligned}$$

1. Master metodu

$T(n) = aT(n/b) + f(n)$
 $a \geq 1, b > 1$ ve f asimptotik pozitif ise
 $f(n) : n^{\log_b a}$ ile karşılaştırın.

Durum 1: Eğer $f(n) = O(n^{\log_b a - \varepsilon}), \varepsilon > 0$
yani $f(n) n^{\log_b a}$ dan daha yavaş büyür
ise **Gözüm:** $T(n) = \Theta(n^{\log_b a})$

Örnek: $T(n) = 4T(n/2) + n = \Theta(n^2)$

n ile $n^{\log_2 4} = n^2$ yi karşılaştır.

$$n = O(n^{2-\varepsilon}) \quad \varepsilon=1 \text{ için}$$

Özaman Durum 1, $T(n) = \Theta(n^2)$

Durum 2: Eğer $f(n) = \Theta(n^{\log_b a})$, yani f ile $n^{\log_b a}$ aynı oranda büyür ise
Gözüm: $T(n) = \Theta(n^{\log_b a} \cdot \lg n)$

Örnek: $T(n) = 4T(n/2) + n^2$
 n^2 ile $n^{\log_2 4} = n^2$ yi karşılaştır.
 $n^2 = \Theta(n^2)$ dir.

Durum 2 dir.

Gözüm: $T(n) = \Theta(n^2 \lg n)$

Durum 3: $f(n) = \Omega(n^{\log_b a} + \varepsilon)$, $\varepsilon > 0$ sabit
 ve $f(n)$ düzenlilik koşulunu sağlıyor ise

Gözüm: $T(n) = \Theta(f(n))$

Düzenlilik koşulu: $a \cdot f(n/b) \leq c \cdot f(n)$ için
 $c < 1$ olan bir sabittir.

Durum 3 de $f(n) = n^{\log_b a}$ dan daha hızlı büyür.

Örnek: $T(n) = 4T(n/2) + n^3$ → $f(n)$
 n^3 ile $n^{\log_2 4} = n^2$ yi karşılaştır.
 $n^3 = \Omega(n^2 + \varepsilon)$ $\varepsilon = 1$ için

**düzenlilik
koşulu** {
$$\begin{aligned} a \cdot f(n/b) &\leq c \cdot f(n) \\ 4 \cdot f(n/2) &\leq c \cdot f(n) \\ 4 \cdot \frac{n^3}{8} &\leq c \cdot n^3 \\ \frac{1}{2} &\leq c \quad \text{buradan } c = \frac{1}{2} < 1 \end{aligned}$$
 seçilebilir. }

Durum 3 olur. O zaman $T(n) = \Theta(n^3)$

$$T(n) = 4T(n/2) + n = \Theta(n^2) \quad \text{durum 1}$$

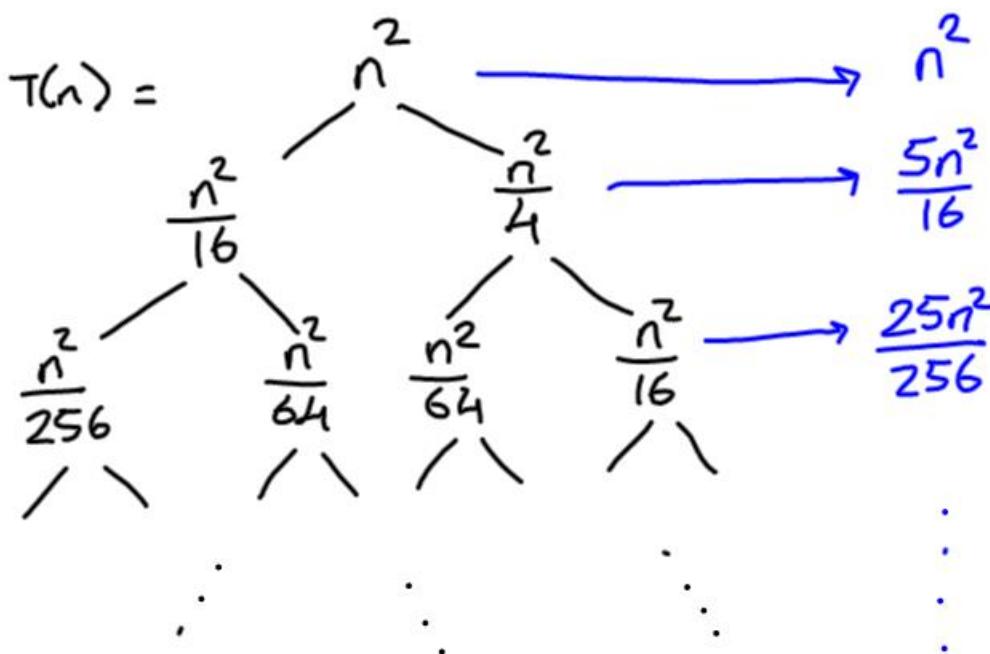
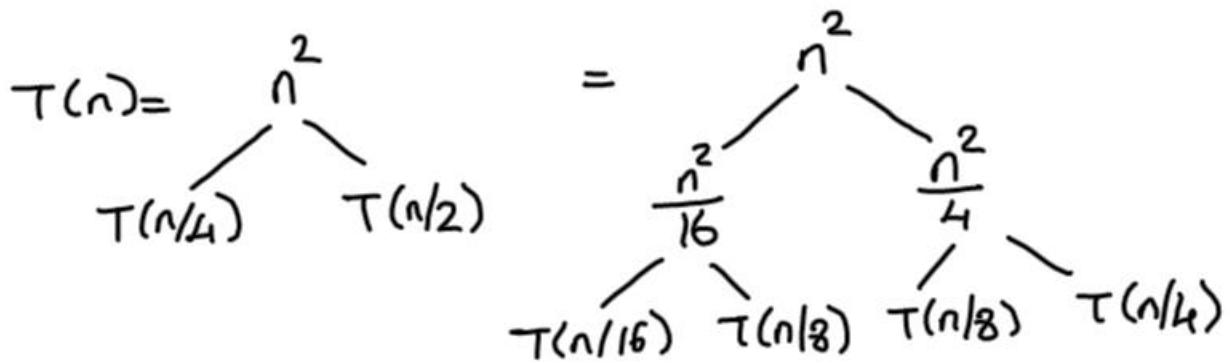
$$T(n) = 4T(n/2) + n^2 = \Theta(n^2 \lg n) \quad \text{durum 2}$$

$$T(n) = 4T(n/2) + n^3 = \Theta(n^3) \quad \text{durum 3}$$

2. Yineleme Ağacı Metodu

$$T(n) = T(n/4) + T(n/2) + n^2 \text{ yinelemesini}\newline \text{çözelim.}$$

$$\begin{aligned} T(n/4) &= T(n/16) + T(n/8) + \frac{n^2}{16} \\ T(n/2) &= T(n/8) + T(n/4) + \frac{n^2}{4} \end{aligned}$$



$\Theta(1)$

$\Theta(1)$

$$T(n) = n^2 \left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \dots + \left(\frac{5}{16}\right)^k \right)$$

$$n^2 \left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \dots + \left(\frac{5}{16}\right)^k \right) < n^2 \underbrace{\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \dots \right)}_{\text{Geometrik seri}}$$

$$\text{NOT: } 1 + r + r^2 + \dots = \frac{1}{1-r} \text{ eğer } |r| < 1$$

$$\underbrace{n^2 \left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \dots + \left(\frac{5}{16}\right)^k \right)}_{\downarrow} < n^2 \frac{1}{1 - \frac{5}{16}} = \frac{16n^2}{11}$$

$$T(n) < \frac{16n^2}{11}$$

$$T(n) = O(n^2)$$

Exercises

4.5-1

Use the master method to give tight asymptotic bounds for the following recurrences.

a. $T(n) = 2T(n/4) + 1.$

$$1 \text{ ile } n^{\log_4 2} = n^{1/2} \text{ yi karşılaştır } 1 = O(n^{1/2-\epsilon}) \quad \epsilon < 1$$

b. $T(n) = 2T(n/4) + \sqrt{n}.$

$$\sqrt{n} ? \sqrt{n} \rightarrow T_n = \Theta(\sqrt{n}) \text{ durum 1: } T(n) = \Theta(\sqrt{n})^2$$

c. $T(n) = 2T(n/4) + n.$

d. $T(n) = 2T(n/4) + n^2.$

$$\text{durum 2: } T(n) = \Theta(\sqrt{n} \lg n)$$

4.4-4

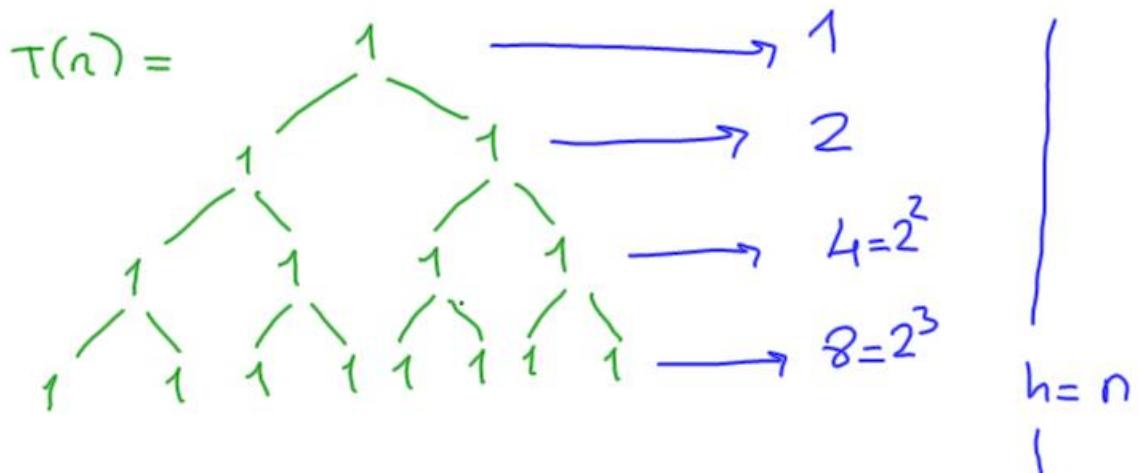
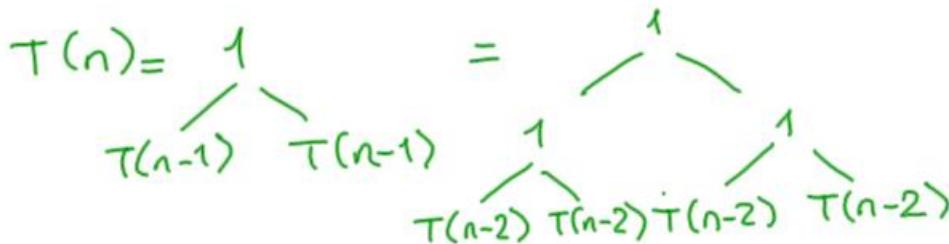
Use a recursion tree to determine a good asymptotic upper bound on the recurrence

$$T(n) = 2T(n-1) + 1. \text{ Use the substitution method to verify your answer.}$$

4.4-5

Use a recursion tree to determine a good asymptotic upper bound on the recurrence

$$T(n) = T(n-1) + T(n/2) + n. \text{ Use the substitution method to verify your answer.}$$



$$\Theta(1) \dots$$

$$\Theta(1) \rightarrow \frac{2^n}{+}$$

$$T(n) = 1 + 2 + 2^2 + \dots + 2^n$$

Geometrik seri:

$$= \frac{2^{n+1} - 1}{2 - 1} = 2^{n+1} - 1$$

$$T(n) = \Theta(2^n)$$

\nwarrow
eksponensiyel çalışma zamanı
(üssel)

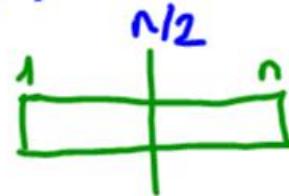
kötü bir algoritma

Böl ve Fethet (Hükmet) Metodu

Divide & Conquer

- 1) Problemi alt problemlere böl Böl
- 2) Alt problemleri özyinelemeli olarak Hükmet
çöz ve hükmet
- 3) Alt problemlerin çözümlerini birleştir

Merge Sort



- 1) Bölmek:

- 2) Hükmetmek: 2 alt problemi özyinelemeli böl
- 3) Birleştirmek: Doğrusal zamanda, $\Theta(n)$

$$T(n) = 2 \cdot T(\underbrace{n/2}) + \Theta(n)$$

alt problem
sayısı

alt problemlerin
boyutu

Birleştirme
zamani

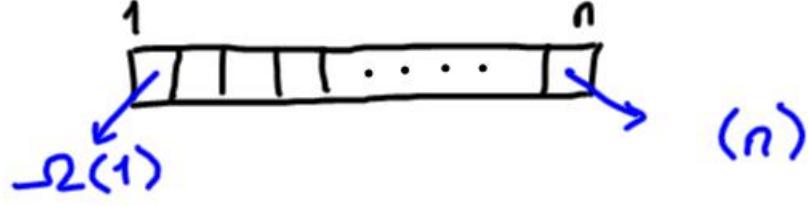
Master metod ile: $n \overset{?}{\leftrightarrow} n^{\log_2 2} = n$
 $n = \Theta(n)$

durum 2, $T(n) = \Theta(n \cdot \lg n)$

Problem: İkili Arama (Binary Search)

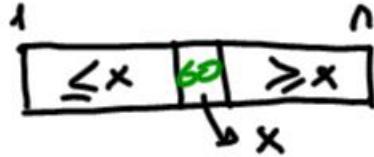
Sıralanmış bir dizide x elemanını bulma

Saf Algoritma:



Böl ve Fethet

1. Böl: Orta elemanı bul x ile karşılaştır.



2. Hükmet: 1 alt dizide özyinelemeli arama yap

3. Birleştir: Yok

Girişim Zamanı: $T(n) = 1 \cdot T(n/2) + \Theta(1)$

1 alt problem
var

Master metod ile: $1 \overset{?}{\leftrightarrow} n^{\log_2 1} = n^0 = 1$
 $1 = \Theta(1)$

durum 2, $T(n) = \Theta(\lg n)$

İkili Arama için sözdeler

1) İteratif:

İkili_Arama(A, X)

$$p=1$$

$$q=n$$

while $p \leq q$

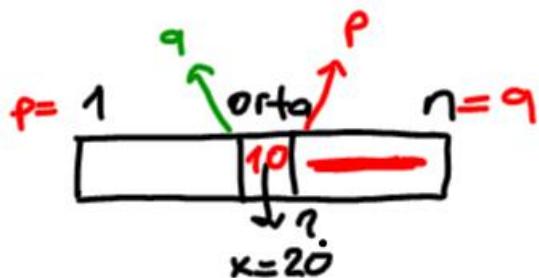
$$\text{orta} = \left\lfloor \frac{p+q}{2} \right\rfloor$$

if $A[\text{orta}] = X$
return true

if $A[\text{orta}] < X$
 $p = \text{orta} + 1$

else $q = \text{orta} - 1$

return false



2. Rekürsif (recursive)

İkili_Arama_Rec(A, X, p, q)

if $p \leq q$

$$\text{orta} = \left\lfloor \frac{p+q}{2} \right\rfloor$$

if $A[\text{orta}] = X$
return true

if $A[\text{orta}] < X$

ikili_Arama_Rec(A, X, orta+1, q)

else ikili_Arama_Rec(A, X, p, orta-1)

else return false

ilk Çağrım: ikili_Arama_Rec(A, X, 1, n)

Problem: Bir sayının kuvvetini hesaplama

x bir sayı ve n pozitif bir sayı isin x^n bul.

Saf Algoritma: $x^n = \underbrace{x \cdot x \cdots x}_{n \text{ defa}}$

$$T(n) = \Theta(n)$$

Böl ve Fethet Alg:

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & \text{eğer } n \text{ çift ise} \\ x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x & \text{eğer } n \text{ tek ise} \end{cases}$$

$$x^{16} = x^8 \cdot x^8$$

$$\downarrow$$

$$x^4 \cdot x^4$$

$$\downarrow$$

$$x^2 \cdot x^2$$

$$\downarrow$$

$$x \cdot x$$

Güçlendirme Zamanı

$$T(n) = 1 \cdot T(n/2) + \Theta(1)$$

$$T(n) = \Theta(\lg n)$$

UsAI(x, n)

if $n=1$

return x

Yanlış

if n çift ise

return **UsAI($x, n/2$) * UsAI($x, n/2$)**

if n tek ise

return **UsAI($x, (n-1)/2$) * UsAI($x, (n-1)/2$) * x**

UsAI(2, 7)



UsAI(2, 3) * UsAI(2, 3) * x



UsAI(2, 1) * UsAI(2, 1) * x

→

UsAI(2, 1) * UsAI(2, 1) * x

$$T(n) = 2 \cdot T(n/2) + \Theta(1)$$

$$1 \leftrightarrow n^{\log_2 2} = n^1$$

$$1 = O(n^{1-\varepsilon}) \quad \varepsilon = 1 \text{ için}$$

durum 1, $T(n) = \Theta(n)$

Doğrusu

UsAl(x, n)

if $n=1$

return x

if n çift ise

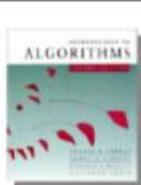
result = UsAl(x, n/2)

return result * result

if n tek ise

result = UsAl(x, (n-1)/2)

return result * result * x



Matrislerde çarpma

Girdi: $A = [a_{ij}]$, $B = [b_{ij}]$. **Cıktı:** $C = [c_{ij}] = A \cdot B$. $i, j = 1, 2, \dots, n$.

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj}$$



Standart algoritma

```

    for i ← 1 to n      (i 1'den n'ye kadar)
    do for j ← 1 to n  (j 1'den n'ye kadar)
        do cij ← 0
            for k ← 1 to n
                do cij ← cij + aik · bkj
    
```

$$T(n) = \Theta(n^3)$$

```

    for i=1 to n
    ...
    for j=1 to n
    ...
    j=j*2
    
```

$$T(n) = \Theta(n \lg n)$$

$$j=1 \ 2 \ 4 \ 8 \ 16 \ \dots \ n$$

```

    for i=1 to n
    ...
    for j=1 to √n
    ...
    for k=n to 1
    ...
    k=k/2
    
```

$$T(n) = \Theta(n^{3/2} \lg n)$$

```

    for i=1 to n/2
    ...
    for j=1 to n
    ...
    j=j+2
    for k=1 to n
    ...
    k=k+2
    
```

$$T(n) = \frac{n}{2} \cdot \frac{n}{2} + n$$

$$T(n) = \Theta(n^2)$$

Böl-ve-fethet algoritması

FIKİR

$n \times n$ matris = $(n/2) \times (n/2)$ altmatrisin 2×2 matrisi: 4 adet $n/2 \times n/2$ boyutlarında alt matrislere bölenir.

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}_{n \times n}$$

$$C = A \cdot B$$

$$\left. \begin{array}{l} r = ae + bg \\ s = af + bh \\ t = ce + dg \\ u = cf + dh \end{array} \right\}$$

8 tane matris çarpımı
 4 tane " toplamı

Galışma Zamanı

$$T(n) = 8T(n/2) + \Theta(n^2)$$

alt matrislerin toplama'sı

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \dots & b_{nn} \end{bmatrix} = \begin{bmatrix} \vdots & \dots & \vdots \\ \vdots & \dots & \vdots \\ n & n & \vdots \\ \hline & & n \end{bmatrix}$$

n adet top.
 n adet top.
 n
 \vdots
 n^2 adet toplama
gerekir.

Master Metodu ile çözelim

n^2 ile $n^{\log_2 8} = n^3$ karşılaştır.

$$n^2 = O(n^{3-\varepsilon}) \quad \varepsilon = 1 \text{ için}$$

Durum 1: $T(n) = \Theta(n^3)$

Standart algoritmadan daha iyi değil.

Strassen'in fikri

- 2×2 matrisleri yalnız 7 özyinelemeli çarpma ile çöz.

$$P_1 = a \cdot (f - h)$$

$$P_2 = (a + b) \cdot h$$

$$P_3 = (c + d) \cdot e$$

$$P_4 = d \cdot (g - e)$$

$$P_5 = (a + d) \cdot (e + h)$$

$$P_6 = (b - d) \cdot (g + h)$$

$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$

$$s = P_1 + P_2$$

$$t = P_3 + P_4$$

$$u = P_5 + P_1 - P_3 - P_7$$

$$r = ae + bg - dh + dg - de - ah - bh + bg + bh$$

$$r = ae + bg$$

7 tane çarpma

18 tane toplama/çıkarma

Galışma Zamanı

$$T(n) = 7T(n/2) + \Theta(n^2)$$

$$n^2 \text{ ile } n^{\log_2 7} \approx n^{2.81..} \text{ i karşılaştır.}$$

$$n^2 = O(n^{2.81-\varepsilon}) \quad \varepsilon = 0.81$$

$$\text{Durum 1: } T(n) = \Theta(n^{\log_2 7})$$

Sorular ve Çözümleri

1- Alg1 ($A[1..n]$)

```
if  $n \leq 0$ 
    return
 $q = \lfloor n/2 \rfloor$ 
if  $q$  is even
    Alg1( $A[1..q]$ )
else
    Alg1( $A[q+1..n]$ )
```

Alg2 ($A[1..n]$)

```
if  $n \leq 0$ 
    return
 $q = \lfloor n/4 \rfloor$ 
if  $q$  is even
    Alg2( $A[1..q]$ )
    Alg2( $A[q+1..2q]$ )
else
    Alg2( $A[2q+1..3q]$ )
    Alg2( $A[3q+1..n]$ )
```

Alg1 ve Alg2 nin çalısmaya zamanlarını bulunuz.
Hangisi daha iyidir?

Cözüm: Alg1 için $T(n) = 1 \cdot T(n/2) + \Theta(1)$
 $= \Theta(\lg n)$

Alg2 için $T(n) = 2 \cdot T(n/4) + \Theta(1)$

1 ile $n^{\log_4 2} = n^{1/2}$ karşılaştır.

$1 = O(n^{1/2-\varepsilon})$ $\varepsilon = 1/2$

durum 1, $T(n) = \Theta(\sqrt{n})$

Alg1 Alg2 den daha iyidir.

2. Tek Doruklu Arama

Bir $A[1..n]$ dizisi eğer aranır bir alt diziyi takip eden azalan bir alt diziyi sahip ise tek doruklu bir dizidir.

Yani $A[i] < A[i+1] \quad \forall 1 \leq i < m$

$A[j] > A[j+1] \quad \forall m \leq j < n$



Soru: Tek doruklu dizide maksimum elemanı $\Theta(\lg n)$ zamanda bulan bir algoritma geliştiriniz.

A: 1 2 3 4 5 6 | 7 8 9 10 11
2 3 7 8 11 9 | 8 5 4 1 -2
2 3 7 | 8 11 9

Tek_Doruklu_Arama ($A[1..n]$)

a=1

b=n

while a < b

$$\text{mid} = \left\lfloor \frac{a+b}{2} \right\rfloor$$

if $A[\text{mid}] > A[\text{mid}+1]$

b = mid

else

a = mid + 1

return A[a]

a=1 2 3 4 5 6 ^{mid} 7 8 9 10 11 = b
2 3 7 8 11 9 8 5 4 1 -2

a=1 mid b=6
2 3 7 8 11 9

a=4 mid b=6

8 11 9

a=4 b=5

8 11

mid

11

a=5

b=5

Girişim Zamanı: $T(n) = 1 \cdot T(n/2) + \Theta(1) = \Theta(\lg n)$

Ödev: Tek_Doruklu_Arama algoritmasını özinelemeli algoritmaya dönüştürünüz.

3. n elemanlı bir dizide tekrarlı elemen varsa True yok ise False döndüren verimli bir algoritma tasarlayın.

Tekrarlimi ($A[1 \dots n]$)

```
for i=1 to n  
    for j=i+1 to n  
        if A[i]=A[j]  
            return True
```

return False

$$T(n) = \Theta(n^2)$$

Tekrarlimi_v2 ($A[1 \dots n]$)

$\Theta(n \lg n) \leftarrow \text{Merge-Sort}(A[1 \dots n])$

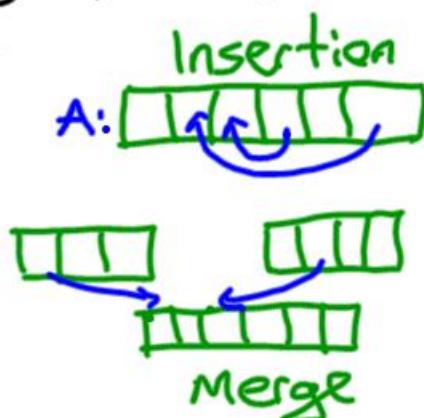
$\Theta(n) \leftarrow$

```
for i=1 to n-1  
    if A[i]=A[i+1]  
        return True  
return False
```

$$T(n) = \Theta(n \lg n) + \Theta(n) = \Theta(n \lg n)$$

Çabuk (Quick) Sıralama

- Hoare tarafından 1962 de geliştirilmiş
- Böl ve Fethet paradigması
- Yerinde sıralama yapar.
- Pratiktir.



Böl ve Fethet

- Böl:** Diziyi pivot x elemanının etrafında iki alt diziyi böl öylelikle
 pivot
 $\leq x \quad x \quad \geq x$
sadece değil
- Fethet:** iki alt diziyi özyinelemeli sırala
- Birleştir:** Gerek yok.

Anahtar işlen: Bölüntüleme (Partition)

Partition (A, p, q)

$\Theta(1)$ $x = A[p]$ //pivot

$\Theta(1)$ $i = p$

for $j = p+1$ to q

if $A[j] \leq x$ $\Theta(1)$

$i = i + 1$ $\Theta(1)$

$A[i] \leftrightarrow A[j]$ $\Theta(1)$

$\Theta(n)$

$\Theta(1)$ $A[p] \leftrightarrow A[i]$

$\Theta(1)$ return i

n eleman için
bölüntüleme
Galisma Zamanı:

$T(n) = \Theta(n)$

Örnek: Bölütüleme

$p=1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8=9$

A: $\boxed{6 \quad 10 \quad 13 \quad 5 \quad 8 \quad 3 \quad 2 \quad 11}$

$x=6 \quad i \quad j \rightarrow$

$\boxed{6 \quad 10 \quad 13 \quad 5 \quad 8 \quad 3 \quad 2 \quad 11}$

$i \quad j \rightarrow$

$\boxed{6 \quad 10 \quad 13 \quad 5 \quad 8 \quad 3 \quad 2 \quad 11}$

$i \quad j$

$\boxed{6 \quad 5 \quad 13 \quad 10 \quad 8 \quad 3 \quad 2 \quad 11}$

$i \quad j$

$\boxed{6 \quad 5 \quad 13 \quad 10 \quad 8 \quad 3 \quad 2 \quad 11}$

$i \quad j \rightarrow$

$\boxed{6 \quad 5 \quad 3 \quad 10 \quad 8 \quad 13 \quad 2 \quad 11}$

$i \quad j$

$\boxed{6 \quad 5 \quad 3 \quad 10 \quad 8 \quad 13 \quad 2 \quad 11}$

$i \rightarrow \quad i \quad j$

$\boxed{6 \quad 5 \quad 3 \quad 2 \quad 8 \quad 13 \quad 10 \quad 11}$

$i \quad j \rightarrow$

$\boxed{6 \quad 5 \quad 3 \quad 2 \quad 8 \quad 13 \quad 10 \quad 11}$

$i \quad j$

Partition (A, p, q)
 $x = A[p] \quad // \text{pivot}$
 $i = p$
for $j=p+1$ to q
if $A[j] \leq x$
 $i = i+1$
 $A[:i] \leftrightarrow A[j]$
 $A[p] \leftrightarrow A[i]$
return i

for döngüsü
bitti

$\boxed{2 \quad 5 \quad 3 \quad 6 \quad 8 \quad 13 \quad 10 \quad 11}$

$i \quad j$

≤ 6

> 6

$p=1$ 2 3 4 5 6 7 = 9
10 8 3 20 4 6 14

$x=10$

$\begin{array}{ccccccc} & \text{J} \\ & | \\ 10 & 8 & 3 & 20 & 4 & 6 & 14 \end{array}$

$\begin{array}{ccccccc} & \text{J} \\ & | \\ 10 & 8 & 3 & 20 & 4 & 6 & 14 \\ & & i & \text{J} \end{array}$

$\begin{array}{ccccccc} 10 & 8 & 3 & 4 & 20 & 6 & 14 \\ & & & ; & \text{J} \end{array}$

$\begin{array}{ccccccc} \underline{10} & 8 & 3 & 4 & \underline{\frac{6}{;}} & 20 & \underline{14} \\ & & & & & & \text{J} \end{array}$

$\begin{array}{cccccc} 6 & 8 & 3 & 4 & \boxed{10} & 20 \ 14 \\ \underbrace{\quad}_{<10} & & & & \downarrow & \underbrace{\quad}_{>10} \\ & & & & & \vdots \end{array}$

$\begin{array}{ccccc} \vdots & & & & \vdots \\ 3 & 4 & 6 & 8 & 14 \ 20 \end{array}$

Gabuk Sıralama Sözde Kodu

Quicksort(A, p, q)

if $\rho < q$

$r = \text{Partition}(A, P, q)$

Quicksort(A, p, r-1)

Quicksort(A, i, j)

Quicke'sort (1973)

Açılıd refak sıralama özgürne

bölütüleme işlemlerinden başka birsey dēgil
Gabuk Sıralama Analizi

ayaların farklı olduğunu

En kötü durum analizi

Yukarıda sıralı veya ters sıralı

1 A 6 7 10 12 15

$$\Rightarrow \exists (a) : T(a-1)$$

$$T(n) = T(n-1) + \Theta(1)$$

$T(n) = T(n-1) + O(1)$

İeme oğaci ile çözümü

$$T(n) = T(n-1) + cn$$

100% - 1

$$T(n) = c \sum_{i=1}^n T(i-1)$$

$$\begin{array}{c}
 c(n) \\
 | \\
 c(n-2) \\
 | \\
 \vdots \\
 \Theta(1) \\
 + \\
 \hline
 c(n + (n-1) + (n-2) + \dots + 1)
 \end{array}$$

Aritmatik seri

C. 5

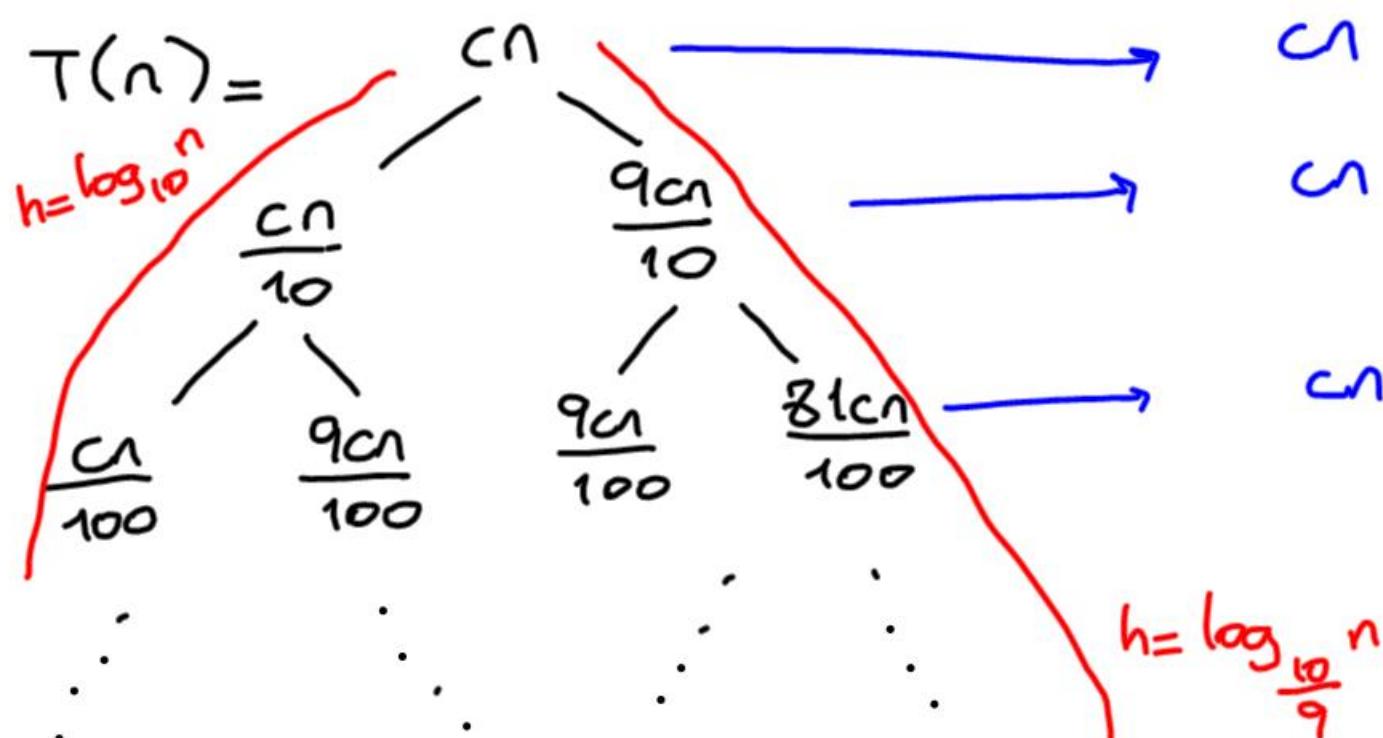
En iyi durum analizi

Eğer sonsuzsa, partition dizisi iki eşit parçaya böler.

$$T(n) = 2T(n/2) + \Theta(n)$$
$$= \Theta(n \lg n)$$

Soru: Bölümüleme $\frac{1}{10} : \frac{9}{10}$ oronunda olsaydı

$$T(n) = T(n/10) + T(9n/10) + \Theta(n)$$



$\Theta(1)$

$\Theta(1)$

$\Theta(1)$

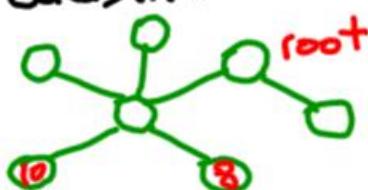
+

$$cn \log_{10} n < T(n) < cn \log_{\frac{10}{9}} n$$

$$T(n) = \Theta(n \lg n)$$

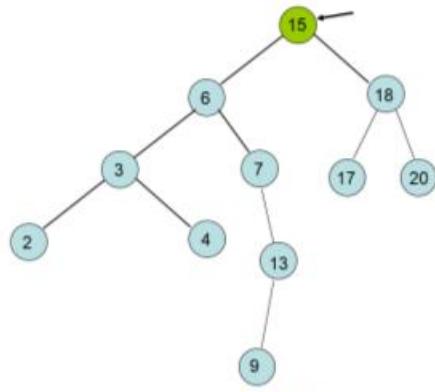
İkili Arama Ağacıları (BST)

Ağac: Döngü içermeyen bir graf. tler iki düğüm arasında kesinlikle bir yol olan graf.

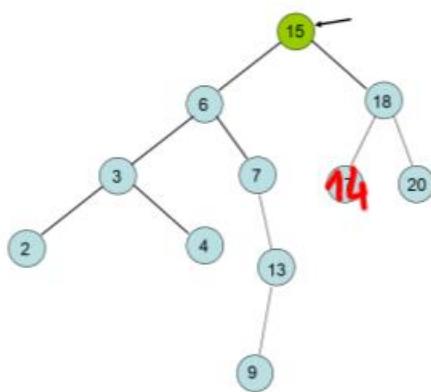


BST: x ikili arama ağacında bir düğüm olsun
Eğer y düğümü x'in sol alt ağacında ise
 $\text{key}[y] \leq \text{key}[x]$

Eğer y düğümü x'in sağ alt ağacında ise
 $\text{key}[x] \leq \text{key}[y]$



BST



BST değil

Ağaclarda İşlemler

BST de dolaşma

Inorder Tree walk: left **root** right

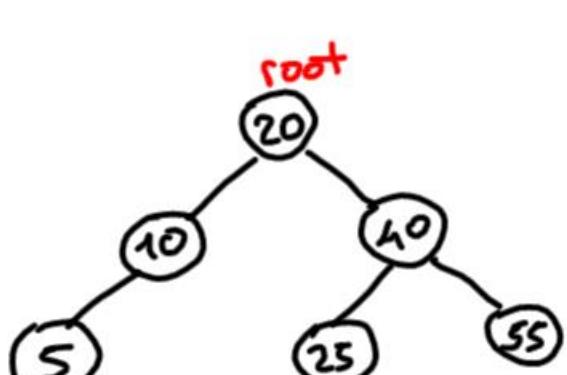
Preorder Tree walk: **root** left right

Postorder Tree walk: left right **root**

Inorder-Tree-walk(x)

if $x \neq \text{NIL} // \text{NULL}$

preorder } Inorder-Tree-walk(left[x])
postorder } print key[x]
} Inorder-Tree-walk(right[x])



inorder: 5 10 20 25 40 55

preorder: 20 10 5 40 25 55

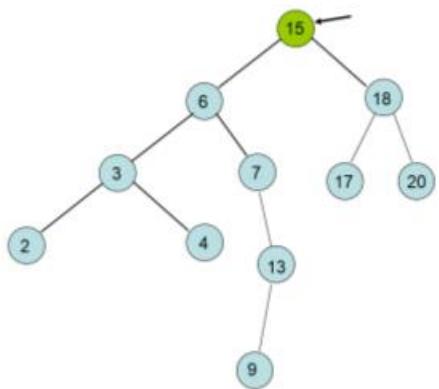
postorder: 5 10 25 55 40 20

left, right, root

$$T(n) = T(n-d-1) + T(d) + \Theta(1)$$

$$T(n) = \Theta(n)$$

Verilen bir değeri Arama



Tree-Search(x, k)

if $x = \text{NIL}$ or $\text{key}[x] = k$
return x

if $k < \text{key}[x]$

return Tree-Search(left[x], k)

else

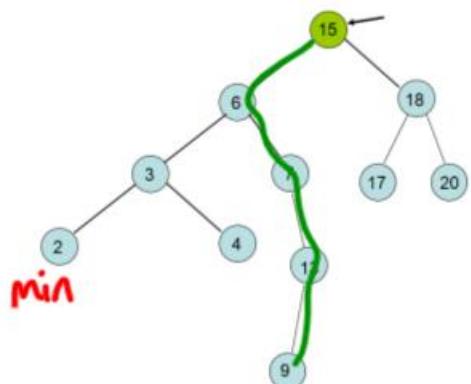
return Tree-Search(right[x], k)

$$T(n) = O(h)$$

üst sınır

h : ağacın yüksekliği

Minimumu Bulma



Tree-Min(x)

if $\text{left}[x] \neq \text{NIL}$

Tree-Min(left[x])

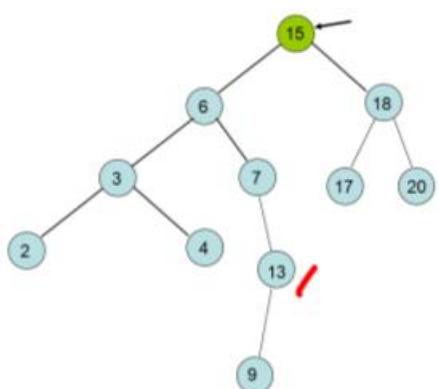
return key[x]

$$T(n) = O(h)$$

HW: Tree-Search alg. suni iteratif yazın

Successor (Ardılı) Bulma

Bir x düğümünün ardılı, key degeri $\text{key}[x]$ den büyük en küçük ^{olan} düğümdür.



13'in ardılı: 15, 13 den büyük olanların en küçükü

6 nin ardılı: 7
15 in ardılı: 17
4'in ardılı: 6

Tree-Successor(x)

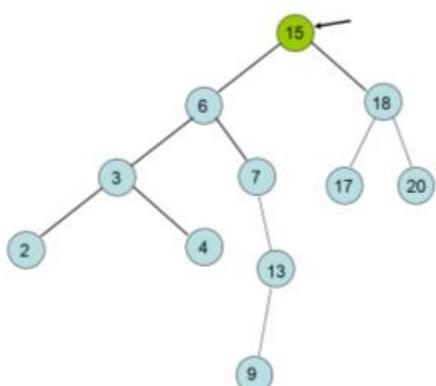
```
if right[x] ≠ NIL Θ(1)  
    return Tree-Min(right[x]) O(h)
```

$y = p[x]$ Θ(1)
while $y \neq \text{NIL}$ and $x = \text{right}[y]$
 $\quad \left\{ \begin{array}{l} O(h) \\ x = y \\ y = p[y] \end{array} \right.$
return y

$$T(n) = O(h)$$

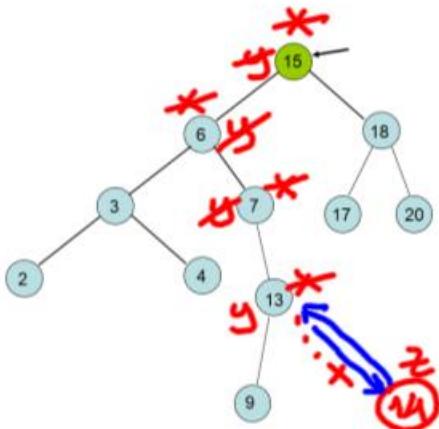
Predecessor (Öncülü) Bulma

Bir x düğümünün öncülü x in key degerinden küçük en büyük düğümdür.



17 nin öncülü: 15
13'in öncülü: 9
6 nn " : 4
15 in " : 13

Dügüm Ekleme



TREE-INSERT(T, z)

```

1   $y \leftarrow \text{NILL}$ 
2   $x \leftarrow \text{root}[T]$ 
3  while  $x \neq \text{NILL}$ 
4      do  $y \leftarrow x$ 
5          if  $\text{key}[z] < \text{key}[x]$ 
6              then  $x \leftarrow \text{left}[x]$ 
7          else  $x \leftarrow \text{right}[x]$ 
8   $p[z] \leftarrow y$ 
9  if  $y = \text{NILL}$ 
10     then  $\text{root}[T] \leftarrow z$        $\nabla$  Tree  $T$  was empty
11     else if  $\text{key}[z] < \text{key}[y]$ 
12         then  $\text{left}[y] \leftarrow z$ 
13         else  $\text{right}[y] \leftarrow z$ 

```

$\tau(n) = O(h)$

Dinamik Programlama (DP)

Genel, güslü bir alg. tasarım teknigidir.
DP: alt problem + tekrar kullanma

Örnek: Fibonacci Sayiları

$$1, 1, 2, 3, 5, 8, 13, \dots$$
$$\text{fib}(1) \quad \text{fib}(2) \quad F_n = F_{n-1} + F_{n-2}$$

Saf Özginelemeli Alg.

$\text{fib}(n)$

if $n \leq 2$

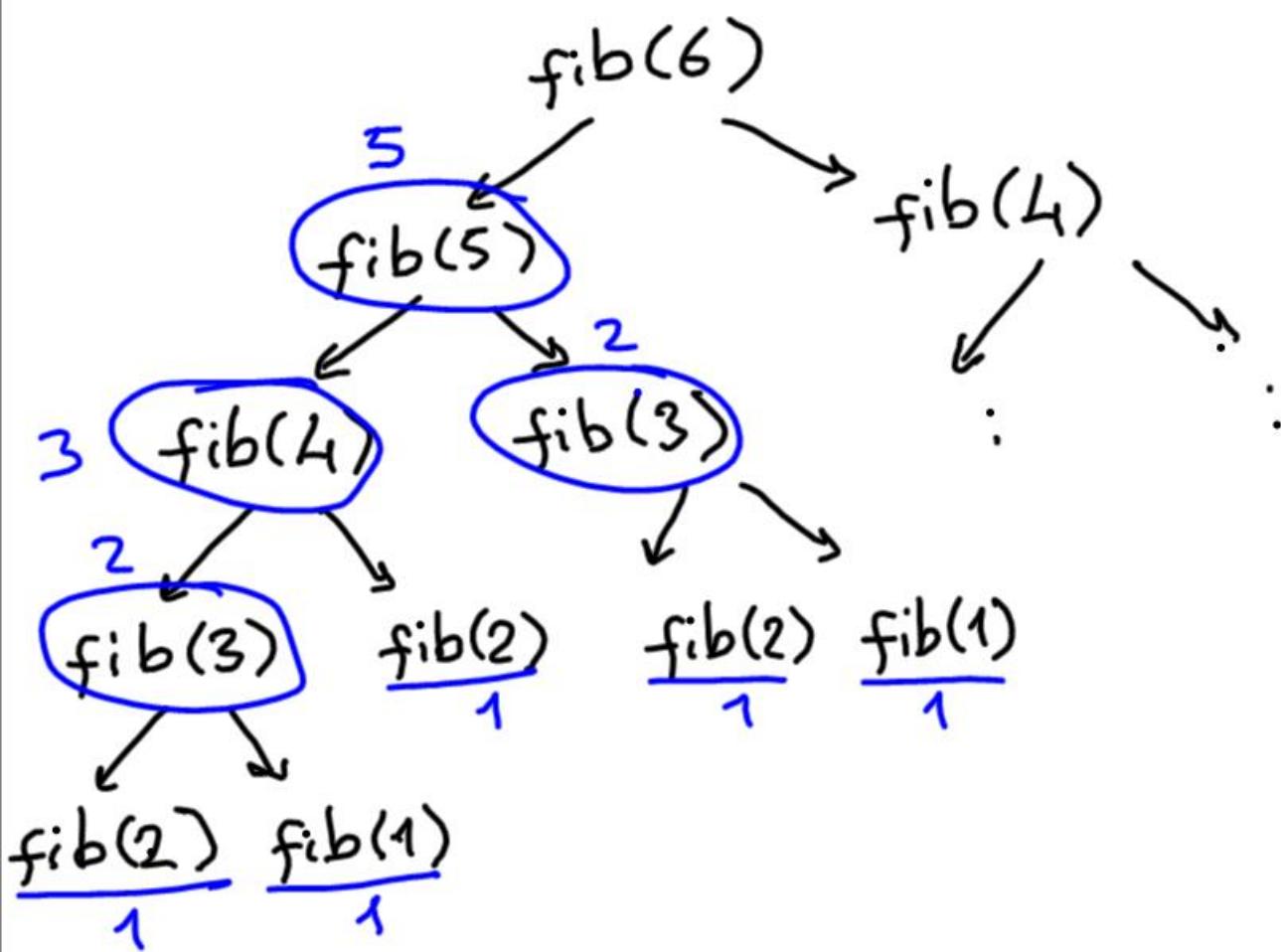
$f=1$

else

$f = \text{fib}(n-1) + \text{fib}(n-2)$

return f

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) + \Theta(1) \\ &= \Theta(2^n) \quad \text{üstel, kötü} \end{aligned}$$



Hafızalı (Memoized) DP algoritması

memo = {}

fib(n)

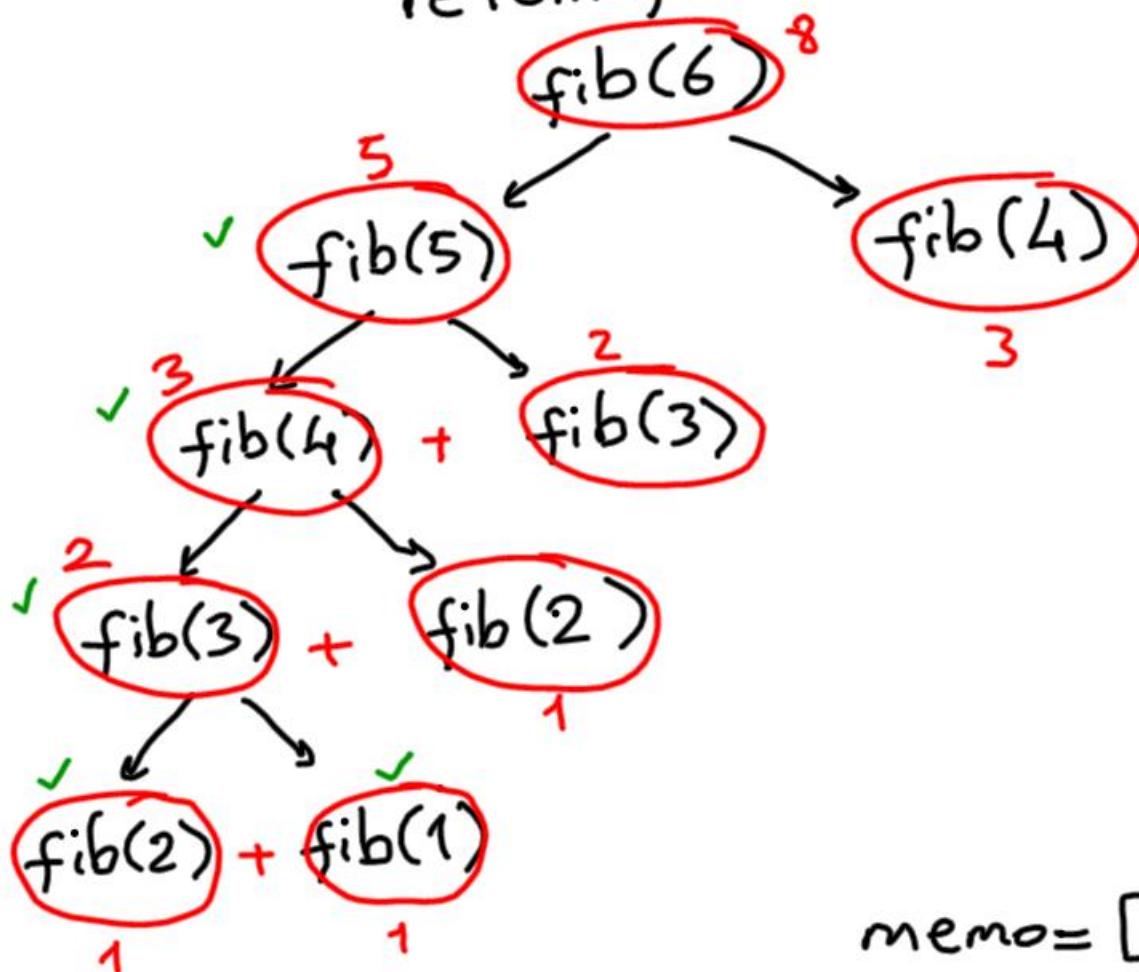
if $n \in \text{memo}$
return memo[n]

if $n \leq 2$
 $f = 1$

else
 $f = \text{fib}(n-1) + \text{fib}(n-2)$

memo[n] = f

return f



memo = [1 1 2 3 5 8]

Hafızalı (Memoized) DP algoritması Analizi

her k için, $\text{fib}(k)$ ilk çağırıldığında özyinelemeli çalışır.

Hafızadaki çözümlere (değerlere) erişim zamanı $\Theta(1)$.

Hafızada olmayan çağrımlar n adettir.

$\text{fib}(1), \text{fib}(2), \dots, \text{fib}(n)$

$$T(n) = n \cdot \Theta(1) = \Theta(n)$$

Dinamik Programlama Detayı

- Böl ve Fethet gibi bir alg. tasarım teknigidir.
- B&F de alt problemler bağımsız olmalı.
- AH problemler bağımlı ise DP uygulanır.
- DP her alt problemi bir defa çözer ve sonucu bir tabloda saklar
- DP genellikle optimizasyon problemlerine uygulanır.

DP geliştirmek için şu dört adım izlenir.

1. Optimal çözümün yapısının karakteristiği ortaya çıkarılmalı.
2. Özyinelemeli olarak çözümün değerini tanımlamalı
3. Altta-üste (bottom-up) mantığı ile bir optimal çözümün değerini hesaplamalı
4. Hesaplanan bilgilerden optimal çözüm elde edilir.

$$\min. f(x) = x^2 + 2$$

$$f(0) = 2$$

optimal çözüm

optimal değer

Matris Zinciri Çarpımı

$\langle A_1, A_2, \dots, A_n \rangle$ n adet matrisden oluşan bir matris zinciri olsun.

$A_1 \cdot A_2 \cdot A_3 \cdot \dots \cdot A_n$ çarpımı isteniyor.

Matris çarpımının birleşme özelliği var ve her parantezleme sayısı sonucu versin.

Örn: $\langle A_1, A_2, A_3, A_4 \rangle$

$$\begin{array}{ll} ((A_1 \cdot A_2) \cdot (A_3 \cdot A_4)) & (A_1 \cdot ((A_2 \cdot A_3) \cdot A_4)) \\ (((A_1 \cdot A_2) \cdot A_3) \cdot A_4) & ((A_1 \cdot (A_2 \cdot A_3)) \cdot A_4) \\ (A_1 \cdot (A_2 \cdot (A_3 \cdot A_4))) & \end{array}$$

Fakat her birinin farklı maliyetleri var.

iki matrisin çarpım maliyeti: Toplam skalar çarpım sayısı:

$A_{m \times n} \cdot B_{n \times k}$ için toplam $m \cdot n \cdot k$ skalar çarpma gerekir.

$$\left[\begin{matrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{matrix} \right]_{m \times n} \cdot \left[\begin{matrix} b_{11} & \dots & b_{1k} \\ b_{21} & \dots & b_{2k} \\ \vdots & & \\ b_{n1} & \dots & b_{nk} \end{matrix} \right]_{n \times k} = \left[\begin{matrix} n & n & \dots & n \\ n & n & \dots & n \\ \vdots & & & \\ n & \dots & n \end{matrix} \right]_{m \times k} \begin{array}{l} \xrightarrow{n \cdot k \text{ çarpma}} \\ \xrightarrow{n \cdot k} \\ \xrightarrow{n \cdot k} \\ + \\ \hline m \cdot n \cdot k \end{array}$$

Örnek: $\langle A_1, A_2, A_3 \rangle$ matris zinciri olsun.

Boyutları sırası ile $10 \times 100, 100 \times 5, 5 \times 50$ olsun.

Amaç $A_1 A_2 A_3$ çarpımını bulmak

1. alternatif: $(A_1 \cdot A_2) A_3 =$

$A_1 \cdot A_2$ için $10 \cdot 100 \cdot 5 = 5000$ adet skalar çarpması

$B \cdot A_3$ $10 \times 5 \quad 5 \times 50$ $10 \cdot 5 \cdot 50 = 2500$ adet " "

+

Toplam 7500 adet çarpması var

2. alternatif: $A_1 \cdot (A_2 \cdot A_3)$

$A_2 \cdot A_3$ için $100 \cdot 5 \cdot 50 = 25000$ adet çarpması

$A_1 \cdot C$ $10 \times 100 \quad 100 \times 50$ $10 \cdot 100 \cdot 50 = 50000$ adet çarpması

+

Toplam 75000 adet çarpması var

n elemanlı matris zinciri için $O(2^n)$ parantezleme vardır. Yani üstel (exponential), o zaman brute-force çalışmaz.

Problem: n adet matristen oluşan bir $\langle A_1, A_2, \dots, A_n \rangle$ zinciri verilsin. A_i matrisinin boyutu $p_{i-1} \times p_i$ olsun. $A_1 \cdot A_2 \cdot \dots \cdot A_n$ çarpımı için hangi parentezleme optimaldir yani minimum adet çarpması gerektir?

DP geliştirmek için şu dört adım izlenir.

1. Optimal çözümün yapısının karakteristiği ortaya çıkarılmalı.
2. Özinelemeli olarak çözümün değerini tanımlamalı
3. Altta-üste (bottom-up) mantığı ile bir optimal çözümün değerini hesaplamalı
4. Hesaplanan bilgilerden optimal çözüm elde edilir.

DP ile Matris Zinciri Probleminin Çözümü

1. Adım: Optimal parantezlemenin yapısı.

$$A_{i \dots j} = A_i A_{i+1} \dots A_j , \quad i \leq j$$

$A_i \dots A_j$ çarpımı için şu şekilde bir parantezleme olur.

$$(A_i A_{i+1} A_{i+2} \dots A_k) (A_{k+1} \dots A_j)$$

$1 \leq i \leq k < j \leq n$ olmak üzere

2. Adım. Özinelemeli bir çözüm

$m[i, j]$ değeri $A_{i \dots j}$ çarpımı hesaplamak için gereken minimum skalar çarpım sayısını olsun.

$$m[i, j] = \begin{cases} 0 & i=j \text{ ise} \\ \min \left\{ m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \right\} \text{ aksi halde} \\ & i \leq k < j \end{cases}$$

A_i matrisinin boyutu $p_{i-1} \times p_i$ dir.

3. Adım: $m[i, j]$ hesaplamak için saf bir recursive (özünelebilir) alg. yazabilirim. Fakat bunun çalışma zamanı üsteldir. Optimal maliyeti bottom-up şeklinde hesaplayacağız.

MATRIX-CHAIN-ORDER(p)

```

1  n = p.length - 1
2  let m[1..n, 1..n] and s[1..n - 1, 2..n] be new tables
3  for i = 1 to n
4    m[i, i] = 0
5  for l = 2 to n           // l is the chain length
6    for i = 1 to n - l + 1
7      j = i + l - 1
8      m[i, j] = ∞
9      for k = i to j - 1
10        q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
11        if q < m[i, j]
12          m[i, j] = q
13          s[i, j] = k
14  return m and s

```

$$p = [p_0, \underbrace{p_1, p_2, \dots, p_{n-1}}_{A_1}, p_n] \quad A_2 \quad A_n$$

$$p = [30, 35, 15, 5, 10, 20, 25]$$

	matrix	A_1	A_2	A_3	A_4	A_5	A_6
dim.		30×35	35×15	15×5	5×10	10×20	20×25

$p_0, p_1, p_2, p_3, p_4, p_5, p_6$

m	1	2	3	4	5	6
1	0	15125	7875			15125
2	0	2625	1375	7125		
3	0		750	2500		
4	0			1000	3500	9125
5	0				5000	l=2
6	0					0

for $i=1$ to n do
 for $j=i+1$ to n do
 for $k=j+1$ to n do
 $q = m[i, k] + m[k, j] + p_{i-1} p_k p_j$
 $q = 0 + 0 + 30 \cdot 35 \cdot 15 = 15125$
 $m[1, 2] = 15125$

for $i=2$ to n do
 for $j=i+1$ to n do
 for $k=j+1$ to n do
 $q = m[2, k] + m[k, j] + p_{1-1} p_k p_j$
 $q = 0 + 0 + 35 \cdot 15 \cdot 5 = 2625$

i=3 için devam

$l=3$ için $i=1$ $j=3$ $m[1, 3] = \infty$
 $k=1$ den 2 ye kadar

$A_1 \cdot (A_2 A_3)$

$$m[1, 3] = \min_{1 \leq k \leq 3} \left\{ \begin{array}{l} m[1, 1] + m[2, 3] + p_0 p_1 p_3 = 7875 \rightarrow \min \\ m[1, 2] + m[2, 3] + p_0 p_2 p_3 = 17375 \end{array} \right.$$

$(A_1 A_2) A_3$

$A_2 A_3 A_4 A_5$

$$m[2, 5] = \min_{2 \leq k \leq 5} \left\{ \begin{array}{l} m[2, 2] + m[3, 5] + p_1 p_2 p_5 = 13000 \\ m[2, 3] + m[3, 5] + p_1 p_3 p_5 = 7125 \rightarrow \min \\ m[2, 4] + m[3, 5] + p_1 p_4 p_5 = 11375 \end{array} \right.$$

4. Adım Optimal parentezlenmesi bulmak:
matris zincirinin nereden bölüneceği
S tablosunda saklanıyor.

S	2	3	4	5	6
1	1	1	3	3	3
2		2	3	3	3
3			3	3	3
4				4	5
5					5

optimal çözüm: $(A_1(A_2 A_3) \backslash (A_4 A_5) A_6)$

optimal değer: 15125

Açgözlü Algoritmalar

Graf: (Hatırlatma)

$$G = (V, E)$$

V - teplerin kumesi

E - aysitlerin kumesi $E \subseteq V \times V$

Yönlü Graf

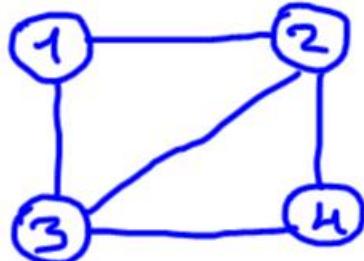
Aysitler sıralı tepe çiftleridir.

$$E = \{(v_1, v_2), (v_3, v_2), (v_2, v_1), \dots\}$$

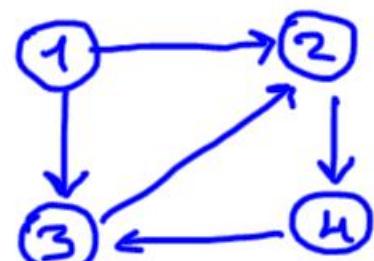
Yönsüz Graf

Aysitler sıralı olmayan tepe çiftleridir.

$$e = (v_1, v_2) = (v_2, v_1)$$



Yönsüz



Yönlü

$$|E| = O(|V|^2)$$

Eğer graf bağlantılı ise $|E| \geq |V|-1$

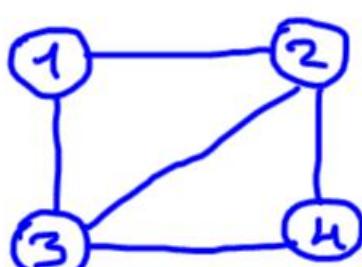
yani $|E| = \Omega(|V|)$

Grafların Tensili

1. Komsuluk matrisi: Bir $G = (V, E)$ grafinin komsuluk matrisi gösterimi $A_{n \times n}$ matrisidir.

$$n = |V|, V = \{1, 2, \dots, n\}$$

$$A[i, j] = \begin{cases} 1 & \text{eğer } (i, j) \in E \\ 0 & \text{eğer } (i, j) \notin E \end{cases}$$



$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 0 & 1 & 1 & 0 \end{bmatrix}_{4 \times 4}$$

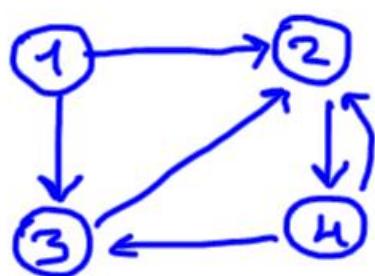
simetrik

Komsuluk matrisi bellekte $\Theta(V^2)$ abn gerektir.

Yoğun graflarda tercih edilir.

2. Komsuluk listesi gösterimi

Bir $v \in V$ tepeşinin v komsuluk listesi
 v ye komşu obrn teplerin listesidir.



$$\begin{aligned}Adj[1] &= \{2, 3\} \\Adj[2] &= \{4\} \\Adj[3] &= \{2\} \\Adj[4] &= \{2, 3\}\end{aligned}$$

$$\begin{aligned}1 &\rightarrow 2 \rightarrow 3 \rightarrow \text{NULL} \\2 &\rightarrow 4 \rightarrow \text{NULL} \\3 &\rightarrow 2 \rightarrow \text{NULL} \\4 &\rightarrow 2 \rightarrow 3 \rightarrow \text{NULL}\end{aligned}$$

Komsuluk listesi bellekte $\Theta(V+E)$ obrn
gerekir.

Seyrek grafalar için uygundur.

Derece

Yönsüz bir grafda bir düğümün derecesi:
düğüme bağıc sayıdır.

Yönlü grafda giren derece, giden derece

Tokalaşma Lemma

(Handshaking Lemma)

Yönsüz bir grafda

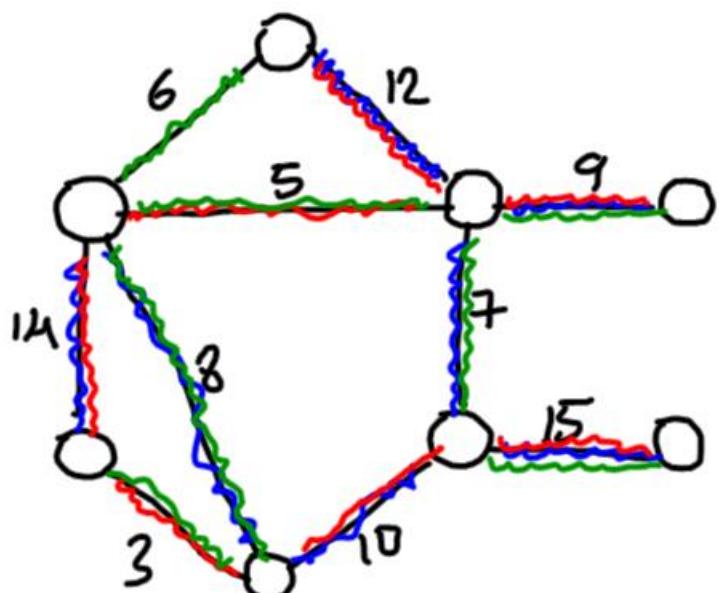
$$\sum_{v \in V} \text{derece}(v) = 2 \cdot |E|$$

Minimum Kapsayan Ağacı (MST) Problemi

Girdi: $w: E \rightarrow \mathbb{R}$ ağırlıklı, bağıntılı bir
 $G = (V, E)$ grafi

Cıktı: Tüm tepeleri içeren en az ağırlıklı
bir T ağacı

$$w(T) = \sum_{e=(u,v) \in T} w(e) \rightarrow \min \text{ olacak}$$

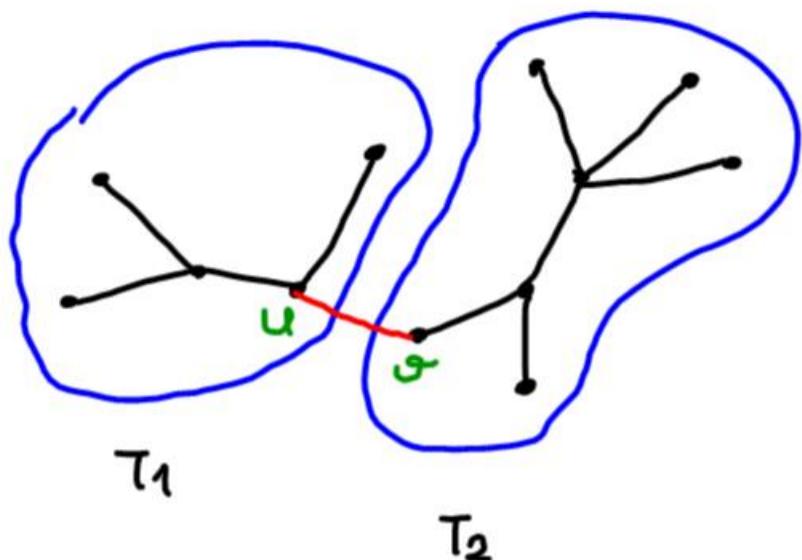


$$T_1: w(T_1) = 12 + 9 + 7 + 15 + 10 + 8 + 14 = 75$$

$$T_2: w(T_2) = 12 + 5 + 9 + 14 + 3 + 10 + 15 = 68$$

$$T_3: w(T_3) = 6 + 5 + 8 + 7 + 9 + 15 + 3 = 53 \rightarrow \min$$

Optimal Mapı

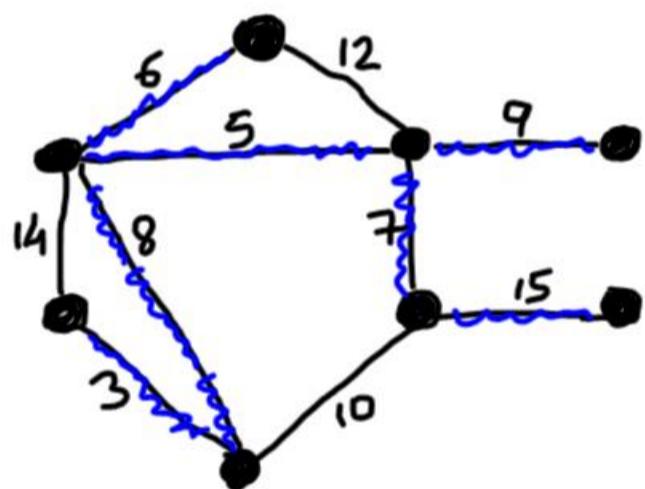


Açgözlü Algoritma seçim özellikleri

Yerel olarak en uygun seçim genel olarak en uygundur.

Teorem: T bir $G = (V, E)$ grafının MKA'sı olsun ve $A \subseteq V$ olsun. (u, v) aysıtı A yi $V - A$ ya bağlayan min ağırlıklı aysıt olsun. \square zaman $(u, v) \in T$ dir.

Prim Algoritması



● $\in A$
○ $\in V - A$

$$w(T) = 6 + 5 + 9 + 7 + 15 + 8 + 3 \\ = 53$$

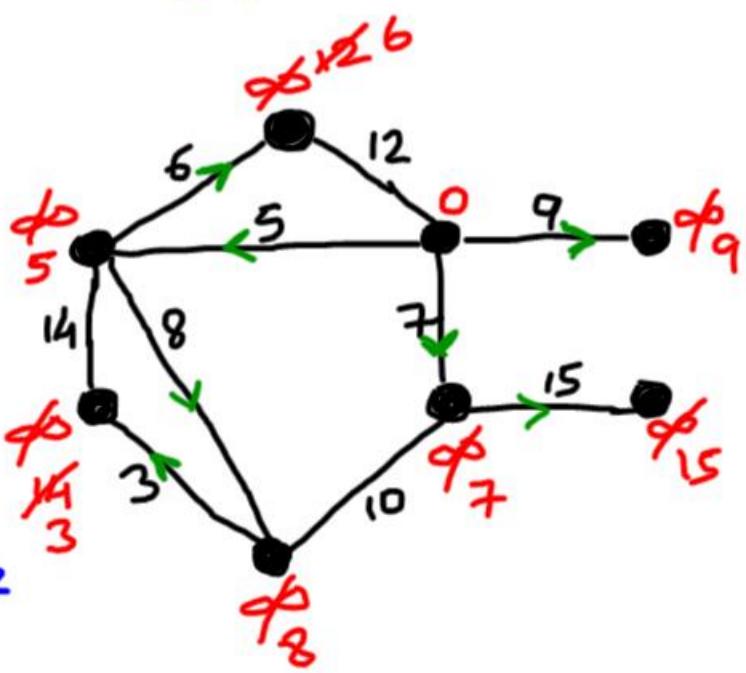
MST-PRIM(G, w, r)

```

1 for each  $u \in G.V$ 
2    $u.key = \infty$ 
3    $u.\pi = \text{NIL}$ 
4    $r.key = 0$ 
5    $Q = G.V$ 
6 while  $Q \neq \emptyset$ 
7    $u = \text{EXTRACT-MIN}(Q)$ 
8   for each  $v \in G.Adj[u]$ 
9     if  $v \in Q$  and  $w(u, v) < v.key$ 
10       $v.\pi = u$ 
11       $v.key = w(u, v)$  } gevretme
  
```

min(key) sırası olası

gevretme



Prim Algoritmasının Analizi

MST-PRIM(G, w, r)

```

 $\Theta(V)$  {  

    1 for each  $u \in G.V$   

    2  $u.key = \infty$        $\Theta(1)$   

    3  $u.\pi = \text{NIL}$        $\Theta(1)$   

    4  $r.key = 0$   

    5  $Q = G.V$   

    6 while  $Q \neq \emptyset$   

    7      $u = \text{EXTRACT-MIN}(Q)$  ?  

    8     for each  $v \in G.Adj[u]$   

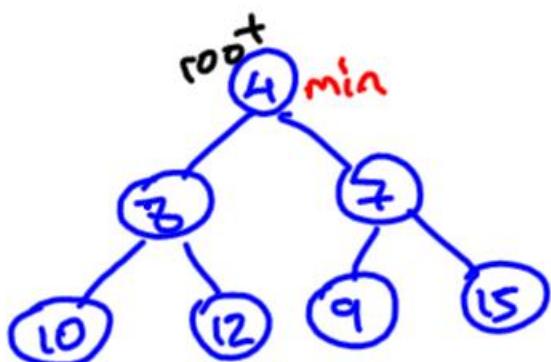
    9       if  $v \in Q$  and  $w(u, v) < v.key$   

    10           $v.\pi = u$   

    11           $v.key = w(u, v)$  } decrease-key
  }
}
  
```

$$T = \Theta(V) \cdot T_{\text{extract_min}} + \Theta(E) \cdot T_{\text{decrease_key}}$$

Q	$T_{\text{extract_min}}$	$T_{\text{decrease_key}}$	T
Array	$\Theta(V)$	$\Theta(1)$	$\Theta(V^2) + \Theta(E) = \Theta(V^2)$
Binary heap	$\Theta(\lg V)$	$\Theta(\lg V)$	$\Theta(V \lg V) + \Theta(E \lg V) = \Theta(E \lg V)$



DFS (Depth First Search) Derinlik Öncelikli Arama

DFS(G, s)

mark(s);

$L = \{s\}$;

while $L \neq \emptyset$

$u = \text{last}(L)$; there exists

if $\exists (u, v)$ such that v is unmarked

choose (u, v) with v of smallest index;

mark(v);

$L = L \cup \{v\}$; // push(L, v)

else

$L = L \setminus \{u\}$; // pop(L)

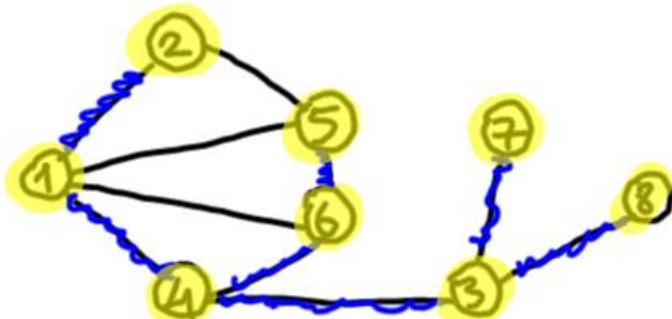
DFS de L isin stack (Last in First out) kılınır.

L (stack)

2	
2, 1	
2, 1, 4	
2, 1, 4, 3	
2, 1, 4, 3, 7	
2, 1, 4, 3	
2, 1, 4, 3, 8	
2, 1, 4, 3	
2, 1, 4	
2, 1, 4, 6	
2, 1, 4, 6, 5	
2, 1, 4, 6	
2, 1, 4	
2, 1	
2	
\emptyset	

Marked

2	
1	
4	
3	
7	
8	
-	
-	
-	
6	
5	
-	
-	
-	
-	
-	
-	

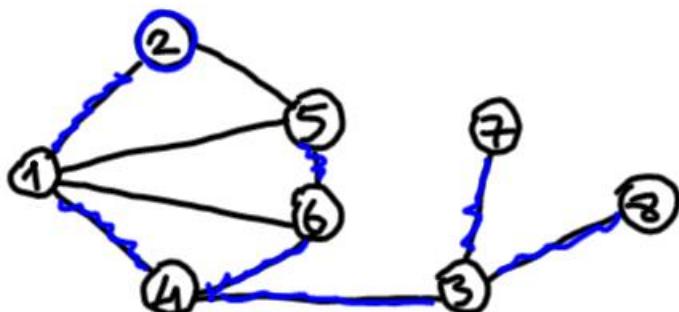


```

while L ≠ ∅
  u = last(L);
  if ∃ (u, v) such that v is unmarked
    choose (u, v) with v of smallest index
    mark(v);
    L = L ∪ {v}; // push(L, v)
  else
    L = L \ {u}; // pop(L)
  
```

Galgma Zanari:

$$T(|V|, |E|) = \Theta(E + V)$$



DFS

BFS (Breadth First Search) Genişlik Öncelikli Arama

BFS (G, s)

mark(s);

$L = \{s\};$ // L kuyruk.

while $L \neq \emptyset$

$u = \text{first}(L);$

if $\exists (u, v)$ such that v is unmarked

choose (u, v) with v of smallest index;

mark(v);

$L = L \cup \{v\};$ // enqueue(L, v)

else

$L = L \setminus \{u\};$ // dequeue(L)

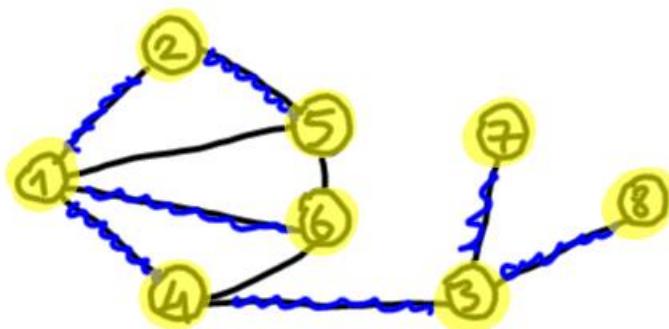
L igin kuyruk veri yapisi (First in first out)
kullanılır.

L (bagruk)

Marked

2	2
2, 1	1
2, 1, 5	5
1, 5	-
1, 5, 4	4
1, 5, 4, 6	6
5, 4, 6	-
4, 6	-
4, 6, 3	3
6, 3	3
3	-
3, 7	7
3, 7, 8	8
7, 8	-
8	-
\emptyset	-

BFS

while $L \neq \emptyset$

u = first(L);

if $\exists (u, v)$ such that v is unmarked
choose (u, v) with v of smallest index;

mark(v);

 $L = L \cup \{v\}$; // enqueue(L, v)

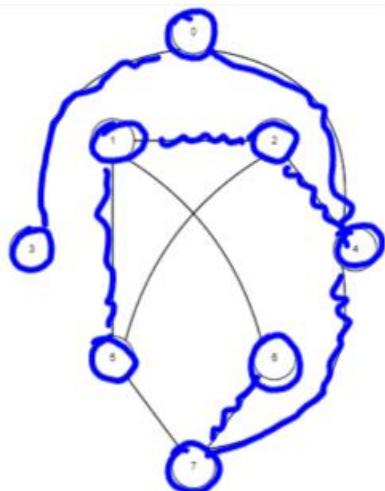
else

 $L = L \setminus \{u\}$; // dequeue(L)

Galısmalar:

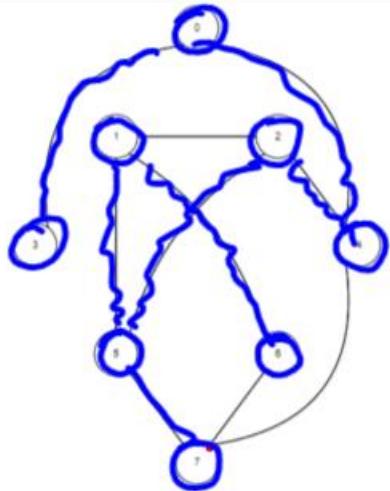
$$T(|V|, |E|) = \Theta(E + V)$$

DFS



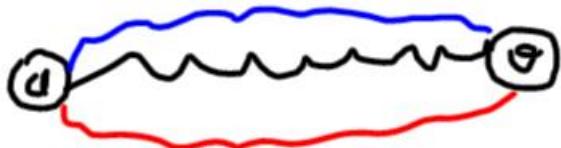
$s=5$

BFS



$s=5$

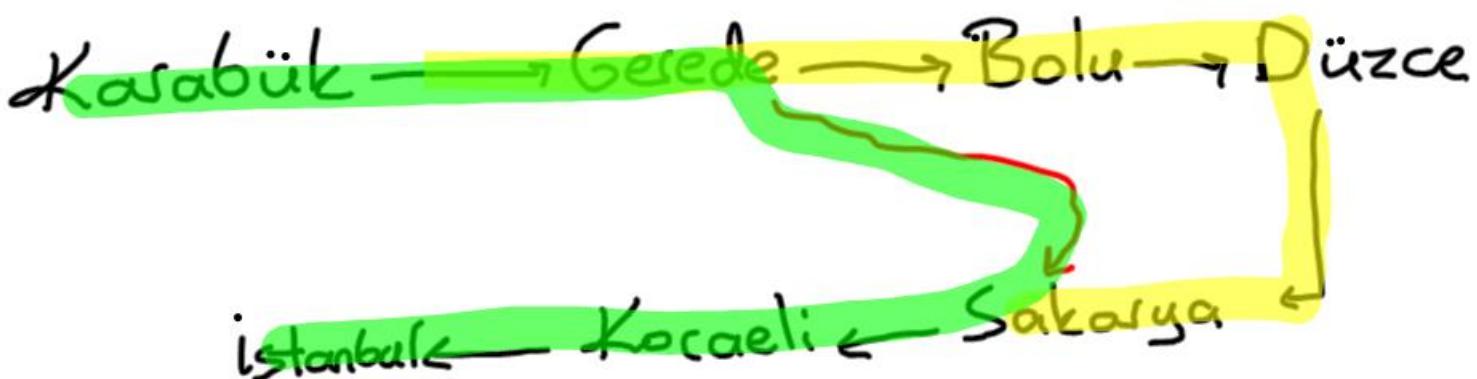
En kısa yollar



$\delta(u, v)$, en kısa yolu
ağırlığı

Optimal Altyapı

Teorem: En kısa yolu bir alt yolu en kısa bir yoldur.



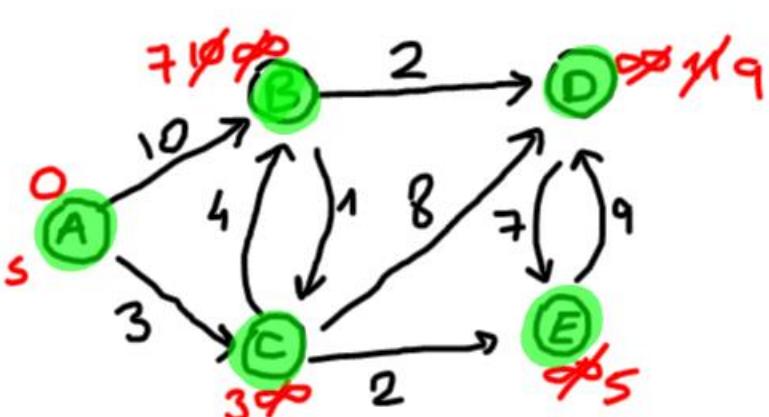
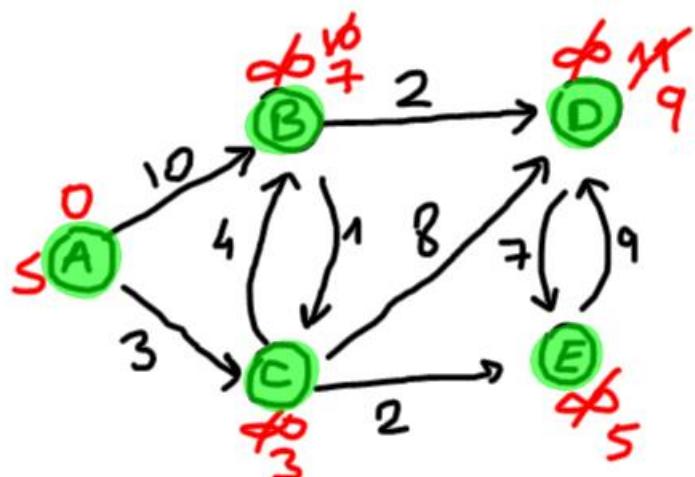
Dijkstra Algoritması

Bu alg. pozitif ağırlıklı veya negatif ağırlıklı çevrime sahip olmayan graflerde çalışır. Bir kaynaktan diğer tüm telpelere varsa en kısa yolları bulur.

Dijkstra(G, s)

```

 $d[s] \leftarrow 0$ ;  $\text{pred}[s] = \text{NULL}$ 
for each  $v \in V - \{s\}$ 
    do  $d[v] \leftarrow \infty$ 
s  $\text{pred}[v] = \text{NULL}$ 
 $Q \leftarrow V$   $\triangleright Q$  is a priority queue maintaining  $V$ 
while  $Q \neq \emptyset$ 
    do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
        for each  $v \in \text{Adj}[u]$ 
            do if  $d[v] > d[u] + w(u, v)$  relaxation step
                then  $d[v] \leftarrow d[u] + w(u, v)$ 
 $\text{pred}[v] = u$ 
```



$\text{pred} = \text{NULL}$ C A B C
 $Q =$

A	B	C	D	E
0	7	3	9	5

 $d =$

0	7	3	9	5
---	---	---	---	---

$A \rightarrow C \rightarrow B \rightarrow D$

$A \rightarrow C \rightarrow E$

Dijkstra(G, s)

```

 $d[s] \leftarrow 0$ ;  $\text{pred}[s] = \text{NULL}$ 
for each  $v \in V - \{s\}$ 
    do  $d[v] \leftarrow \infty$ 
s  $\text{pred}[v] = \text{NULL}$ 
 $Q \leftarrow V$   $\triangleright Q$  is a priority queue maintaining  $V$ 
while  $Q \neq \emptyset$ 
    do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
        for each  $v \in \text{Adj}[u]$ 
            do if  $d[v] > d[u] + w(u, v)$  relaxation step
                then  $d[v] \leftarrow d[u] + w(u, v)$ 
 $\text{pred}[v] = u$ 
```

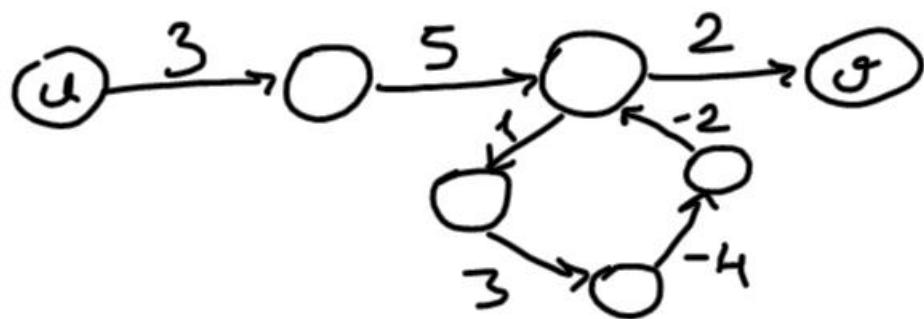
Dijkstra Algoritmasının Analizi

$\Theta(V)$ {
 $\Theta(1)$ $d[s] \leftarrow 0$
 $\Theta(V)$ { for each $v \in V - \{s\}$
 do $d[v] \leftarrow \infty$
~~Start~~
 $\Theta(V)$ $Q \leftarrow V$ $\triangleright Q$ is a priority queue maintaining $V - s$
 while $Q \neq \emptyset$
 do $u \leftarrow \text{EXTRACT-MIN}(Q)$?
~~Start~~
 for each $v \in \text{Adj}[u]$
 { do if $d[v] > d[u] + w(u, v)$ then $d[v] \leftarrow d[u] + w(u, v)$ relaxation step ? decrease key
~~dequeue(u)~~

$$T(E, V) = \Theta(V) \cdot T_{\text{extract-min}} + \Theta(E) \cdot T_{\text{decrease-key}}$$

Q	$T_{\text{extract-min}}$	$T_{\text{decrease-key}}$	$T(E, V)$
dizi	$\Theta(V)$	$\Theta(1)$	$\Theta(V^2)$
ikili heap	$\Theta(\lg V)$	$O(\lg V)$	$\Theta(E \lg V)$

Negatif Ağırlıklı Gevrim



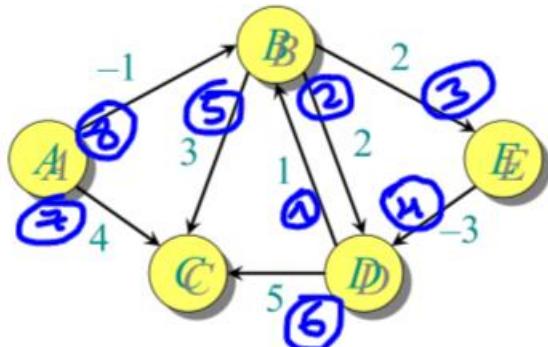
Bellman-Ford Alg.

Bir koungaktan diğer tüm tēpelere en kisa yolları bulus veya grafga negatif ağırlıklı bir gevrim varsa tespit eder.

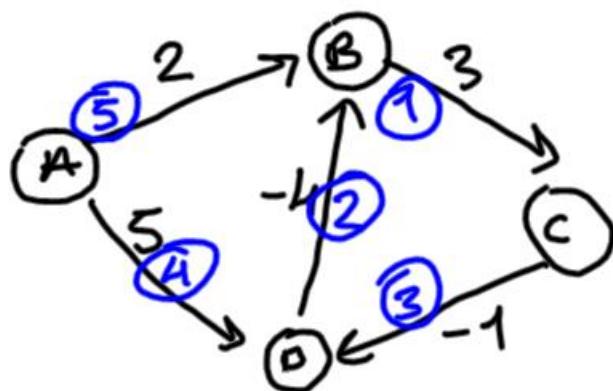
```
d[s] ← 0  
for each  $v \in V - \{s\}$  } initialization  
do  $d[v] \leftarrow \infty$ 
```

```
for  $i \leftarrow 1$  to  $|V| - 1$   
do for each edge  $(u, v) \in E$   
do if  $d[v] > d[u] + w(u, v)$   
then  $d[v] \leftarrow d[u] + w(u, v)$  } relaxation step
```

```
for each edge  $(u, v) \in E$   
do if  $d[v] > d[u] + w(u, v)$   
then "report that a negative weight cycle exists."
```



Örnek:



round 1:

round 2:

round 3:

Girişim Zamanı Analizi

```
d[s] ← 0  
for each  $v \in V - \{s\}$  do  $d[v] \leftarrow \infty$  } initialization  
  
for  $i \leftarrow 1$  to  $|V| - 1$   
do for each edge  $(u, v) \in E$   
do if  $d[v] > d[u] + w(u, v)$  then  $d[v] \leftarrow d[u] + w(u, v)$  } relaxation step  
for each edge  $(u, v) \in E$   
do if  $d[v] > d[u] + w(u, v)$  then "report ..."
```

$T(E, V) =$

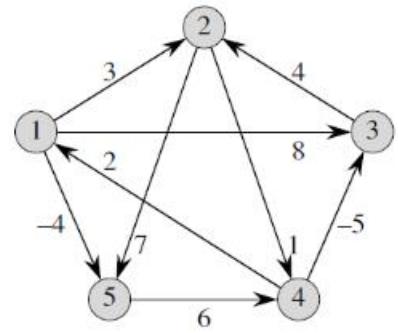
Q3. (a) What is the output of the following function for the call New(5,1). [8P]

```
function New(n,acc)
    if n == 1
        return 1*acc;
    return New(n-1,acc*n);
```

(b) Convert the recursive function to an iterative function. [10P]

FLOYD-WARSHALL(W)

1. $n \leftarrow \text{rows}[W]$
2. $D^{(0)} \leftarrow W$
3. **for** $k \leftarrow 1$ **to** n
4. **do for** $i \leftarrow 1$ **to** n
5. **do for** $j \leftarrow 1$ **to** n
6. $d_{ij}^{(k)} \leftarrow \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
7. **return** $D^{(n)}$



$W = (w_{ij})$

$$w_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \infty & \text{if } i \neq j \text{ and } (i, j) \in E, \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E. \end{cases}$$

$$W = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \end{matrix}$$

$$D^{(0)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \rightarrow D^{(3)} \rightarrow D^{(4)} \rightarrow D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Galışma Zamanı

$T(n=|V|) = \Theta(n^3)$

Sorular

1. Verilen bir stringte A ile başlayan B ile biten tüm alt stringleri sayan

CABAAABACAC
 | |
 A B

- a) Bir kaba-kuvvet (brute-force) alg. yazın.

AB_say(S, n) // S[1...n]

sayac = 0

for i=1 to n-1

if S[i] = 'A'

 for j=i+1 to n

 if A[j] = 'B'

 sayac++

return sayac

$$T(n) = O(n^2)$$

- b) $\Theta(n)$ çalışma zamanına sahip daha verimli bir alg. geliştirin.

AB_say2(S, n)

sayac = 0

sayac_A = 0

for i=1 to n

if S[i] = 'A'

 sayac_A++

if S[i] = **B**

 sayac += sayac_A

i=3 i=5 i=7 return sayac

↑ ↑ ↑

i=2 i=4 i=6 i=8

sayac = $\emptyset \neq 4$

sayac_A = $\emptyset \neq 2 \neq 5$

$$T(n) = \Theta(n)$$

5. Inorder fonksiyonu ile aynı işlevi yapan
yani bir ikili arama ağacında elementleri küçükten
büyükçe dolaşan alternatif bir alg. yazın.

İpucu: BST (iAA) fonksiyonlarını kullanabilirsiniz.

Gözüm: n elemanlı bir BST olduğunu kabul edelim.

Inorder2(x)

$O(h)$ $y = \text{Tree_Min}(x)$

$\Theta(1)$ print key[y]

for i=1 to n-1

$y = \text{Tree_Successor}(y)$ $O(h)$

print key[y] $\Theta(1)$

$\Theta(n)$

$$T(n) = O(h) + \Theta(n)O(h)$$

$$= O(n \cdot h)$$

\downarrow
 $\lg n$ ile n arasında

$$O(n \lg n) \leq T(n) \leq O(n^2)$$

