

6 ödev - 3 LAB - 1 Proje - 1 Final - 2 Vize
 Ç%14 o/o 6 Ç%140 Ç%140

İş algoritma → Zaman ve hafızayı işi kullanır, adapte edilebilir, Basittir, Doğrudur, Optimum, düzük bayan! 1.1.1

Zaman ölçme

```
begin = clock()
*/
/*
end = clock()
```

$$\text{timeSpent} = (\text{double}) (\text{end} - \text{begin}) / \text{CLOCK_PER_SEC}$$

} Empirical analiz

Sistem Langıllı etkiler → Yazılım, donanım, işletim sistemi, internet vs.

“ Başımısız ” → Algoritma → Teorik analiz

2. Hälfte

switch vs if -> az elem and a dacha h.zl.

↓
↳ element always lookup table structures are highly slugger
↳ hashlc

Analysis of asymptotic analysis older of growth $n \rightarrow \infty$

Basic operation \rightarrow En karmasik sistem

düssel Negelede
401 Farbenweg

for (*i*=0; *i*<*n*; *i*++) {
 if (*a*[*i*] == 0)
 (*out*++);
}

i assignment 1
 less than compare *n*+1 lines
 if (*a*[*i*] == 0) → equal ← compare *n* //
 a access ? *i*++
 }
 increment *min* ↑ max 2
 Explanation *b* *n*+2 *s* *n*+2
 min *max*

$$3\gamma^2 + 3\gamma + 1$$

Asymptotic Notations

O(): Big- \mathcal{O}

$\theta()$: Big theta
↓
average

$\int L(\cdot) \cdot \text{Big } \Omega_{\text{new}}$

① Big-Oh notation

$$f(n) = \Theta(n+3) \rightarrow O(n)$$

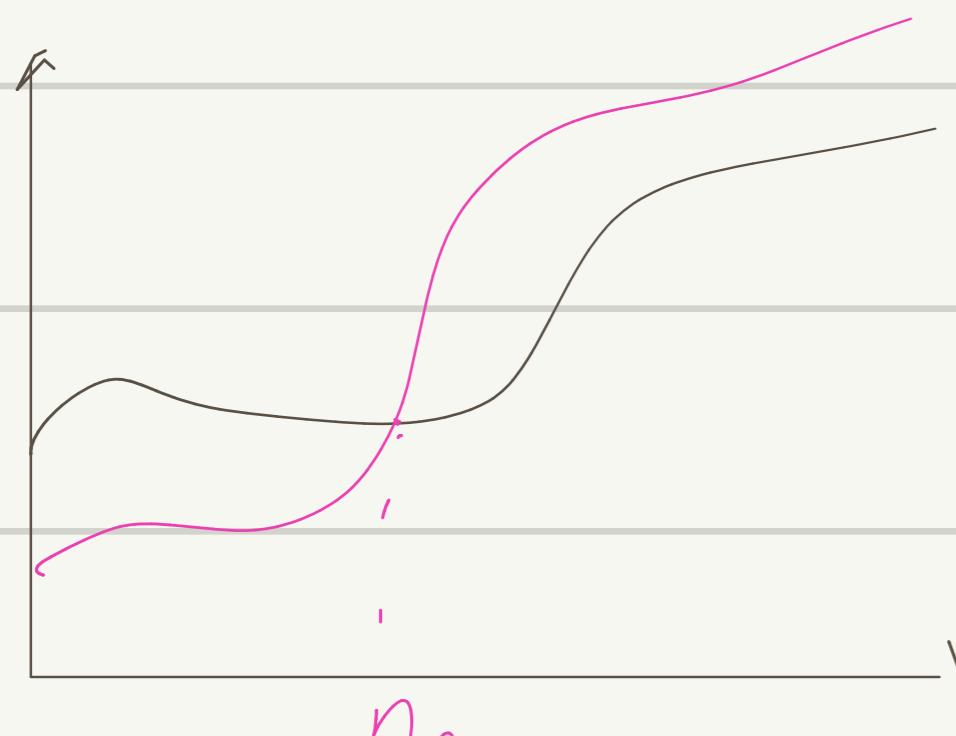
$f(n) \in O(g(n))$ $f(n)$ Lyle yazılabilir ama ne zaman

if $0 \leq f(\gamma) \leq c \neq g(\gamma)$

for all $c \geq 0$, $\eta \geq \gamma > 0$

for all $c > 0$, $\eta \geq \eta_0 > 0$

on the growth of a function $f(\eta)$



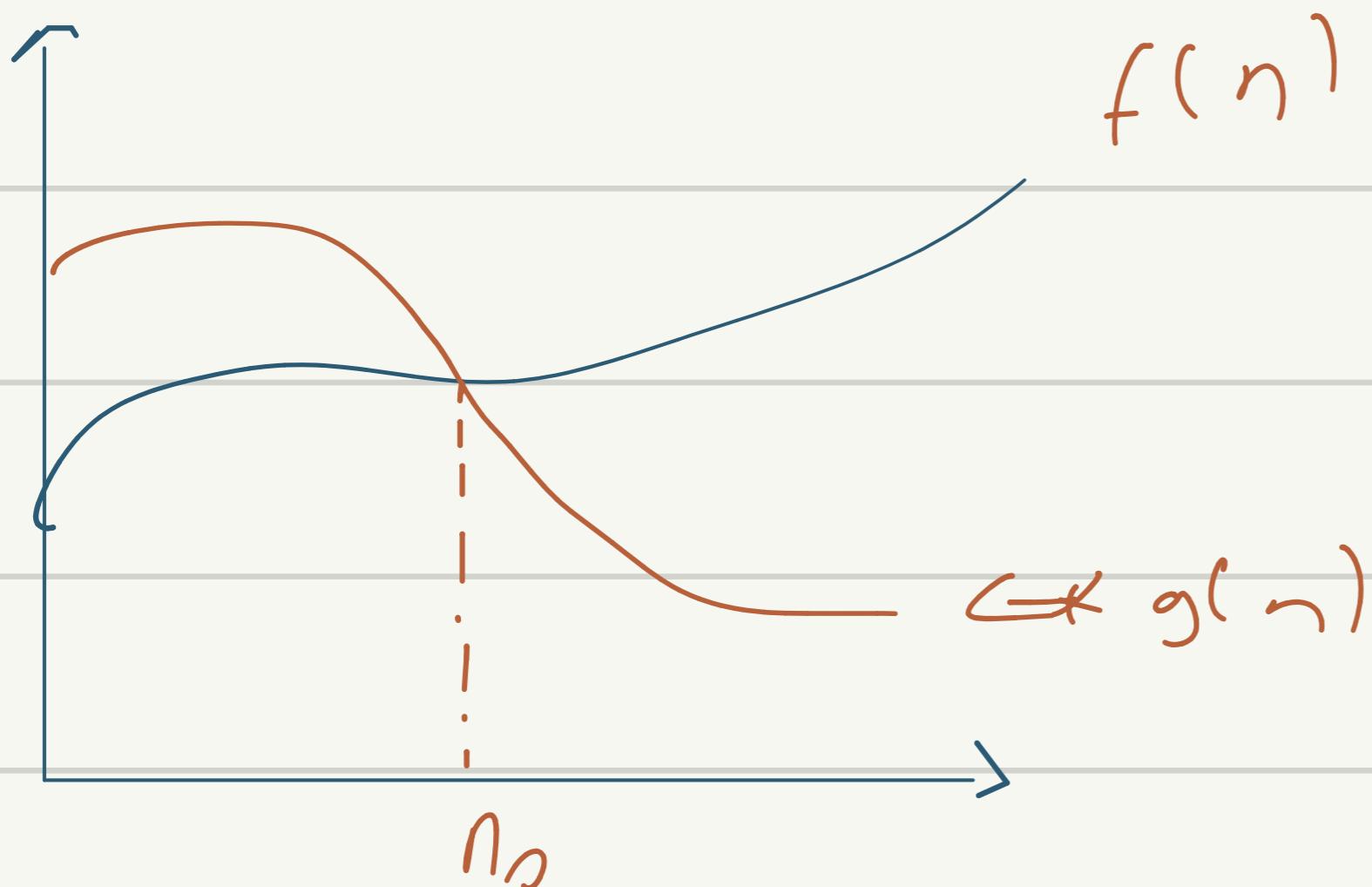
$$0 < 10\gamma + 3 \leq c \gamma g(\gamma) \rightarrow \alpha r$$

$30 < (-.10) \wedge n$

not $\gamma^2, \gamma^3, \gamma^4$ γ^5 her selfs other

② Big Omega notation lower bound

If $f(n) \geq c * g(n) \geq$ for all $c > 0$, $n \geq n_0 > 0$
 then $f(n) \in \Omega(g(n))$



$$5n^2 \in \Omega(n)$$

$$5n^2 \geq n \quad c \geq 0$$

$$5 \geq c \geq 0$$

~~if~~ $4n^2 \in \Omega(n^2)$

$$4n^2 \geq c * n^2$$

$$\begin{array}{l} c=3 \\ n_0=1 \end{array}$$

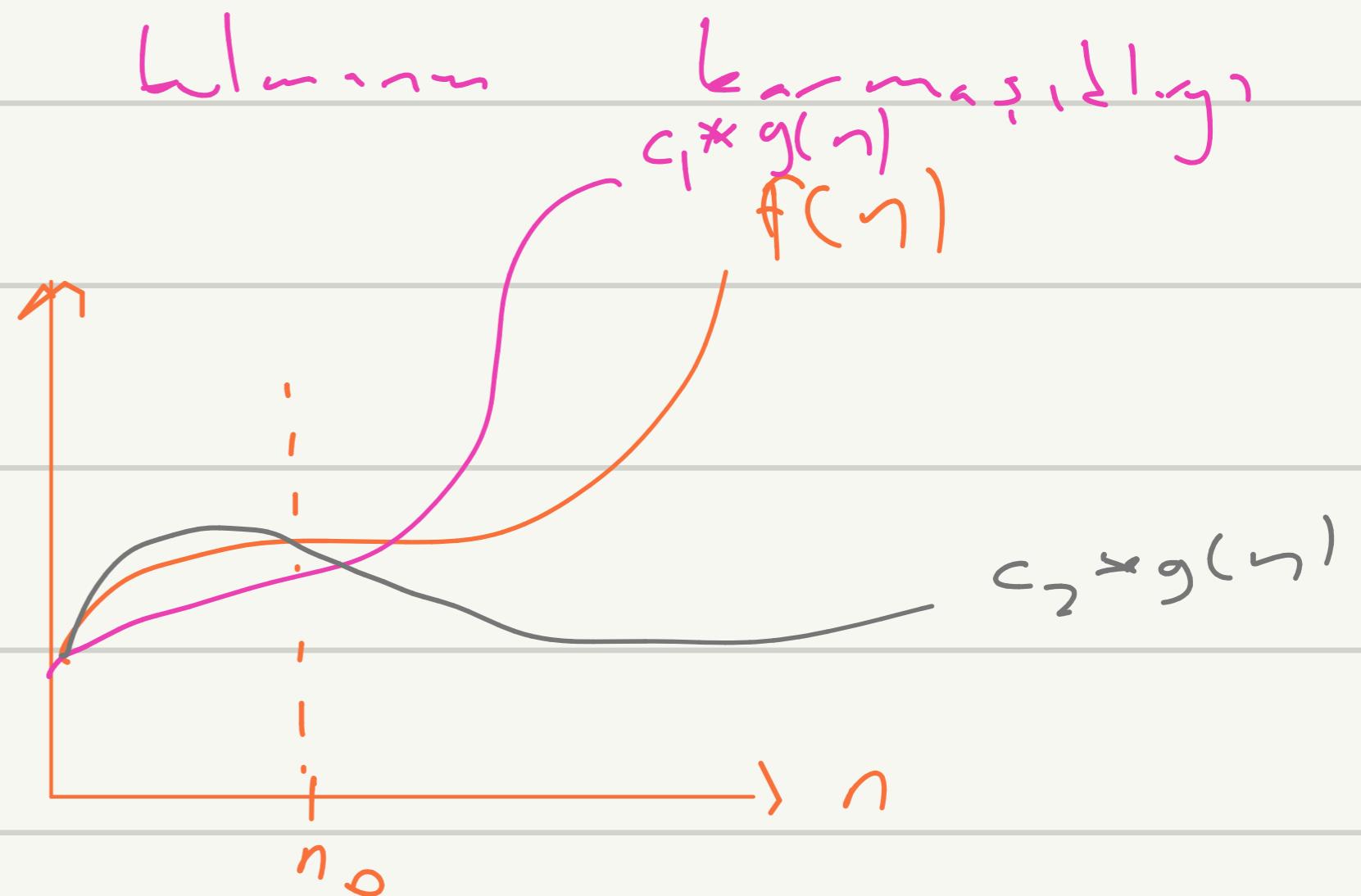
n_0 en c $\Omega(\max)$
 2. orunca degr: 1

③ Big Θ tight bound

Bir dizinin en büyük elemanı b_{\max} karamasıllığı
 worst $\rightarrow N$ best $\rightarrow N$

If $c_2 * g(n) \leq f(n) \leq c_1 * g(n)$
 for all $c_2 \leq c_1$, $c_1 > 0$, $c_2 > 0$

then $f(n) \in \Theta(g(n))$



$f(n)$: is asymptotically equal to $g(n) \rightarrow f(n) \in \Theta(g(n))$

$f(n) \in \Omega(g(n))$

~~if~~ $f(n) = 2n + 5 \rightarrow \Theta(n)$

$$n_0=1 \quad c_1=7 \quad 2n+5 \leq c_1 \cdot g(n) \rightarrow \Theta(n)$$

$$n_0=1 \quad c_1=1 \quad 2n+5 \geq c_1 \cdot g(n) \rightarrow \Omega(n)$$

$$\sqrt{7} > 1$$

3. hafta

① Rekürsif olmayan algoritmaların analizi:

int findMax(int n, int *A) { (1) Algoritmanın basisi Σ operasyonunu belile.

int i, max;

(2) Σ, Θ, O 'yu n için belile.

max = A[0];

```
for(i=1; i < n, i++) {
    if(max < A[i])
        max = A[i];
}
return max;
```

basis operasyon

karsılaştırma sayısı = $\sum_{i=1}^{n-1} 1 = n-1 \rightarrow n = \Sigma$

$k(n) = n$

$k(n) = n$

$k(n) = n$

average

$O(n)$

$\Sigma(n) = O(n)$

$\Theta(n)$

enkapsüle elementlerin ortulanması

② Sequential - Lineer search

int linearSearch(int n, int *A, int x) {

int i = 0;

```
while(i < n && A[i] != x)
    i++;
if(i < n)
    return i;
return -1;
```

basis operasyon: Σ

$C_{best}^n = 1$

$C_{worst}^n = n$

$C_{avg} \rightarrow p: 0 < p \leq 1$ eleman var i^{th} position $\frac{p}{n}$

$$C_{avg}(n) = \left[1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \dots + n \cdot \frac{p}{n} \right] + n \cdot (1-p)$$

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

$$P=0 \Rightarrow \frac{0 \cdot (n+1)}{2} + n \cdot (1-0) \Leftrightarrow n$$

$$\hookrightarrow C_{avg}(n) = n \rightarrow \Theta(n) \rightarrow \Sigma(1) \rightarrow \Theta(n)$$

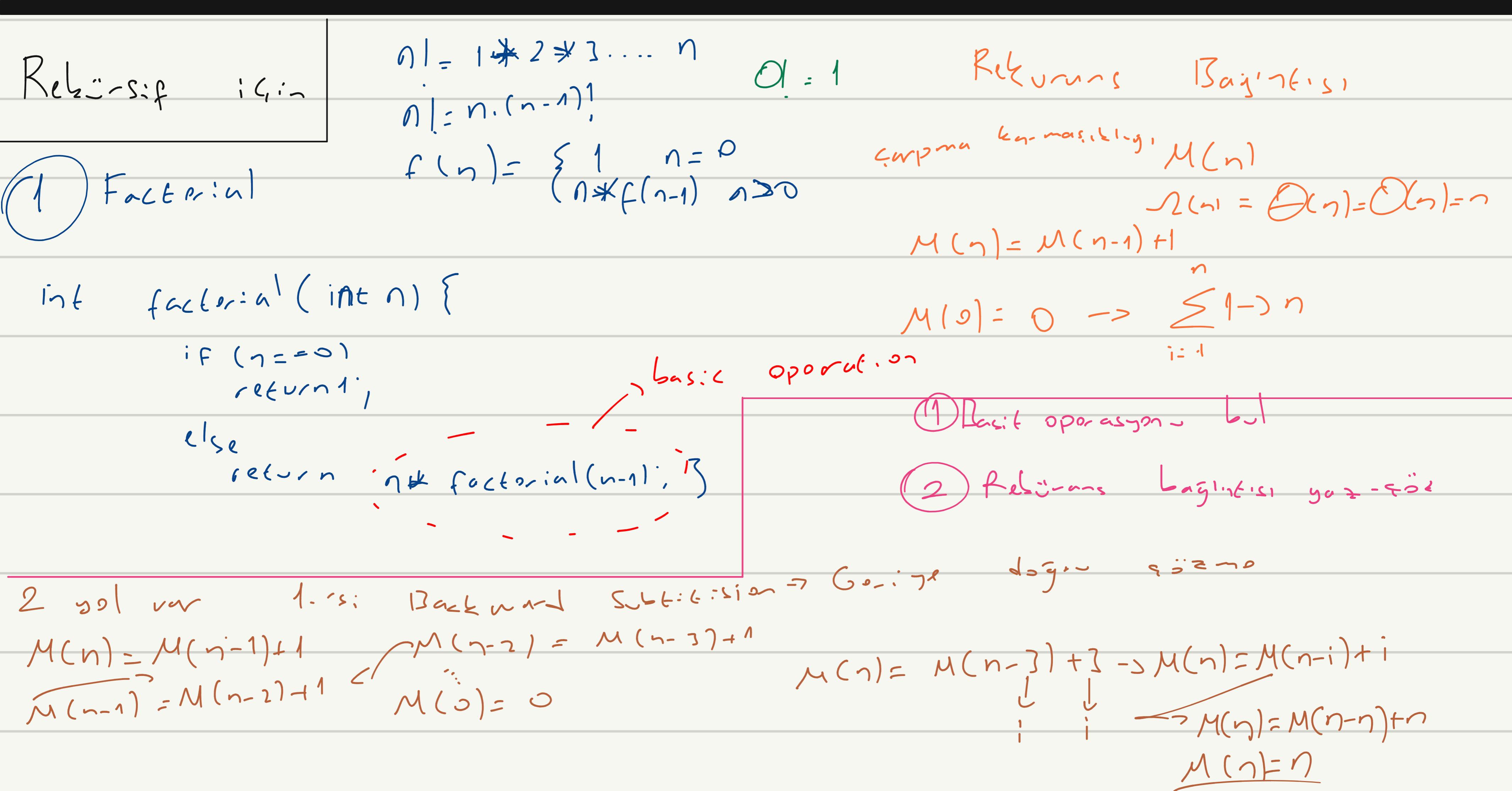
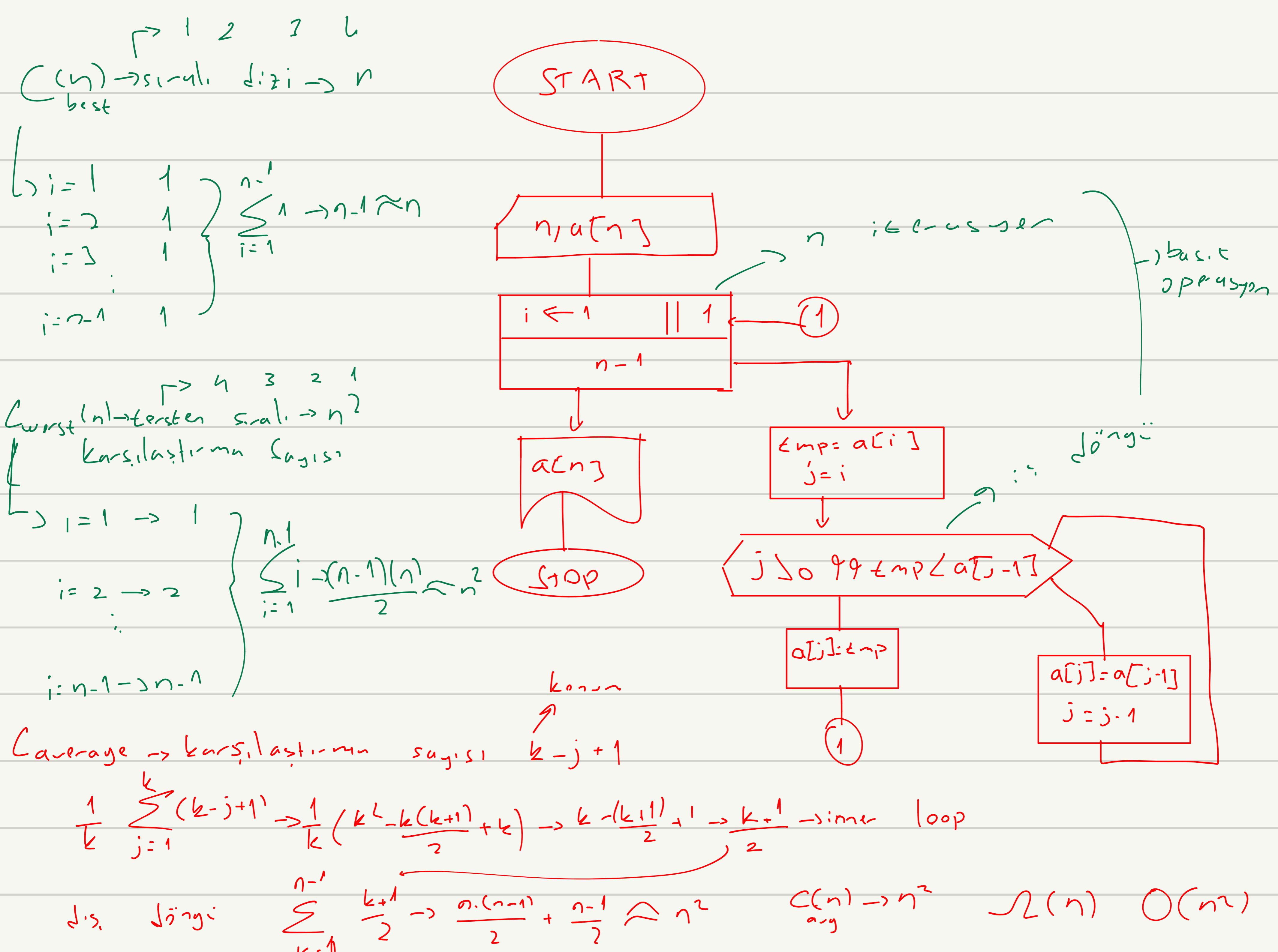
insertion Sort

| | | | | | |
|---|---|---|---|---|---|
| 9 | 3 | 7 | 1 | 4 | 5 |
| 3 | 9 | 7 | 1 | 4 | 5 |
| 3 | 7 | 9 | 1 | 4 | 5 |
| 1 | 3 | 7 | 9 | 4 | 5 |

:

:

1 3 4 5 7 9



Find the number of binary digits possible

```
int binaryDigits(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return 1 + binaryDigits(n/2); }  
↳ basic op
```

$$\begin{array}{r} 27 \\ \overline{-1} \quad \left| \begin{array}{r} 2 \\ 13 \end{array} \right| 2 \\ \overline{-1} \quad \left| \begin{array}{r} 6 \\ 0 \end{array} \right| 2 \\ \overline{-1} \quad \left| \begin{array}{r} 3 \\ 1 \end{array} \right| 1 \end{array} \rightarrow \text{rel}(n) = \text{rel}(n/2) + 1$$

$$A(n) = A(n/2) + 1 \quad n > 1 \quad \rightarrow A(n/2) = A(n/4) + 1 \quad \rightarrow A(n) = A(n/8) + 3$$
$$A(1) = 0 \quad \quad \quad A(n/4) = A(n/8) + 2 \quad \quad \quad A(n) = A(n/2^i) + i$$
$$\therefore A(n) = A(\frac{n}{2^{\log_2 n}}) + \log_2 n \quad \Theta(\log_2 n)$$

forward substitutions

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1)+1 & \text{otherwise} \end{cases}$$

| | |
|---------------|--------------------------|
| $\frac{n}{1}$ | $\frac{T(n)}{1}$ |
| 2 | $T(1) + 1 \rightarrow 2$ |
| 3 | 3 |
| 4 | 4 |
| : | : |
| n | n |

6. Afta

Insertion Sort

10^8 karsılastırma! Sanıge yapanlım bilyisayınca,

$\Theta(n^2)$

1000
anlık

1 milyon
2.8 saat

1 milyar
317 yıl

merge sort

anlık

1 saniye

18 dakika

$\Theta(n \log n)$

Quick sort

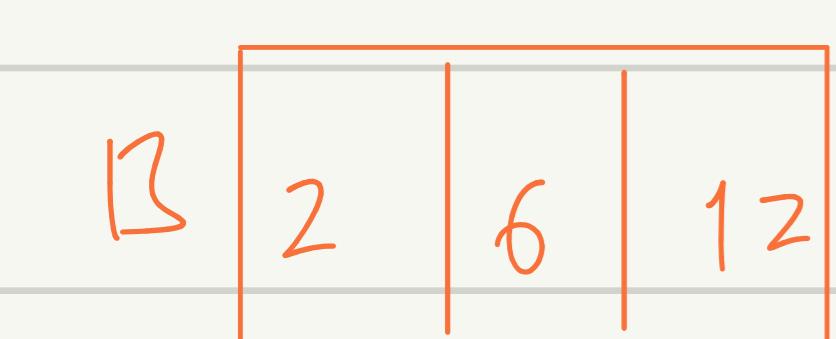
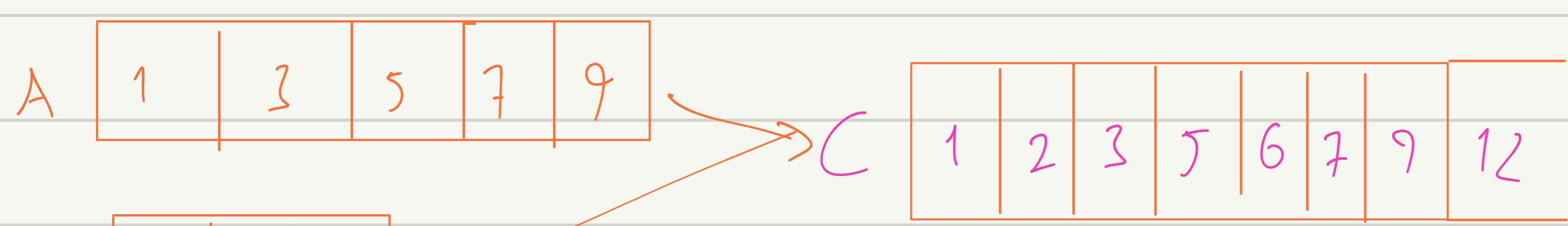
anlık

0.6 saniye

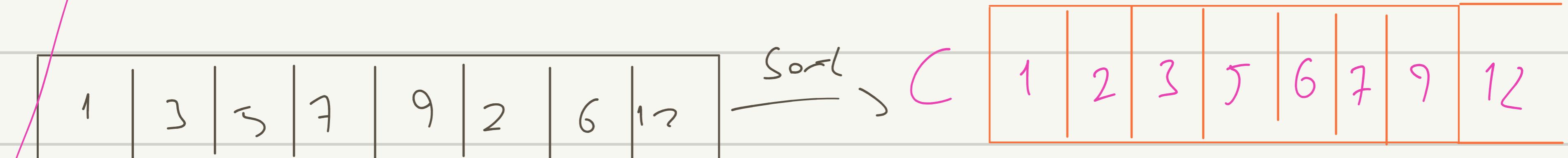
12 dakika

$\Theta(n \log n)$

Divide and conquer

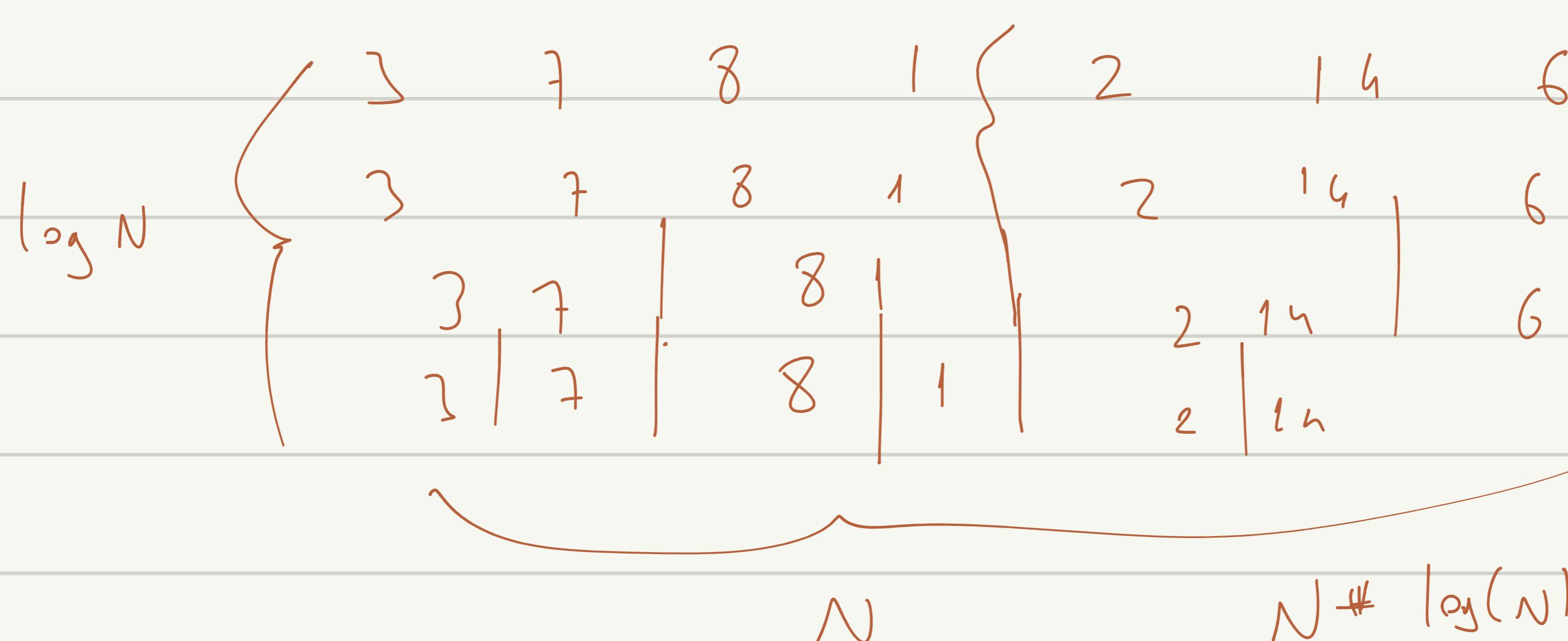


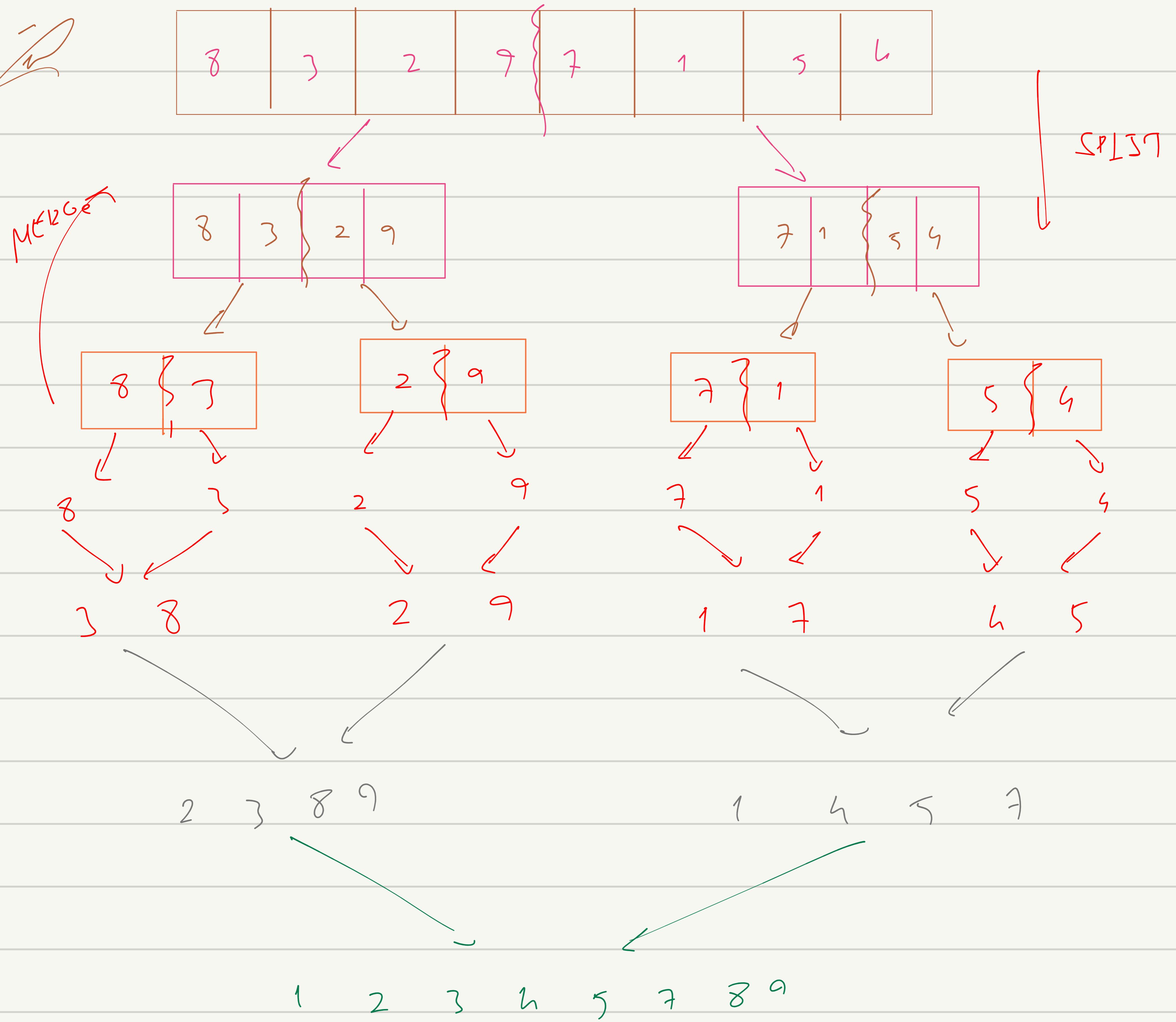
$\Theta(n^2)$



\rightarrow B'ın kontrolle üst üste elemanlar merge ve $\Theta(n^2)$ de olur.

Merge sort bu mantıkla en düzgün pozisyonlara töpü merge ederim.





DIVIDE and Conquer

- 1: DIVIDE \rightarrow Ayni algoritmay uygulayalı oqanızda kilek parçalarını bili
- 2: Bölükteki parçaları işle
- 3: " " birlestir

Merge Sort: 1-) SPLIT the array 2-) Sort each subsequence

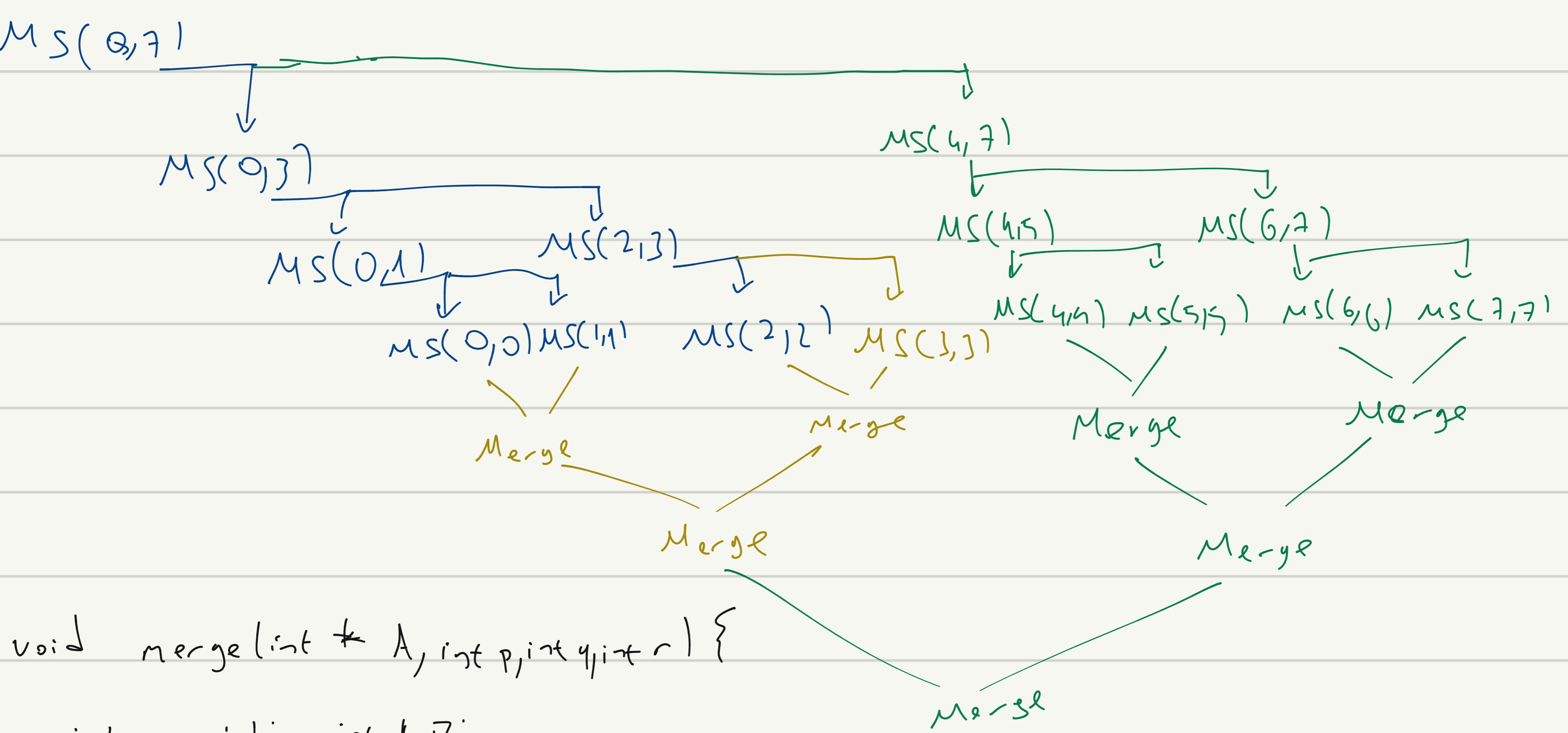
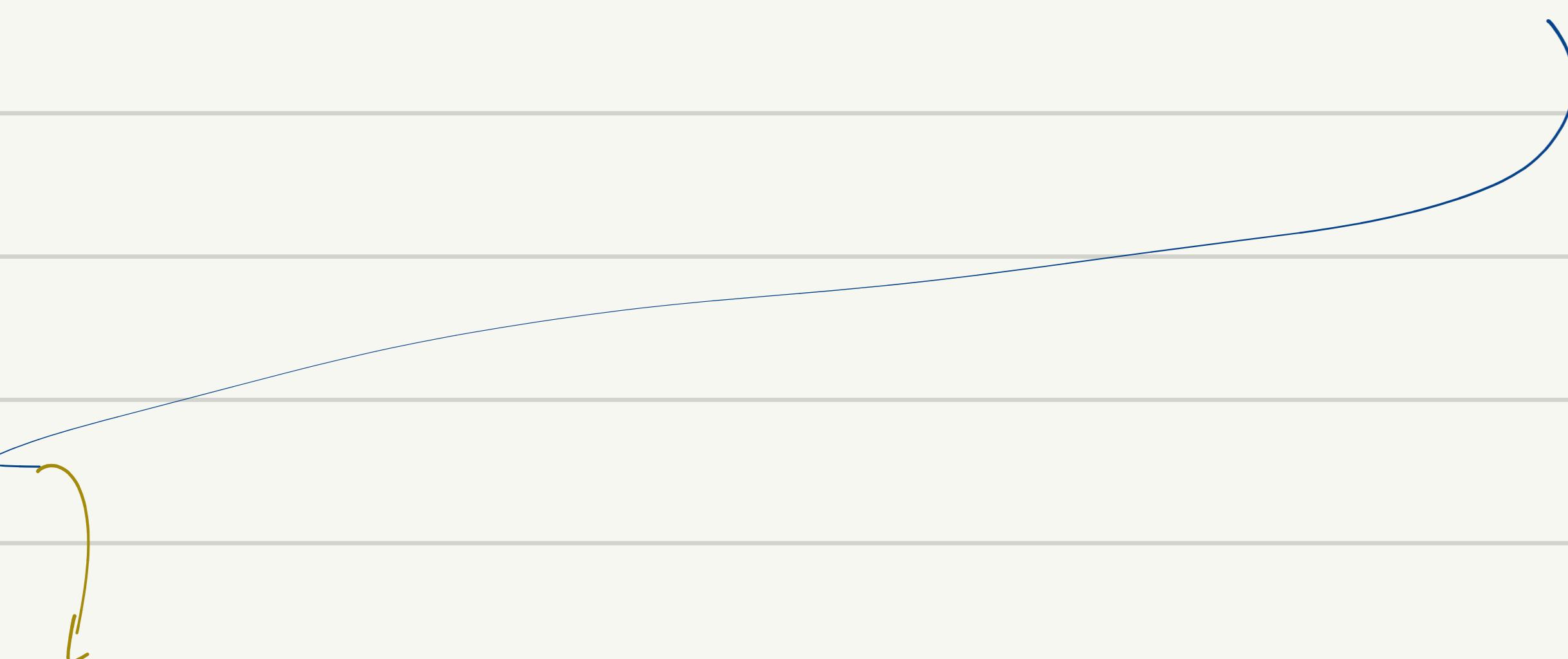
1-) merge the two sorted subsequence

$A[1 \dots n/2]$ $\rightarrow A[1 \dots n/2] \dots$
 $A[n/2 \dots n-1]$

```

void mergeSort (int *A, int p, int r) {
    if (p < r) {
        int q = (p+r)/2;
        mergeSort (A, p, q);
        mergeSort (A, q+1, r);
        merge (A, p, q, r);
    }
}

```



```
void merge (int *A, int p, int q, int r) {
```

```
int i, j, k, int *B;
```

```
n = r - p + 1; // elements says.
```

```
B = int *malloc (n * sizeof (int));
```

```
i = p, j = q + 1, k = 0;
```

```
while (i <= q and j <= r) {
```

```
if (A[i] < A[j])
```

```
B[k++] = A[i++];
```

```
else
```

```
B[k++] = A[j++], } -
```

```
while (i <= q) B[k++] = A[i++];
```

```
while (j <= r) B[k++] = A[j++];
```

```
for (i = p, k = 0; i <= r; i++, k++)
```

```
A[i] = B[k];
```

```
free (B); }
```

Recurrence Relation of Merge Sort

$$T(n) = \begin{cases} O(1) & n=0,1 \\ 2*T(n/2) + n & \text{if } n > 1 \end{cases}$$

$$\therefore \rightarrow O(\log_2 n + n) \rightarrow \Theta(\log_2 n + n)$$

Eğer $b=1$

Genel hali ve şonda da rekürans

$\xrightarrow{\text{bingerler, } b=1 \text{ ve } c=0}$ $T(n) = a*T(\frac{n}{b}) + f(n)$ is. yani

\checkmark Lükas, valid

$\exists f: f(n) \in \Theta(n^\alpha)$ when $\alpha \geq 0$ then $T(n)$

$\checkmark T(n) = T\left(\frac{2}{3}n\right) + 1$
 $a=0 \quad a=1 \quad b=\frac{3}{2}$

$$1^2 = \left(\frac{3}{2}\right)^0 \rightarrow \Theta(\log_2 n \cdot n^\alpha) = \Theta(\log_2 n)$$

Bottom-up Substitution

$$T(n) = 2*T(n/2) + n$$

$$T(n/2) = 2*T(n/4) + n/2$$

$$T(n/4) = 2*T(n/8) + n/4$$

$$T(n/8) = 2*T(n/16) + n/8$$

$$T(n/16) = 2*T(n/32) + n/16$$

$$T(n/32) = 2*T(n/64) + n/32$$

$$T(n/64) = 2*T(n/128) + n/64$$

$$\vdots$$

$$T(n) = 2*T(n/2^i) + n$$

$$= 2^i * T\left(\frac{n}{2^i}\right) + i * n$$

$$n = 2^i = n + T(1) + \log(n) * n$$

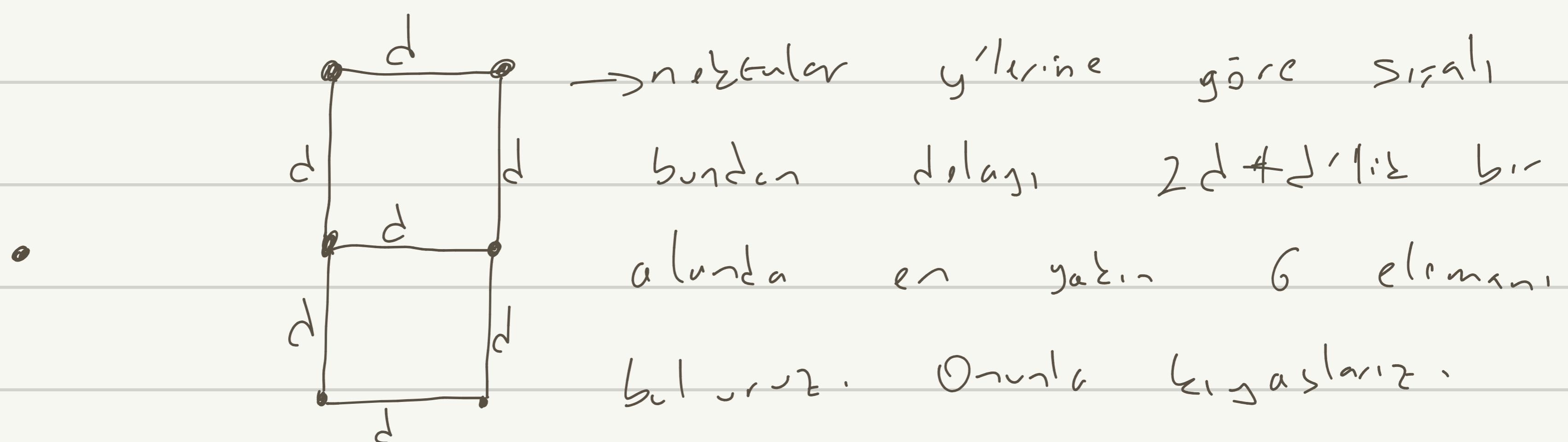
$$= n + n * \log(n)$$

$$\begin{cases} \Theta(n^\alpha) & \text{if } a < b^\alpha \\ \Theta(n^{\log_b a}) & \text{if } a = b^\alpha \\ \Theta(n^{\log_b a}) & \text{if } a > b^\alpha \end{cases}$$

not: Σ ve
 $O^d \downarrow$ olm.

LAB 1

n elemanlı dizinin var ve en yakın 2 elemani bul düşer



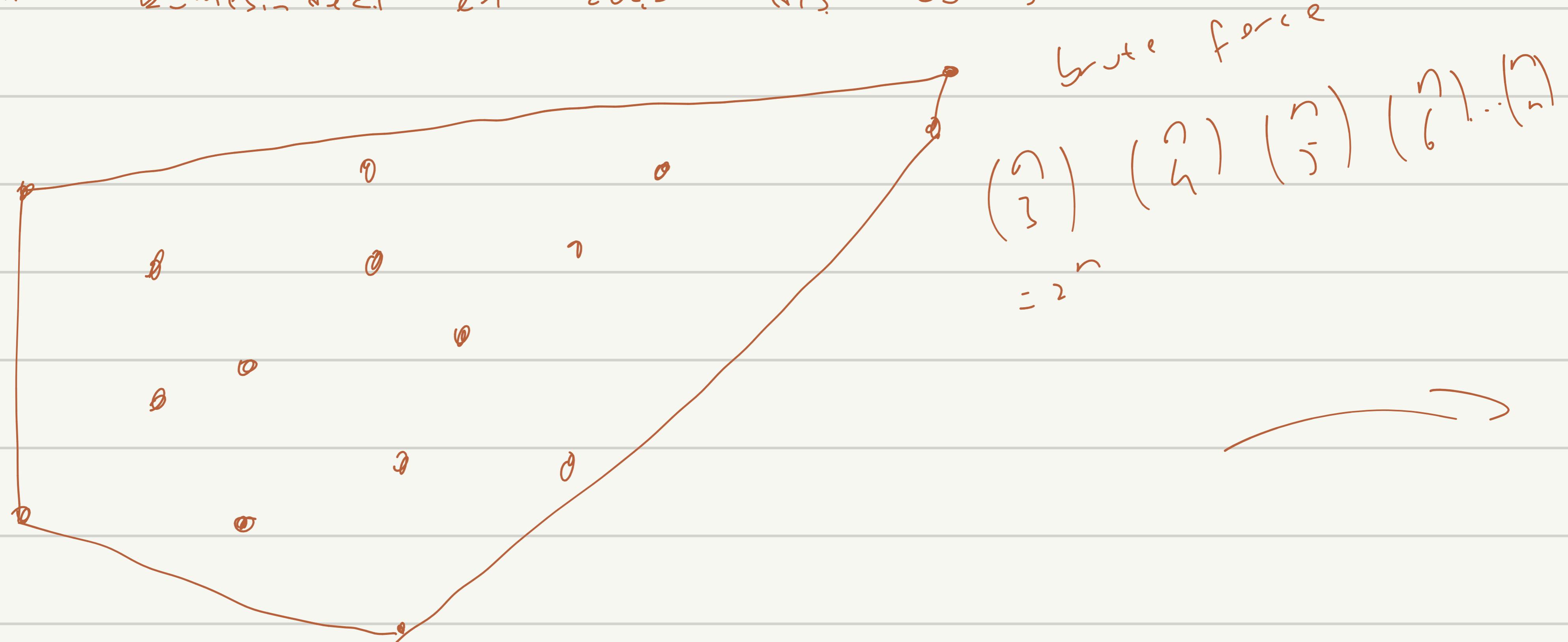
$$T(n) = 2T(\lceil \frac{n}{2} \rceil) + 6N$$

$$a=2 \quad b=2 \quad d=1 \quad a=b^{\alpha} \quad n \log n$$

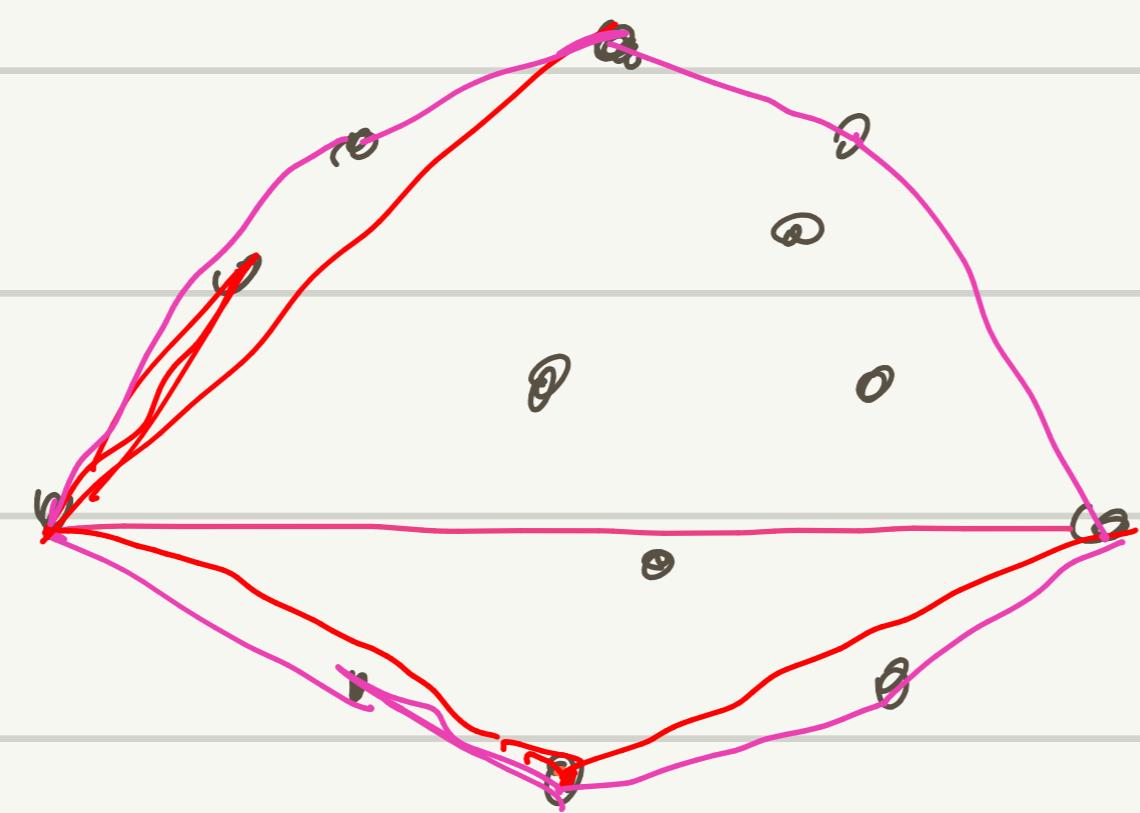
$$n < 3$$



noktalar kimesindeki en küçük dis büküsü



X eksenine göre en uzak 2 noktadan böleriz, sonra
n defa deneyerek Σ noktaya en uzak noktayı
seçiyoruz. Eğer dista kalar ve sa bası, nota seçiyoruz.

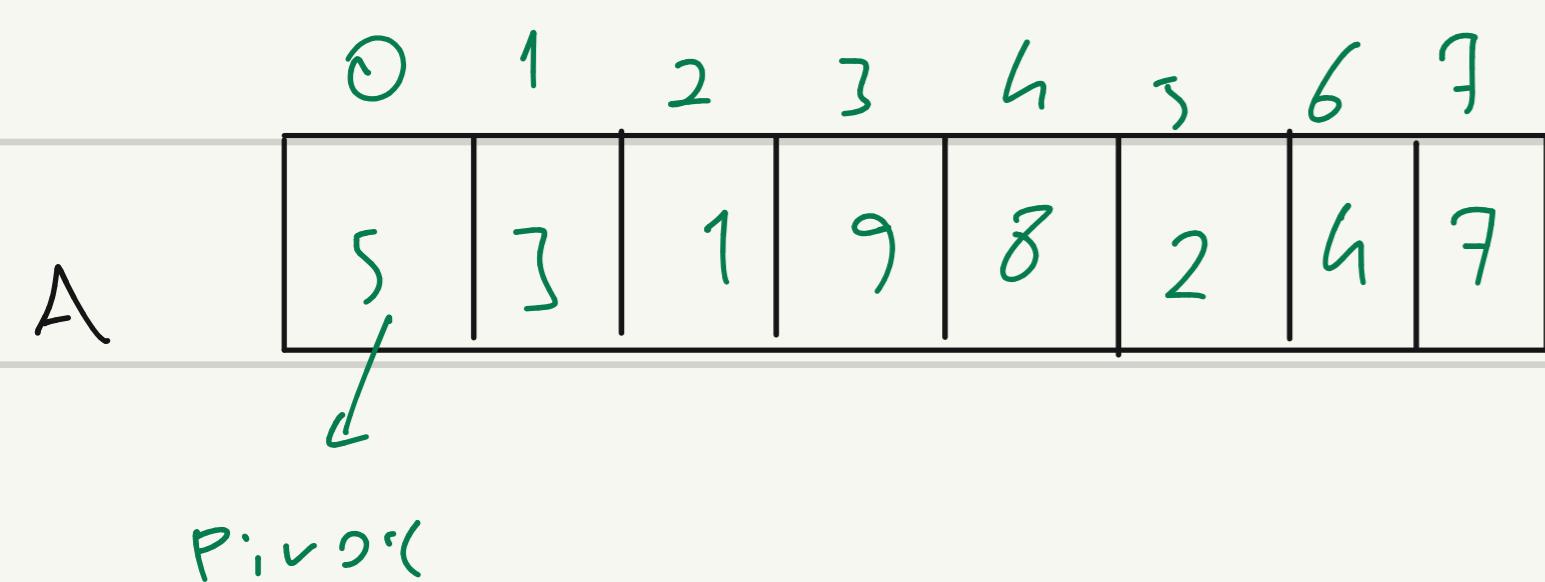


$$T(N) = T(N/2) + N$$

$$= n \log n \rightarrow \text{ortuluma}$$

Halfa 5

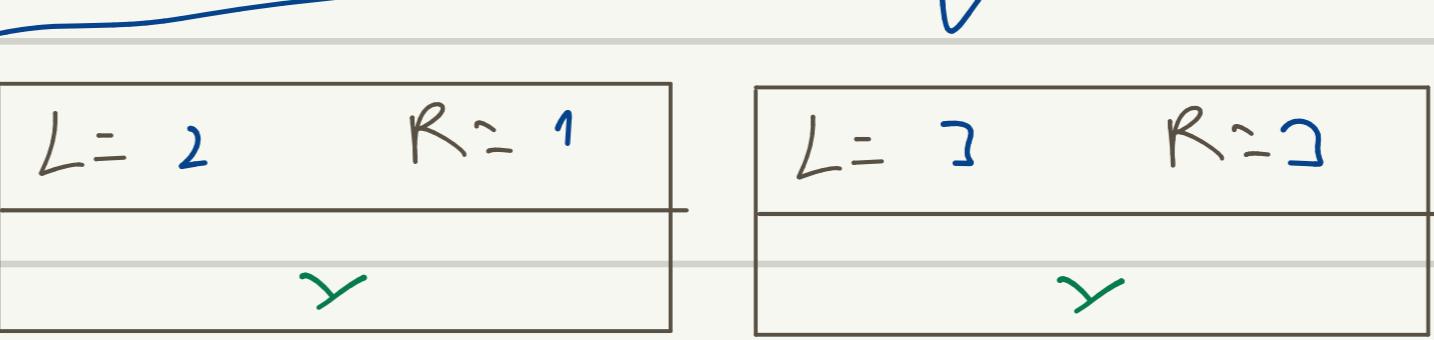
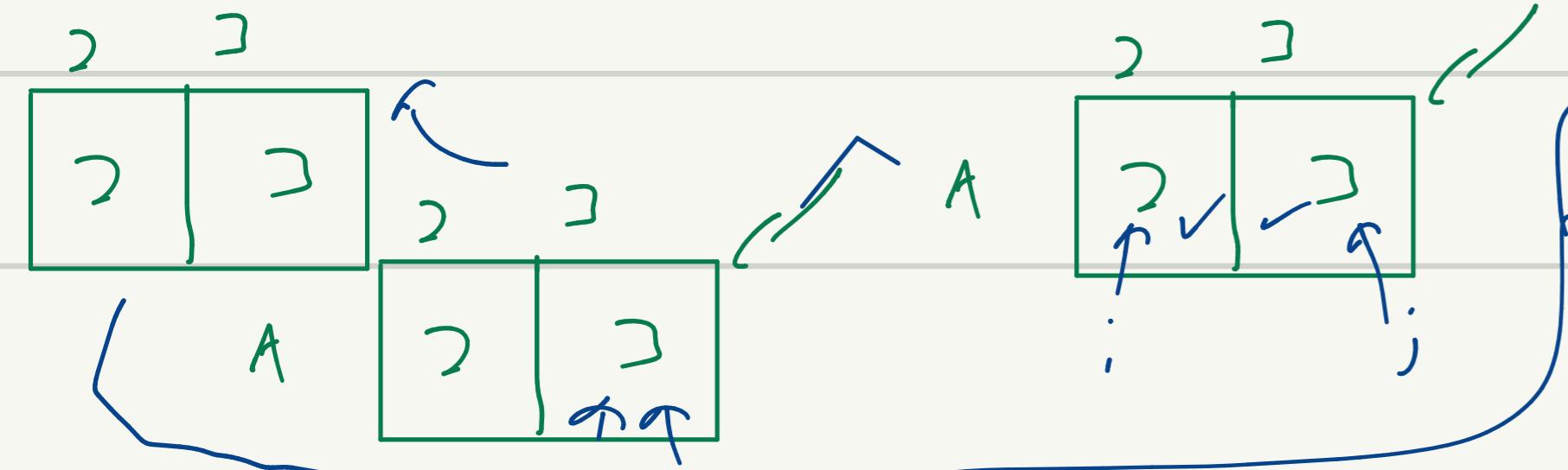
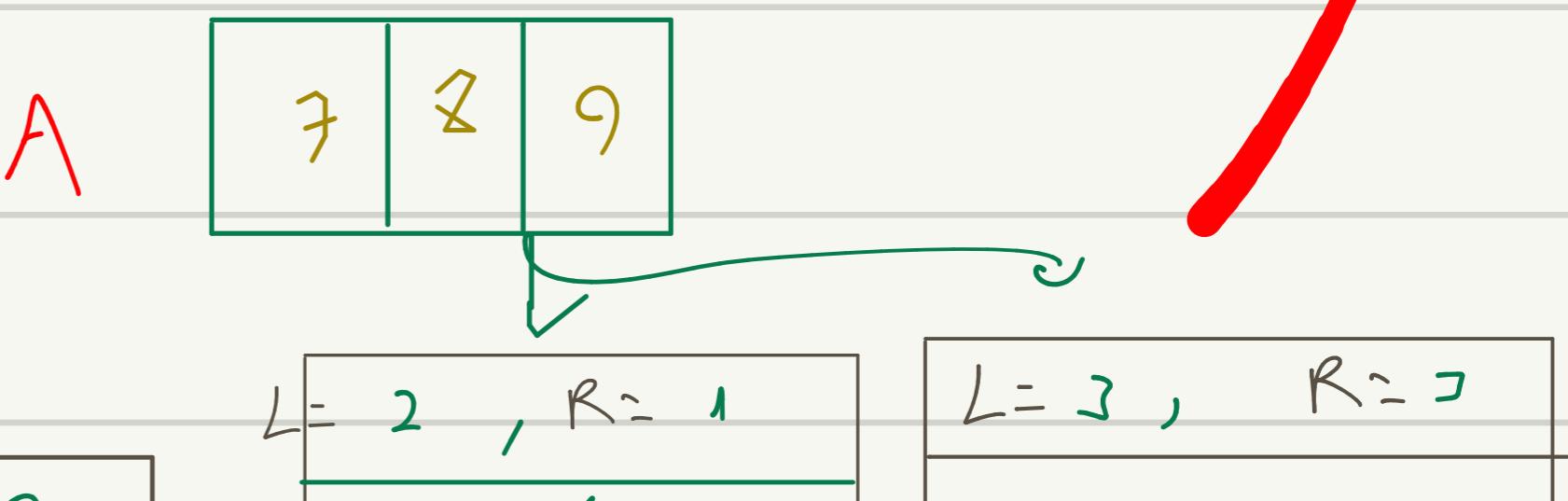
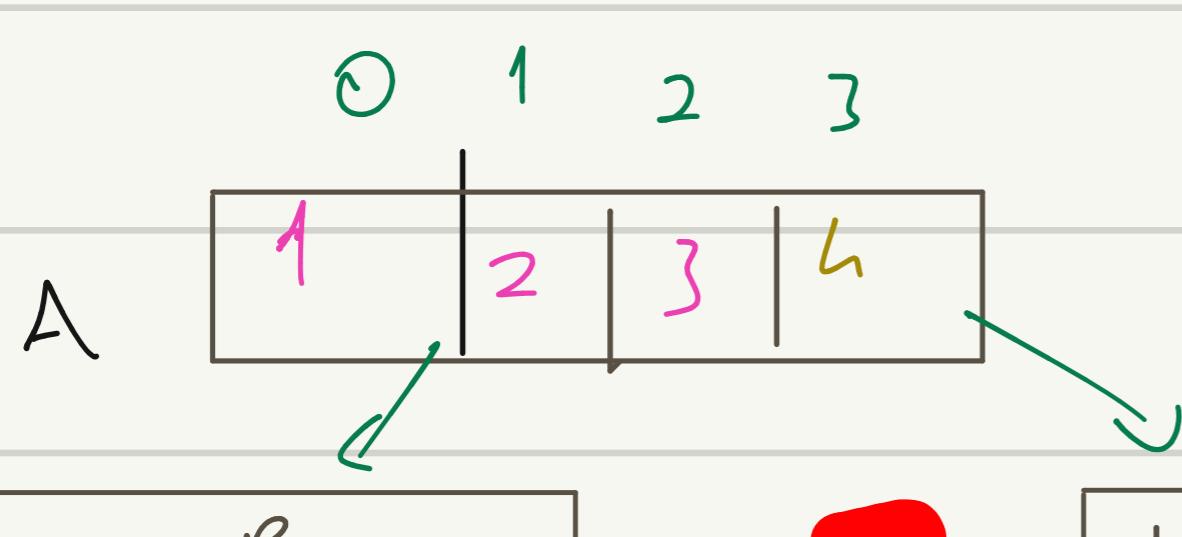
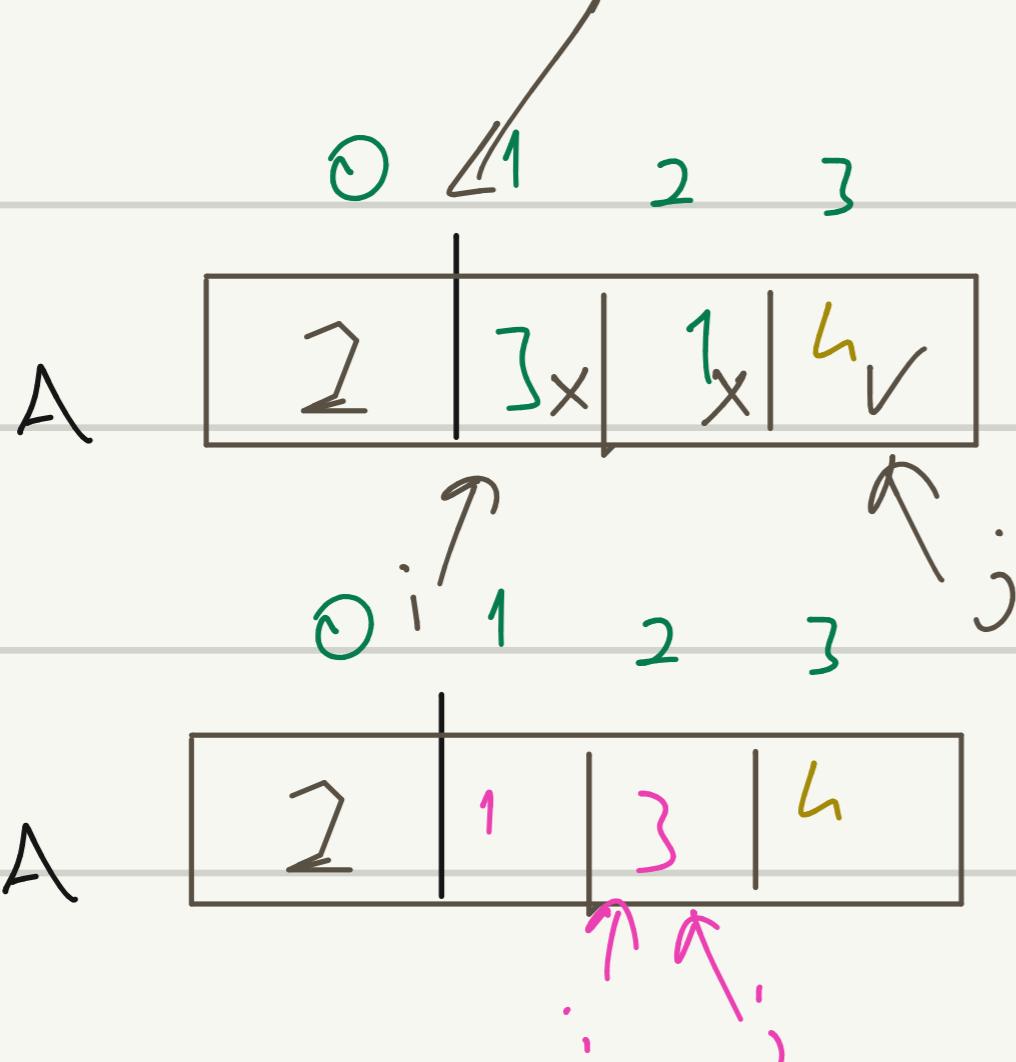
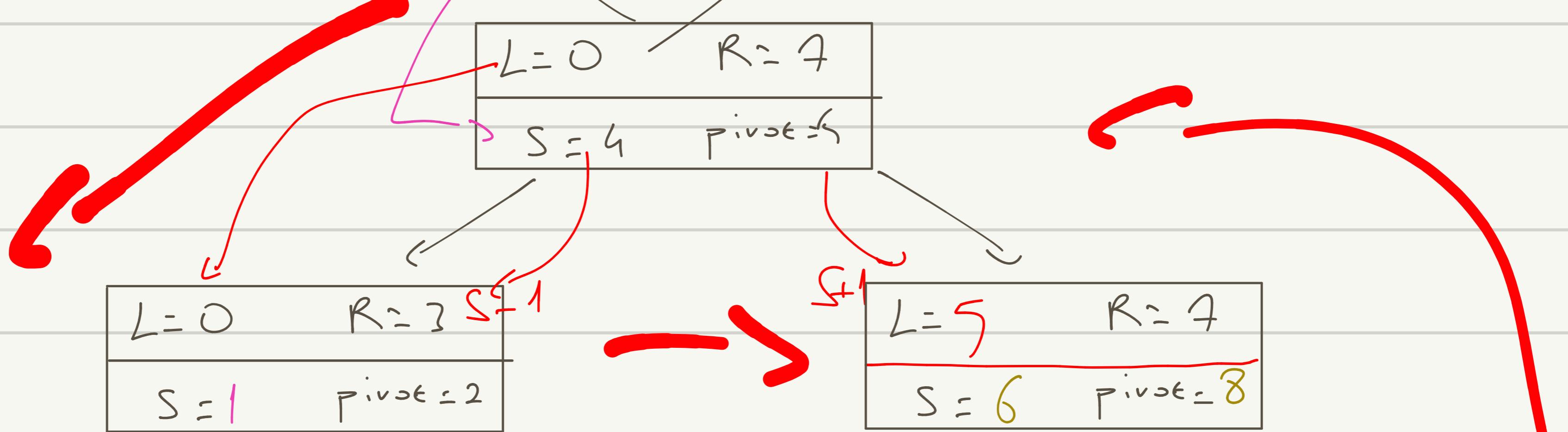
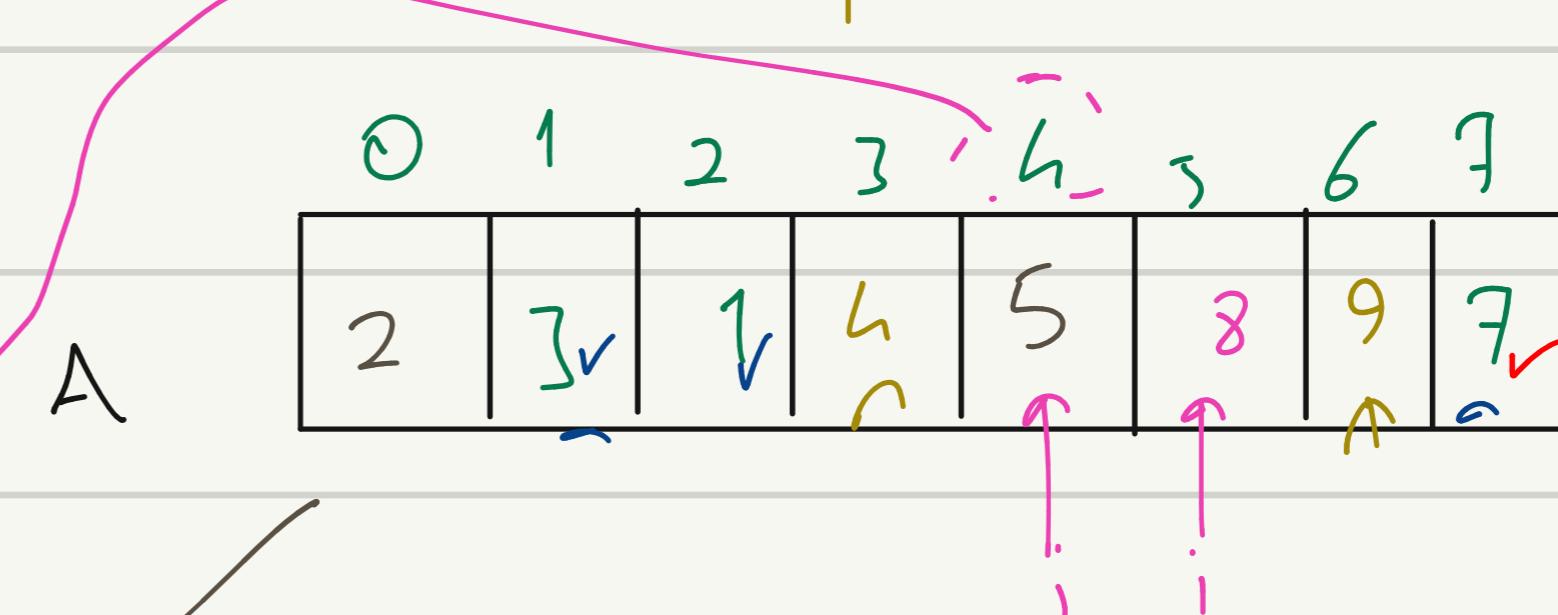
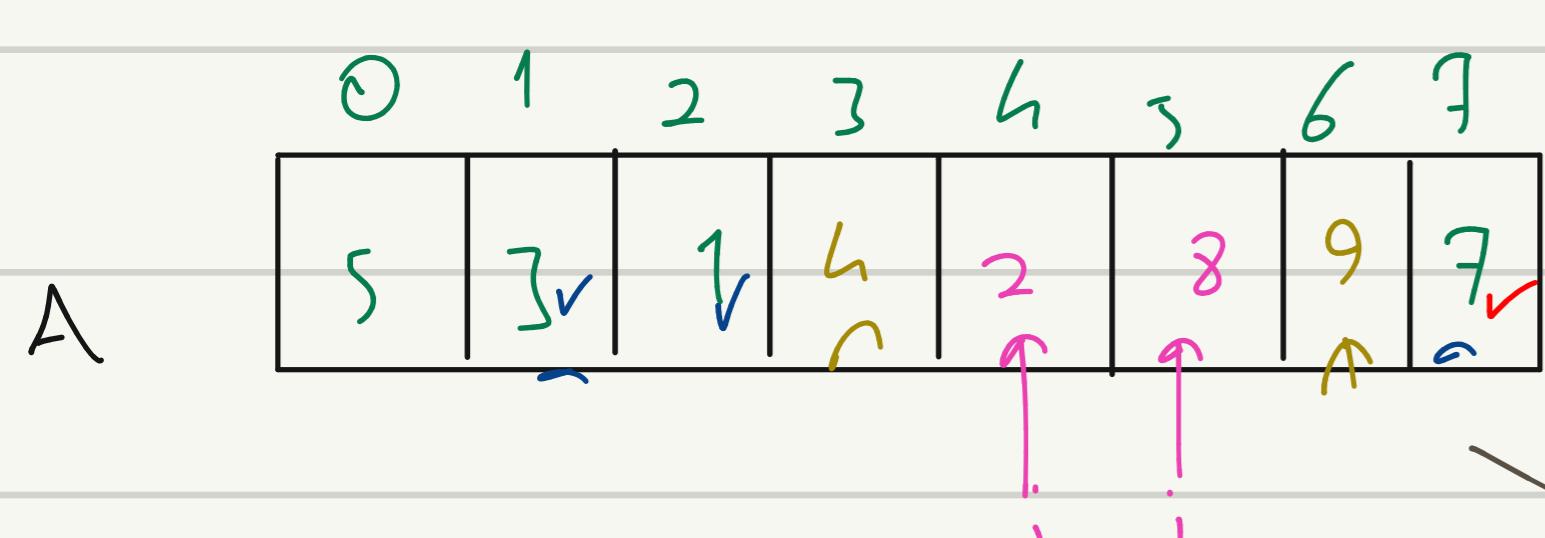
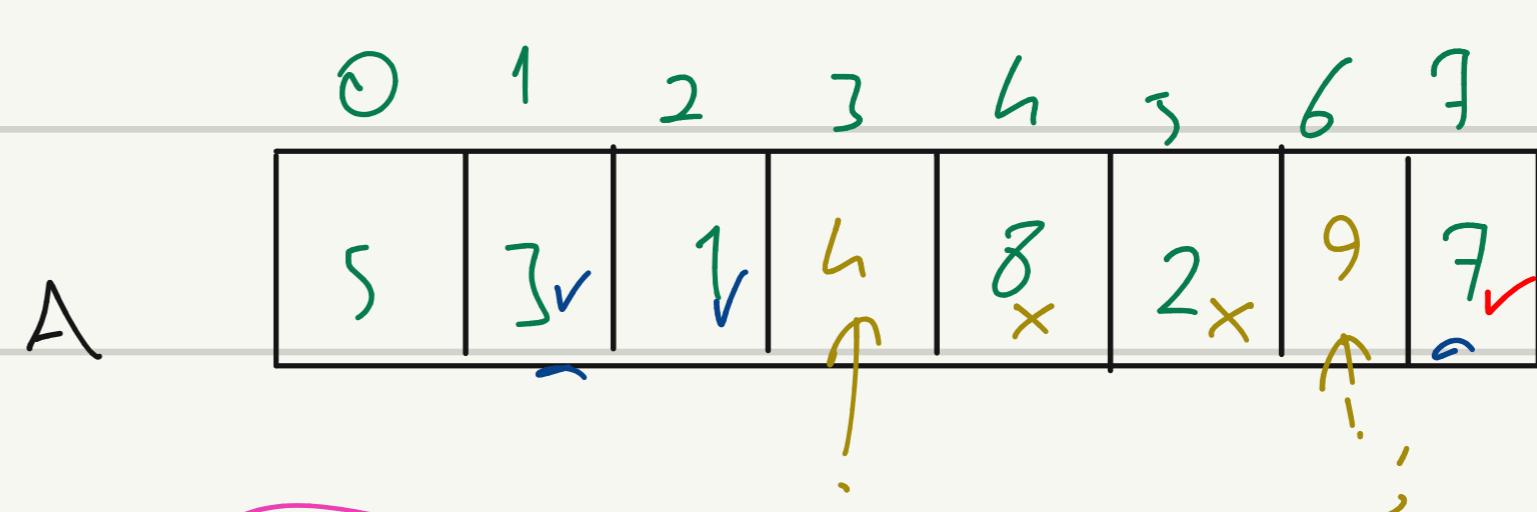
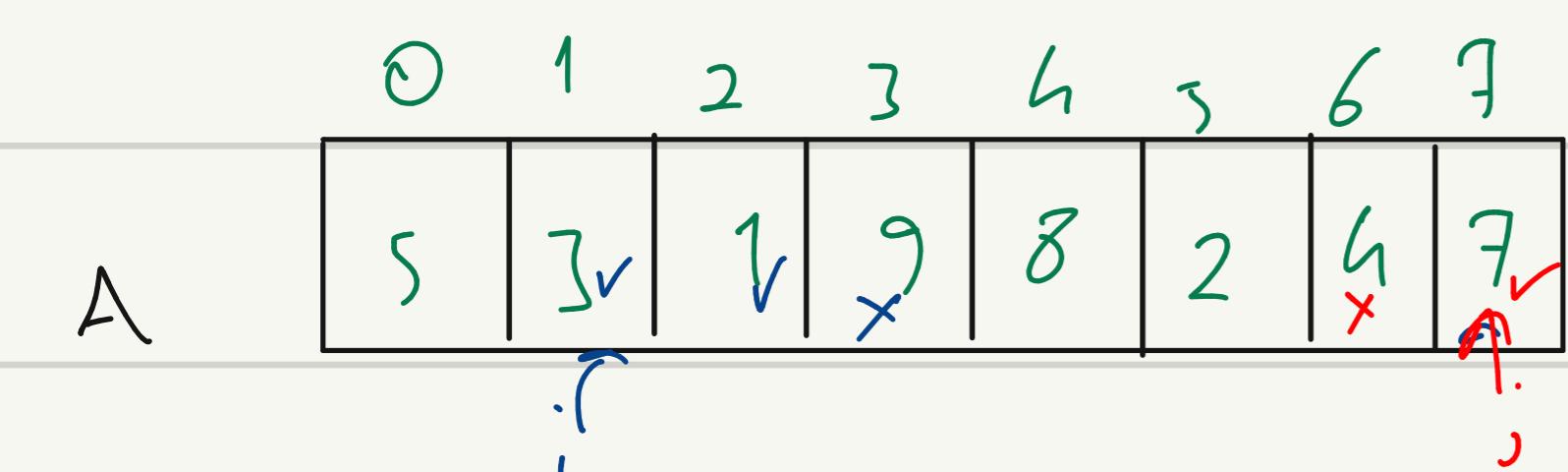
QuickSort → In Place



$A[0 \dots s-1]$ pivot $A[s+1 \dots n-1]$

$A < \text{pivot}$ $\text{pivot} < A$

3 2 1 4 $\overbrace{7}$ 11 13 8 12



PARTITIONING

```
quicksort(n, A[l....r]&{
    if(l < r &{
        s=partition(A[l...r]&; //Pivot yerine yerleşir küçükler
        solunda büyükler sağında kalır ÖDEV
        quicksort(A[l...s-1]&;
        quicksort(A[s+1.....r]&);})}
```

Best Case $\rightarrow T(n) = 2T(n/2) + n$ $T(1) = 0$

$$a=2 \quad b=2 \quad d=1$$

$$2^2 \text{ olduğu için } n^{\log_2 2} \rightarrow n^1$$

Pivot orta Japon best case her zaman olurdu - $\Theta(n \log n)$

Worst Case

büyükten büyüğe-büyükten küçüğe sıralı olursa

$$T(n) = T(n-1) + n \quad \text{for } (n > 1) \quad \Theta(n^2)$$

$$T(0) = T(1) = 1$$

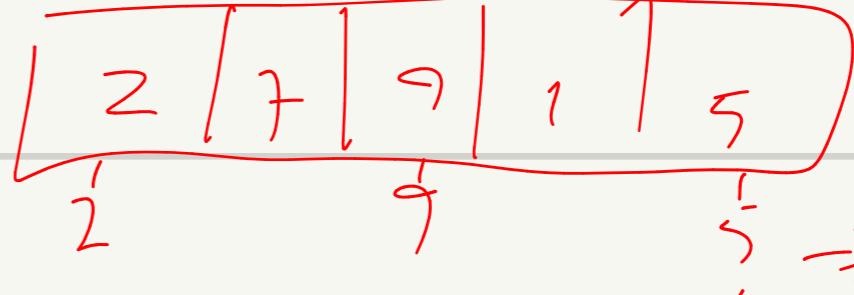
Average

| Position of pivot | left of pivot | .. of p. | $T(n_{left})$ | $T(n_{right})$ |
|-------------------|---------------|----------|---------------|----------------|
| 0 | 0 | $n-1$ | $T(0)$ | $T(n-1)$ |
| 1 | 1 | $n-2$ | $T(1)$ | $T(n-2)$ |
| 2 | 2 | $n-3$ | $T(2)$ | $T(n-3)$ |
| : | | | | |
| $n-1$ | $n-1$ | 0 | $T(n-1)$ | $T(0)$ |

$$T(n) = \frac{1}{N} \sum_{k=1}^N (T(k-1) + T(n-k) + n) \rightarrow T(n) = 1 + N \log_2(N) \in O(n \log_2 n)$$

\hookrightarrow QS merge S. ile % 39 fazla kıyaslaması yapıyor ama elemanın değişimi uz onda dolaylı daha hızlı!

worst case den kürzeste gültig

- ① kürzestes
 - ② median of samples \rightarrow 
 - ③ random pick pivot
- SI: Basal al

Decrease And Conquer

- ① Decrease by a constant factor

binary search

problem of size n

" " " " $n/2$

solution to the original problem

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

- ② Variable size decrease

$$\text{GCD}(a, b) = \begin{cases} a & \text{if } b = 0 \\ \text{GCD}(b, a \% b) & \text{otherwise} \end{cases}$$

$$\text{GCD}(80, 44)$$

$$\hookrightarrow \text{GCD}(44, 36)$$

$$\hookrightarrow \text{GCD}(36, 8)$$

$$\hookrightarrow \text{GCD}(8, 4)$$

$$\hookrightarrow \text{GCD}(4, 0)$$

$$\hookrightarrow 4$$

- ② Decrease by a constant topological sort

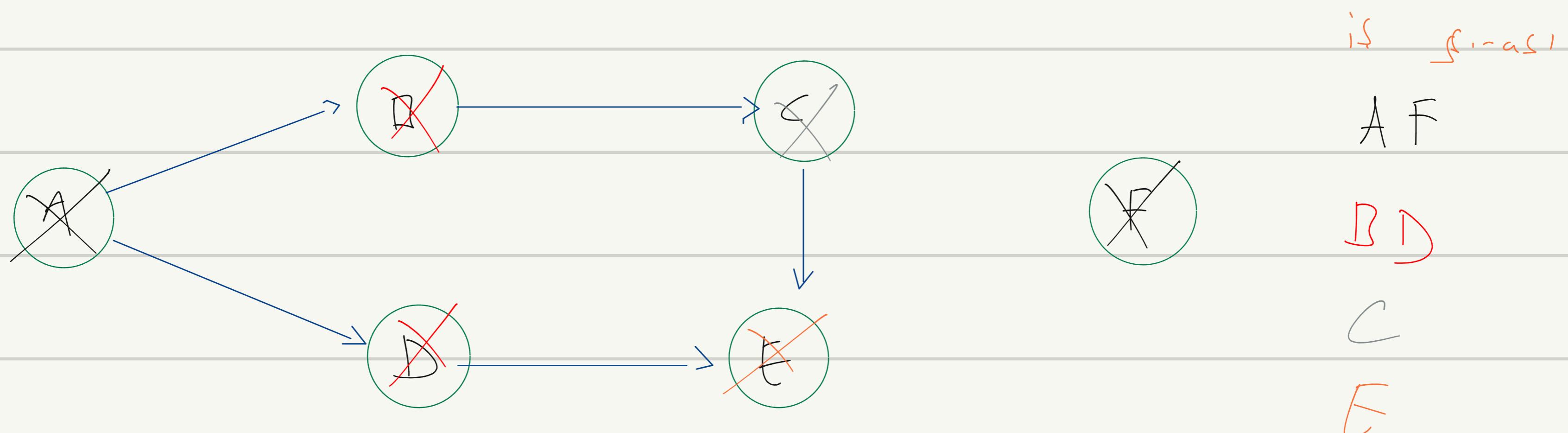
G. Hafıza

Decrease and conquer

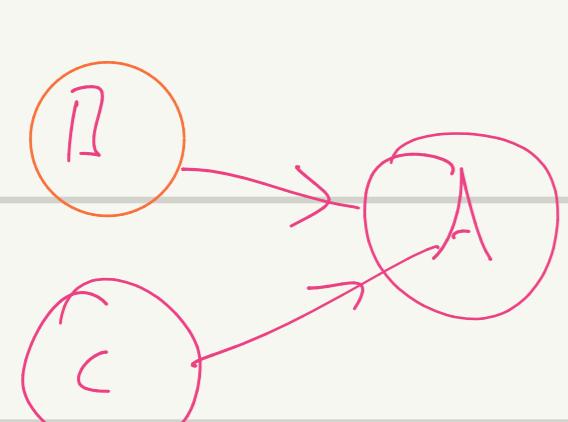
Decrease by a constant

Topological sort → Graf'a sıralama

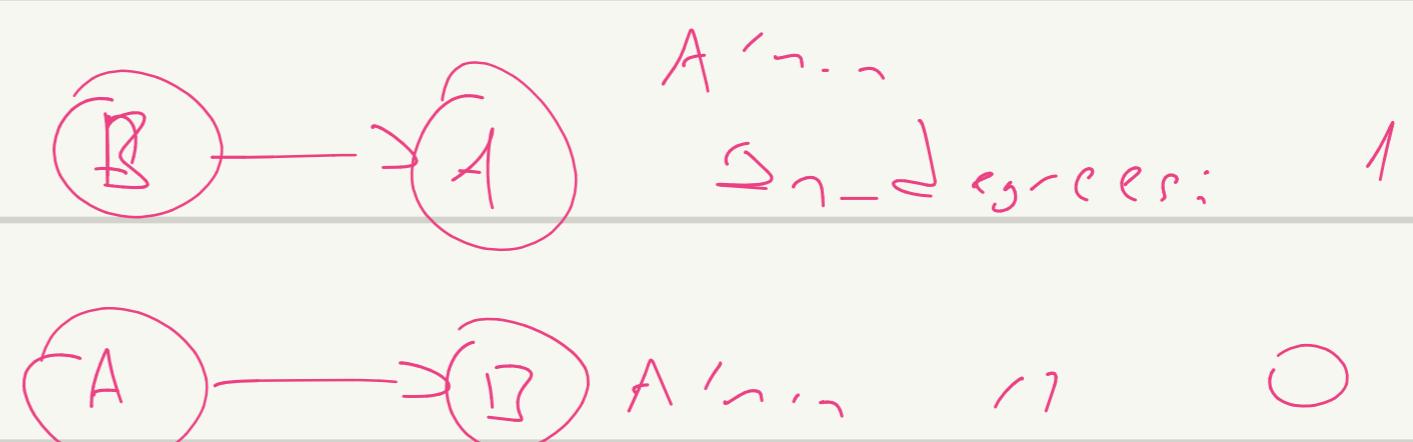
Dairesel olmamalı, birer birer olmak için surolar verilir:



In-degree



A'nın
in-değeri: 2



A'nın
in-değeri: 1



A'nın
in: 1

Konsoluk Listesi:

in-degree

| | |
|---|-----------|
| 0 | A → B → D |
| 1 | B → C |
| 1 | C → E |
| 1 | D → E |
| 2 | E → Null |
| 0 | F → Null |

indegree sayısı Lümla

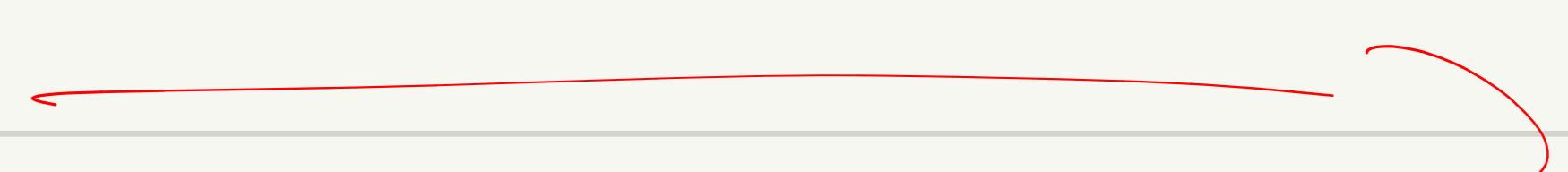
(listeyi gezeniz, nextlediginiz neye eklesin
onu artırmayın)

↳ Bu bir işin yapılabilesi için başlıca biri bittesi:

gerçekliğini söylemek ~~E~~ E işin önce A, sonra B, sonra C'nin bittesi lazımlı. A'nın indegree[E]=2

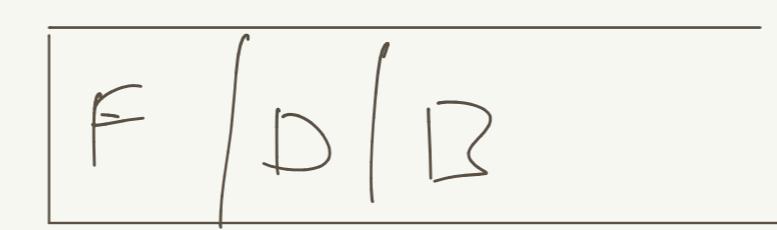
~~E~~ A bitince B ve C'si azaltmak lazımlı. $\text{Indegree}[B] =$

Hangi isten başlayacağının bittisi işin queyeğe atınız -

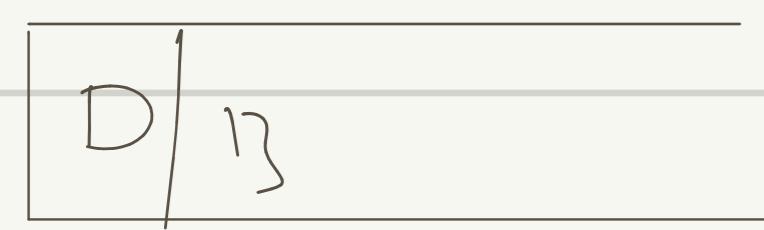




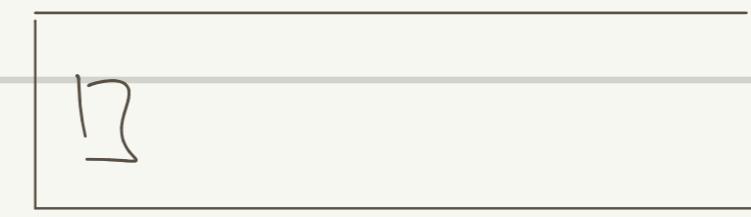
| | |
|---|---------------|
| 0 | A → B → C → D |
| 1 | B → C |
| 2 | C → F |
| 1 | B → E |
| 2 | E → Null |
| 0 | F → Null |



| | |
|---|---------------|
| 0 | A → B → C → D |
| 0 | B → C |
| 1 | C → F |
| 0 | B → E |
| 2 | E → Null |
| 0 | F → Null |

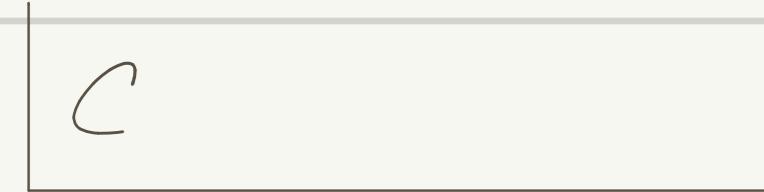


| | |
|---|---------------|
| 0 | A → B → C → D |
| 0 | B → C |
| 1 | C → F |
| 0 | B → E |
| 2 | E → Null |
| 0 | F → Null |



| | |
|---|---------------|
| 0 | A → B → C → D |
| 0 | B → C |
| 1 | C → F |
| 0 | B → E |
| 1 | E → Null |
| 0 | F → Null |

Queue



| | |
|---|---------------|
| 0 | A → B → C → D |
| 0 | B → C |
| 0 | C → F |
| 0 | B → E |
| 1 | E → Null |
| 0 | F → Null |



| | |
|---|---------------|
| 0 | A → B → C → D |
| 0 | B → C |
| 0 | C → F |
| 0 | B → E |
| 0 | E → Null |
| 0 | F → Null |

Que size $\rightarrow O(V+E)$

① Store each vertex's in-degree in array $O(E)$

② While there are vertices remaining:

find a vertex with in-degree zero $O(V^2)$

reduce in-degree of all vertices adjacent to it by 1 $O(E)$

mark in vertex's in-degree - 1

Queue $\rightarrow O(V+E)$

① Store each vertex's in-degree in array $O(E)$

② Initialize a queue with all in-degree zero vertices $O(V)$

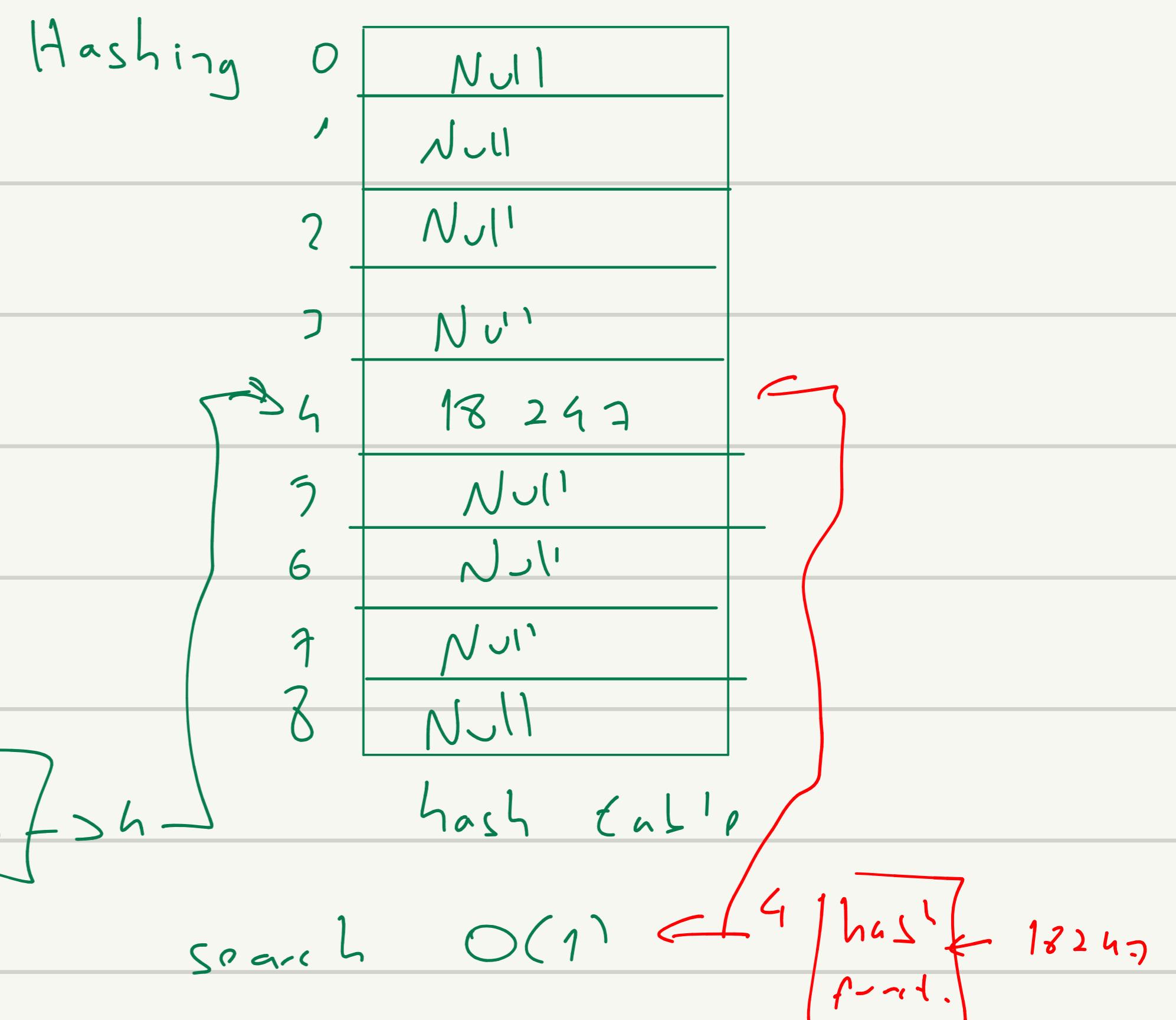
③ While there are vertices remaining in the queue

- Dequeue and output $O(V)$

- Reduce in-degree of all vertices adjacent to it by 1 $O(E)$

- Enqueue vertices with indegree zero $O(E)$

| | Worst | | Average | | |
|-------------|------------|------------|------------|---------------|---------------|
| | Search | Insert | Search | Insert | Delete |
| Linear List | n | 1 | n | $\frac{n}{2}$ | 1 |
| Spiral List | $\log_2 n$ | n | n | $\log_2 n$ | $\frac{n}{2}$ |
| Binary Tree | n | n | n | $\log_2 n$ | $\log_2 n$ |
| Dense Tree | $\log_2 n$ | $\log_2 n$ | $\log_2 n$ | $\log_2 n$ | $\log_2 n$ |



① Hash table'in uzunluğu ne olmalı? Ayni indekslerdeki sayılar aynı anamak iken.

② Hash fonksiyonu nasıl sağlanır?

③ Çarpma ve bölme ne olursa

Hash functions

1 Division method

$$\text{hash function (key)} = \text{key} \bmod m$$

$$\text{key} = 25$$

$$\text{hash}(25) = 25 \% 7 \rightarrow 6$$

$$\text{hash}(7) = 7 \% 7$$

| | |
|---|----|
| 0 | 7 |
| 1 | |
| 2 | |
| 3 | |
| 4 | 25 |
| 5 | |
| 6 | |

table size → 0 → ~1
 asal sayı olmalıdır
 2'nin katları olamaz
 da 5 olabilir.
 Amma 7 olursa
 7'nin katları 97

knot

2 Multiplication method

$$\text{hash function (key)} = \lfloor m * (\text{key} * A) \bmod 1 \rfloor$$

↑
size

$$A = \frac{\sqrt{s}-1}{2} = 0,618033$$

$$\text{key} = 123.456$$

$$\text{hash function (123.456)} = \lfloor 10.000 * [(123.456 * 0.618033) \bmod 1] \rfloor$$

$$123.456 * 0.618033 \bmod 1 \\ = 0.0000000000000001$$

$$10.000 * 0.0000000000000001$$

$$41 \quad |123.456|$$

$$41.151 = 41$$

HAFTA 7

Hashing

1-) Division method

$$\text{hash}(\text{key}) = \text{key} \% m \quad \xrightarrow{\text{table size}}$$

2-) M-1e. ..

$$\text{hash}(\text{key}) = \lfloor m * (\lceil \log_2 k \rceil \% 1) \rfloor$$

Collision problem
↳冲突

① Open Addressing

| 0 | ... | - - - | m-1 | table size PRIME |
|---|-----|-------|-----|------------------|
| 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 |

key = 5
 $\text{hash}(\text{key}) = 5 \% 11$

key = 9
 $\text{hash}(\text{key}) = 9 \% 11$

key = 16
 $\text{hash}(\text{key}) = 16 \% 11 = 5$??

1.1 Linear Probing → each check is a probe

$$\text{hash}(\text{key}, i) = [\text{hash}(\text{key}) + i] \% m \quad \left. \begin{array}{l} \text{6. size } \text{bos,} \\ \text{insert} \end{array} \right)$$

key = 16 i=0 $\rightarrow \text{hash}(16, 0) = 5$

key = 16 i=1 $\rightarrow \text{hash}(16, 1) = (5+1) \% 11 = 6$

Arama işleminde Null girişi kadar ya da bası düğüm kadar döner. Search(key, i)

kontrol: $i \leq m$ $M \neq N$.

$\rightarrow \text{while}(\text{hash}[\text{adr}] \neq \text{Null} \text{ } \& \& \text{ } i < m \text{ } \& \& \text{hash}[\text{adr}] \neq \text{key}) \{ \}$

Linear Probing'ın clustering problemi var \rightarrow Mesela 5 5'ü geldi:

16 da 6'yu geldi. Bu sonucu iki in 6 asıl yerine gelecektir. Bundan dolayı

1.2.) Double Hashing

$$\text{hash}(k, i) = [\text{hash}_1(k) + i * \underbrace{\text{hash}_2(k)}_{\text{jump size}}] \% m$$

$k=16 \quad \text{hash}(16)=1$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|----|----|---|---|---|---|---|---|----|----|----|----|
| 0 | 16 | 79 | | | | | | | | | | 17 |

$\text{hash}_1(\text{key}) = \text{key} \% 13$

$k=79 \quad \text{hash}(79)=1 \rightarrow \text{hash}(79, 1) = (1 + 1 * 2) \% 13 \quad (\rightarrow 3)$

$\text{hash}_2(\text{key}) = \text{key} \% 11$

Büyükdeas paradox

2 kişiinin doğum gününün farklı olma olasılığı

$$\frac{365}{365} \times \frac{364}{365}$$

$$k \text{ kişi} \rightarrow \prod_{i=0}^{k-1} \frac{365-i}{365} \quad \begin{matrix} \rightarrow \text{T able size} \\ m \cdot m-1 \\ m \end{matrix} \quad k=2 \rightarrow \prod_{i=0}^{k-1} \frac{m-i}{m}$$

Taylor expansion: Yaklaşık olarake

$$e^{-\frac{(k-1)k}{2m}} = P$$

load factor

$$\text{collision problem} = 1 - e^{-\frac{(k-1)k}{2m}}$$

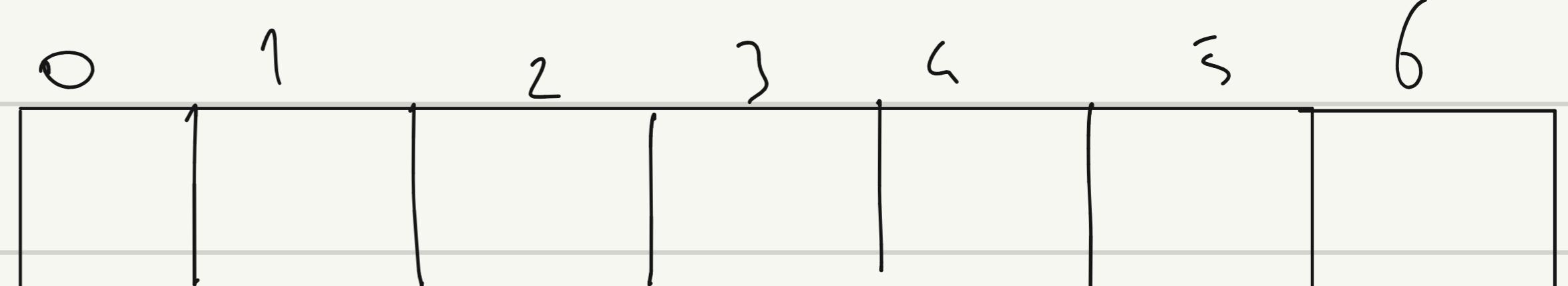
enkripsi olasılığı

$$\frac{N}{M} \quad \frac{10}{100} \rightarrow \% 10$$

Open addressingde load factor \rightarrow yaklaşık $\% 50$

$$\frac{A_m}{M} \approx N$$

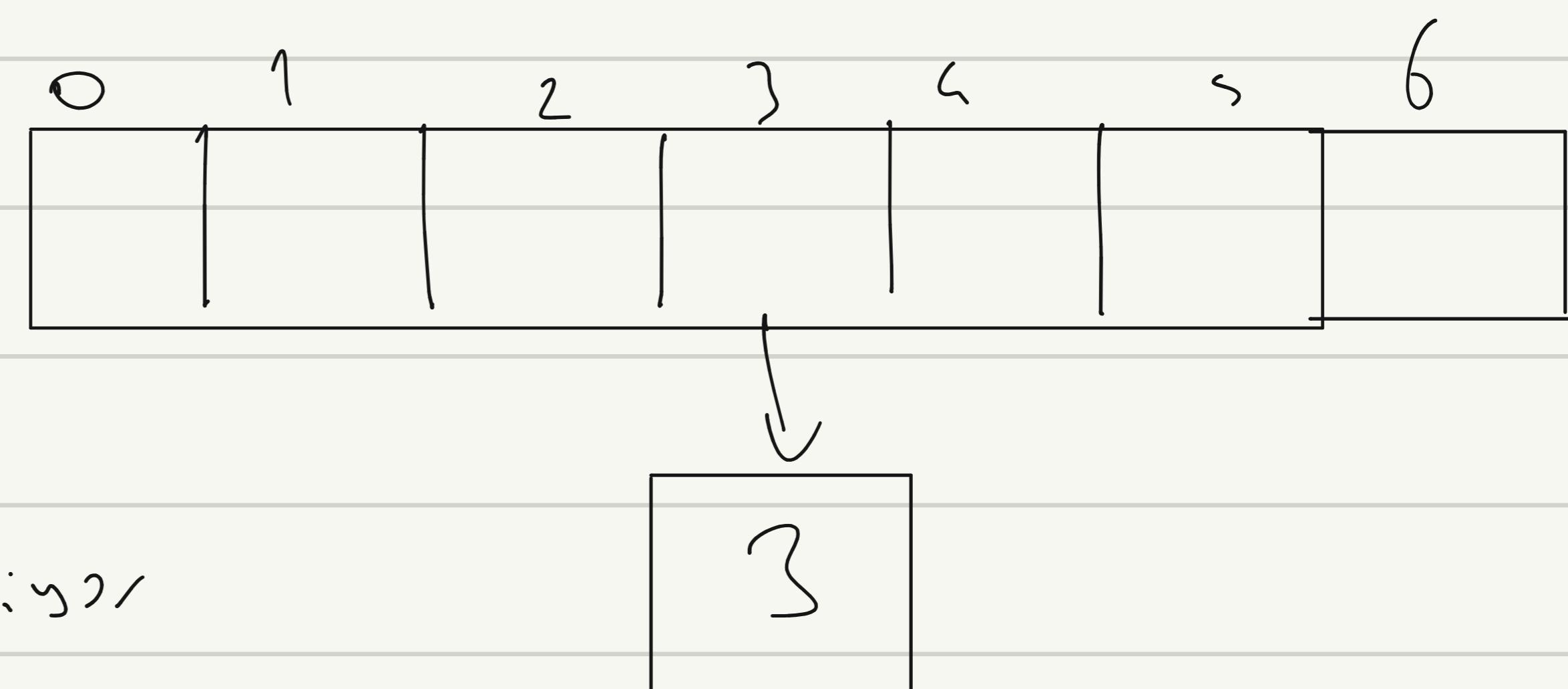
② Separate Chaining



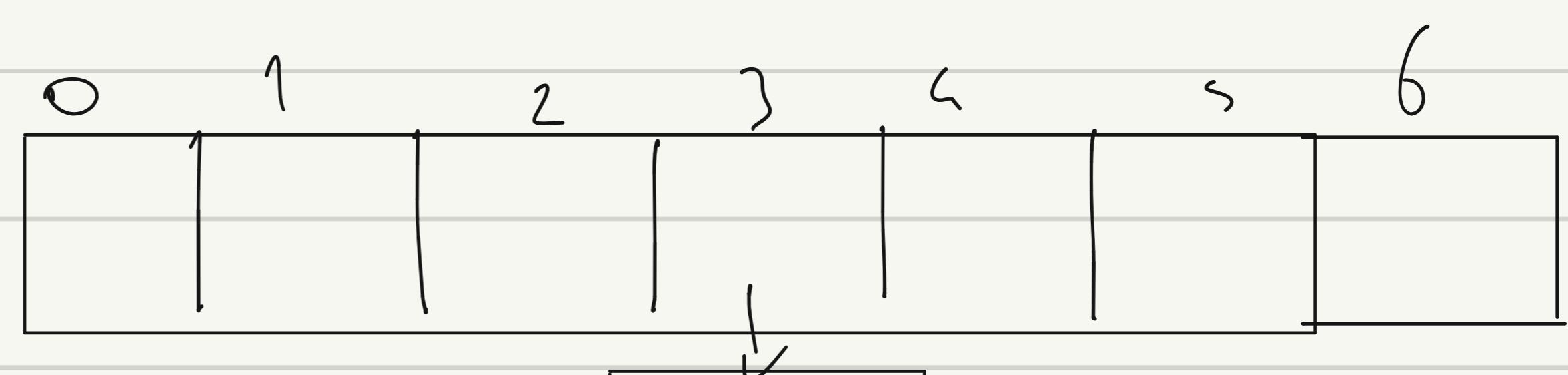
$$M \approx N/5$$

load factor $\approx \% 70$

$$key = 3 \quad 100/7 = 14$$



$$key = 10 \quad 100/7 = 14$$



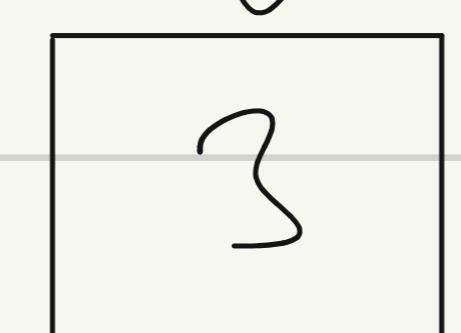
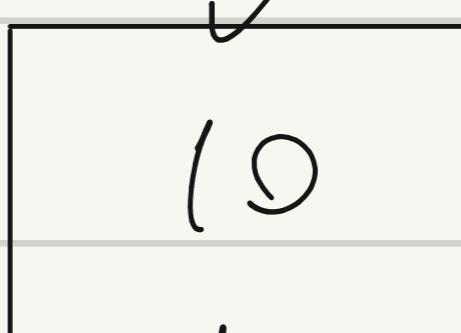
Struct node {

int key;

char name[50];

struct node *next;

}



Keş unique olmali,

① keys strings

"Kayra" hash("kayra")

70 10605010

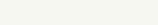
L, L, L, L, L Toplau 200 olur amar argak r. - du degeri 200 olur.

torch.h edit: 1~eq

٦٢

`Length = l` \Rightarrow $k \leftarrow ser[0] + k^{\leftarrow 1} ser[1] \dots k^{\leftarrow l} ser[l]$ i.e. `lijzeriz`.

k asu! ve kück olma!

 Horner's method $b = 31$

hashkey = 0

`length = scrLen(s);`

```
for (i=0; i<length; i++) {
```

`temp = s[i] - 'A' + 1;`

hashkey = k & hashkey + t^{mp}) }

Universal hashing \rightarrow Saldırımlar fonksiyonları
libreler collision olusturmayan fonksiyonlar.

Birden fazla hash fonksiyonu kullanılır
ve rastgele seçilir, konusunda sınıf
luru analayaraklığı için saldırıları zorla-

hash(key) = $[(ak + b) \bmod p] \bmod m$

Hashtakllosun elman eblenecek, eger varsa eklemegem - yoksa eblegen

algoritmayı tasarla

1

$$i = 0$$

```
while (lastValue != null && lastValue != -1) {
```

$i++$

if (i == m)

```
printf("table %d\n")
```

```
else if(hash[cdr] == NULL)
```

```
hash(addr) = key;  
else  
    print("Element zaten var");
```

LAB 2

$$\textcircled{1} \quad \sum_{i=1}^n \sum_{j=1}^n i:j \rightarrow \sum_{i=1}^n \sum_{j=1}^n i \rightarrow \sum_{i=1}^n i \left(\frac{n(n+1)}{2} \right) \rightarrow \left(\frac{n(n+1)}{2} \right)^2$$

$$\textcircled{2} \quad \sum_{i=1}^n i^3 \rightarrow T(N) = T(N-1) + 3 \quad \left. \begin{array}{l} T(N-1) = T(N-2) + 3 \\ T(N-2) = T(N-3) + 3 \\ T(1) = 1 \end{array} \right\} \quad \begin{array}{l} T(N) = T(n-i) + 3 \quad i=1 \\ T(N) = 1 \quad i=n-1 \end{array}$$

$T(N) = T(1) + 3(n-1) \rightarrow 3n-2$

Backward

forward

| n | $T(n)$ | |
|-----|------------|--|
| 1 | 1 | |
| 2 | 4 | |
| 3 | 7 | |
| 4 | 10 | |
| ⋮ | ⋮ | |
| N | $T(n-1)+3$ | |

$\textcircled{3} \quad Q(n)$

```
if (n=1)
    return 1,
return Q(n-1) + 2 * n - 1
```

$$n^2 \text{ algorithm} \rightarrow \text{Backward}$$

$$T(n) = T(n-1) + 2$$

$$T(n) = T(n-2) + 2 + 2$$

$$T(n) = T(n-1) + 2$$

$$T(n-1) = T(n-2) + 2$$

$$\vdots$$

$$\rightarrow \overset{\sim}{\dots} \overset{\sim}{2} \overset{\sim}{n-1}$$

$$T(2) = T(1) + 2 \rightarrow$$

$$T(1) = 1 \rightarrow 1$$

$$T(n) = T(n-i) + 2 \quad i=1$$

$$i=n-1$$

$$T(n) = T(1) + 2 * (n-1)$$

$$= 1 + 2(n-1) = 2n-1$$

4 Selection sort recursive $\rightarrow T(n) = T(n-1) + n$

$$T(n-1) = T(n-2) + n-1$$

$$T(n-2) = T(n-3) + n-2 \rightarrow T(1) = T(n-i) + n - i + 1 + [n-i+2 + n-i+3] \dots n$$

$$T(1) = 1$$

$$T(1) = T(1) + 2 + 3 \dots n \rightarrow n + (n-1)$$

Backward

$$T(n-1) = T(n-2) + n-1$$

$$T(n-2) = T(n-3) + n-2 \quad T(n) = 1 + 2 + 3 \dots n-1 = \sum_{i=1}^n i \rightarrow \frac{n(n+1)}{2}$$

$$T(2) = 1 + 2$$

$$T(1) = 1$$

5 inversion problem

$$\text{dizi}[i] > \text{dizi}[j]$$

bir inversion

ve $i < j$ ise inversion var demektir. Bunu nasıl bulabiliriz?

$$1 \quad 20 \quad 8 \quad 6 \quad | \quad 7 \quad 9 \quad 5 \quad 6 \quad 2$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline & 1 & 20 & | & 8 & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & 1 & | & 20 & | & 8 & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & \uparrow & \\ \hline & 1 & 20 & | & 8 & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline \end{array}$$

Burası kuralı

Diger sayfanın
daha iyi

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & \uparrow & \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & \uparrow & \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline & \uparrow & \\ \hline & 1 & - & 20 & - & 8 & - & 6 & | & 7 & 9 & 5 & 6 & 2 \\ \hline \end{array}$$

\uparrow yine de giderken silmeliyorum

$$1 - 8 \checkmark$$

$$20 - 8 \quad \text{count}++$$

$$20 - 6 \quad \text{count}++$$

Yukarı yukarıda sıralanmış içeri mevcut 5 7 9 ve
 2 6 var. 0 2 5 ile 2 1 2 1 5 7 9 10
 direkt 3 artıyoruz. Çünkü 5 sıfırıysa 7 ve 9'yu sıfırı sonra
 5-6 1 2 3 4 5 6-7 8 9 10 11 12
 1 2 3 4 5 6 7 8 9 10 11 12
 1 2 3 4 5 6 7 8 9 10 11 12

1 2 0 8 6 | 7 9 5 6 2
 1. adım

1 2 0 | 8 6

1 2 0

1 2 0 8 6 | 7 9 5 6 2
 2. adım

1 2 0 | 8 6

count = 0 | 8 6 → count = 1
 not 6 5 4 3 2 1 0
 1 2 0 8 6 | 7 9 5 6 2

1 2 0 | 6 8 → 1 < 6 ✓ 2 > 6 count ++
 count = 0 | count = 1 2 > 8 count ++

1 6 8 2 0 | 7 9 | 5 6 2
 count = 3 7 | 9 5 6 2
 count = 0 | count = 1 7 9

1 6 8 2 0 | 7 9 | 5 6 2
 count = 3 7 | 9 5 6 2
 count = 0 | count = 1 count = 0

| | | | | | | | | | | |
|-----------------------|---|---|----|-----------|-----------|-----------|--|---|---|---|
| 1 | 6 | 8 | 20 | | 7 | 9 | | 5 | 6 | 2 |
| count = 3 | | | | 7 | 9 | 5 | | 6 | | 2 |
| count = 0 count = 1 | | | | count = 0 | count = 0 | count = 0 | | 6 | | 2 |

| | | | | | | | | | | |
|-----------------------|---|---|----|-----------|-----------|-----------|--|-----------|---|---|
| 1 | 6 | 8 | 20 | | 7 | 9 | | 5 | 6 | 2 |
| count = 3 | | | | 7 | 9 | 5 | | 6 | | 2 |
| count = 0 count = 1 | | | | count = 0 | count = 0 | count = 0 | | count = 1 | | |

| | | | | | | | | | | |
|-----------------------|---|---|----|-----------|-----------|-----------|--|-----------|---|---|
| 1 | 6 | 8 | 20 | | 7 | 9 | | 5 | 6 | 2 |
| count = 3 | | | | 7 | 9 | 5 | | 6 | | 2 |
| count = 0 count = 1 | | | | count = 0 | count = 0 | count = 0 | | count = 1 | | |

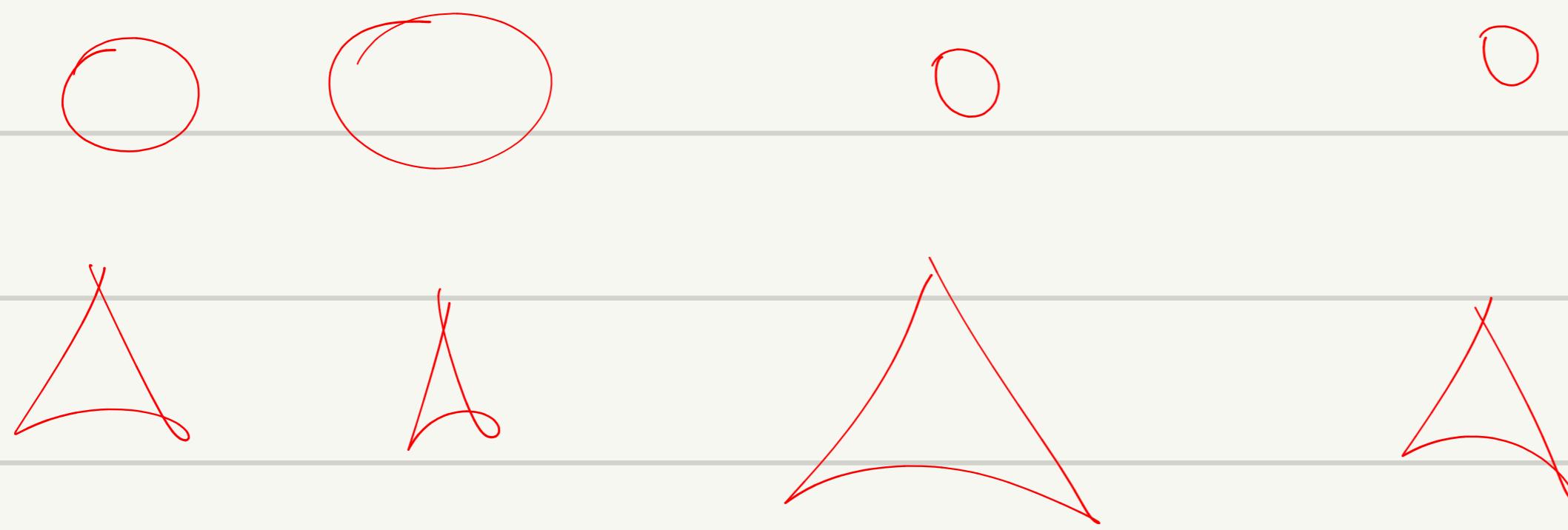
| | | | | | | | | | | |
|-----------------------|---|---|----|-----------|-----------|-----------|---|-------|-------------------|---|
| 1 | 6 | 8 | 20 | | 7 | 9 | | 2 | 5 | 6 |
| count = 3 | | | | count = 0 | count = 2 | count = 2 | → | 7 - 2 | count = count + 2 | |
| count = 0 count = 1 | | | | | | | | 7 - 5 | count = count + 2 | |

| | | | | | | | | | | |
|-----------|---|---|----|--|-----------|-----------|-----------|--------|-------------------|----|
| 1 | 6 | 8 | 20 | | 2 | 5 | 6 | 7 | 9 | |
| count = 3 | | | | | count = 2 | count = 2 | count = 2 | → | 1 - 2 | ✓ |
| | | | | | | | | 6 - 2 | count = count + 2 | |
| | | | | | | | | 6 - 5 | count = count + 2 | |
| | | | | | | | | 6 - 6 | ✓ | |
| | | | | | | | | 8 - 6 | count = count + 2 | |
| | | | | | | | | 8 - 7 | .. | .. |
| | | | | | | | | 8 - 9 | ✓ | |
| | | | | | | | | 20 - 9 | count + 1 | |

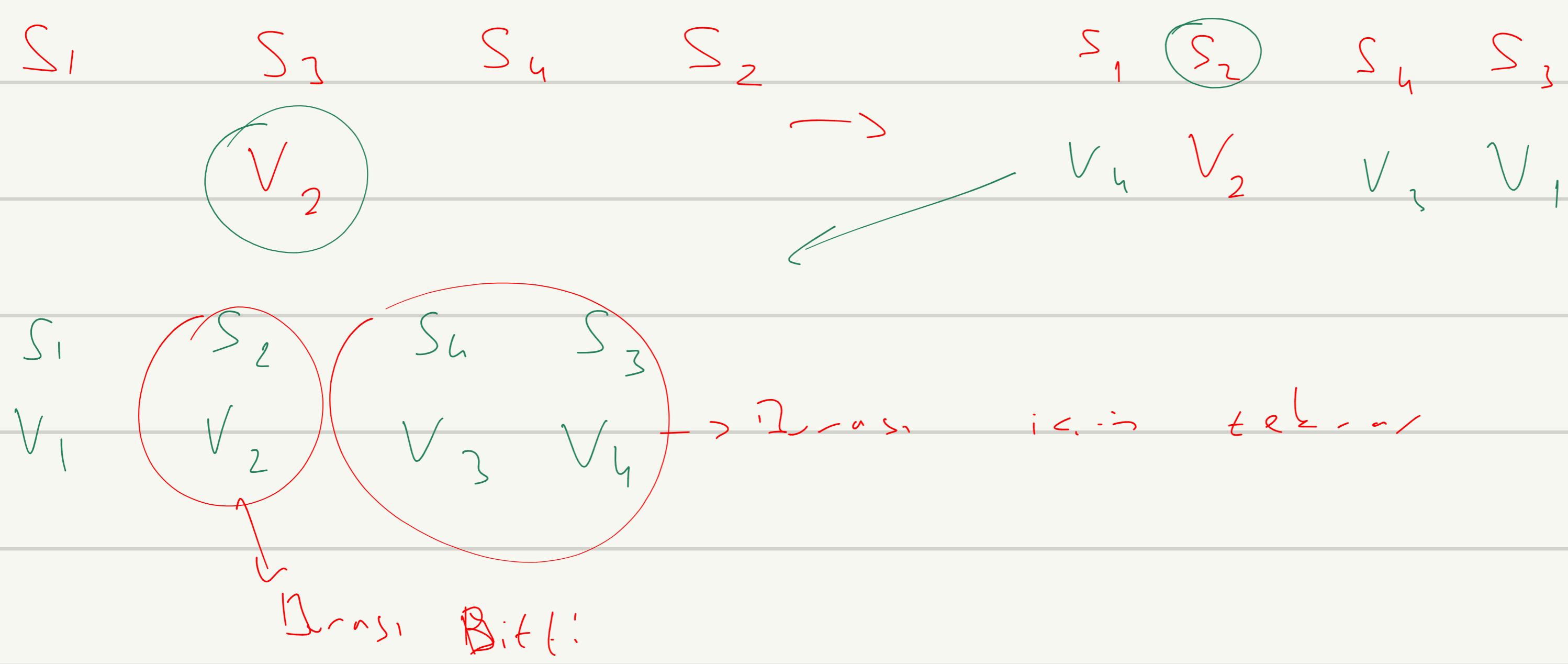
$$10 + 11 = 22$$

⑥ Vidur Sonra

→ hangi vidur hangi sonrada



vidaları arasında kotaslamak yet. sonruları da
burada bir vidur pivot seçeriz ve Quick sort gibi
yaparız.
↳ sonrular için viduları vidular için

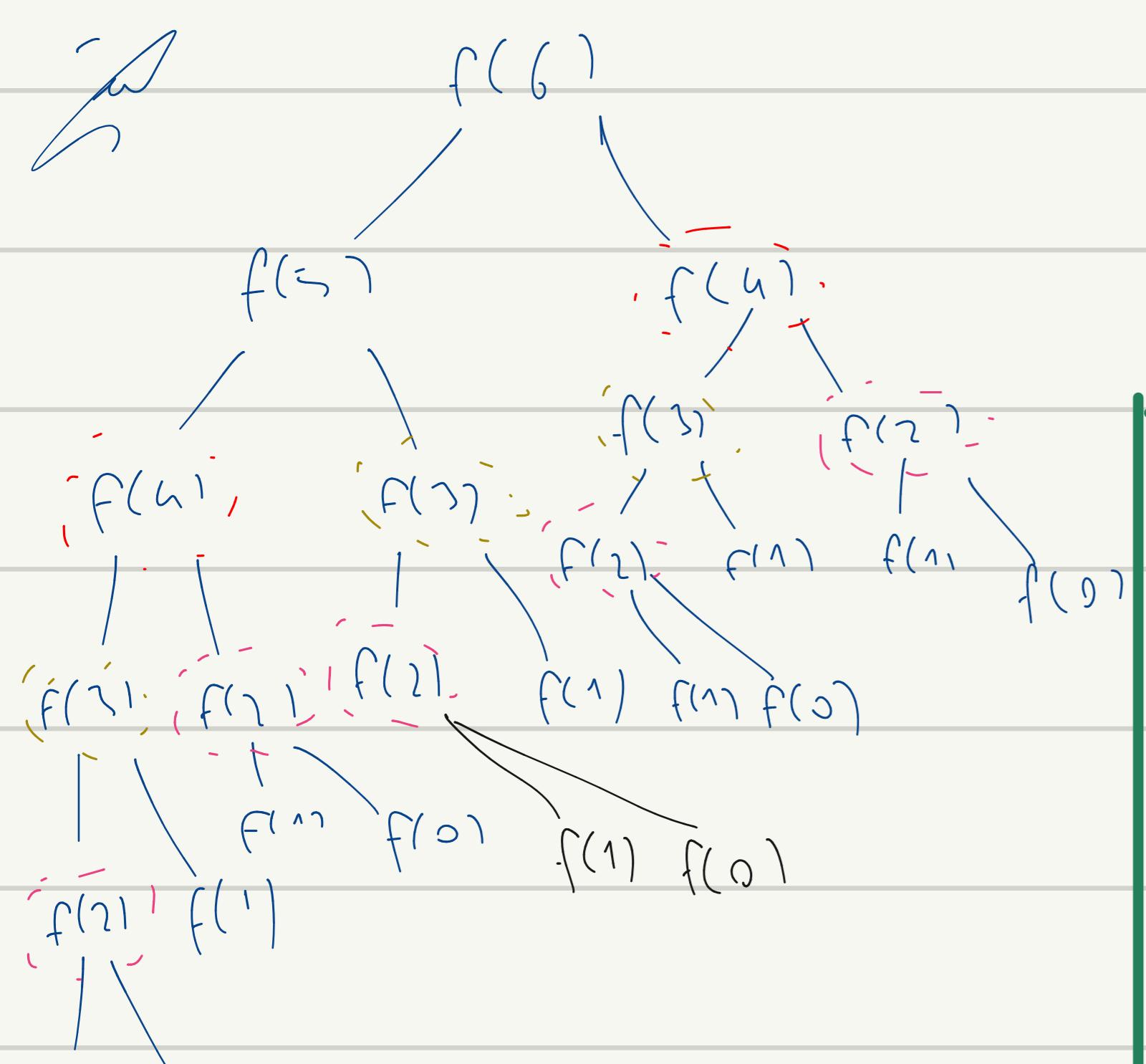


Hofsta 9

Dynamic Programming

Fibonacci numbers

0 1 1 2 3 5 ...



int fibonacci(int n)

```
{
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

Minimum coin change problem

coins = {1, 2, 5} value = 11 coin Answer = 5+5+1

sol 1 $1+1+1+\dots+1$

$1+1+1\dots+2+2$

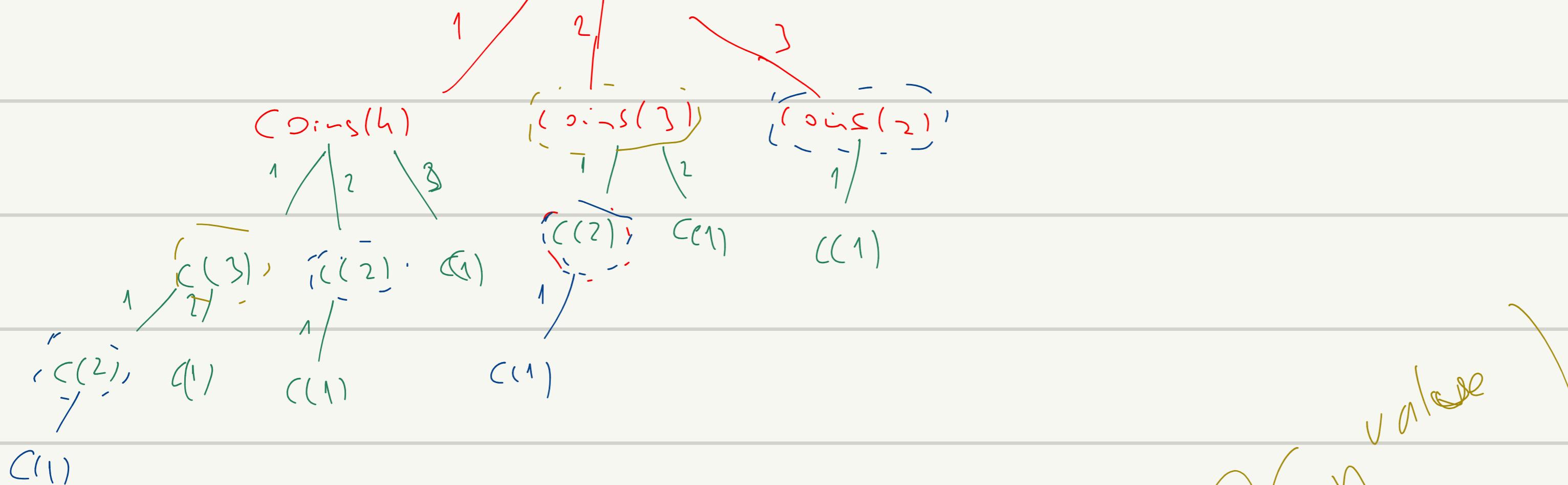
\vdots

\rightarrow Exponential answer

$5+5+1 \rightarrow$ Exhaustive Search

Bottom up doable $\rightarrow P(11) = P(10) + 1$ g.b.i

$C_{coins}(5) \rightarrow C_{coins}[1, 2, 3]$



```

int minCoin(int* coins, int n, int value&{
    int minCount = MAX;
    if(value == 0 &&
       return 0;
    for(i=0;i<n;i++&{
        if(coins[i]<=value &&
           tmp = minCoin(coins,n,value-coins[i]&;
           if(tmp<MAX && tmp+1<minCount &&
              minCount=tmp+1;})
    if(minCount == MAX &&
       return -1;
    else
        return minCount;
    }
}
```

Dynamic Programming

memoization

$\text{coins} = \{1, 2, 5\}$ value = 11 Answer = $5 + 5 + 1$

| 0 | 1 | 2 | 3 | 4 | 5 | → Amount |
|----------|----------|----------|----------|----------|----------|----------|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| 6 | 7 | 8 | 9 | 10 | 11 | → Amount |

↓↓↓↓↓
↓↓↓↓↓
↓↓↓↓↓
↓↓↓↓↓
↓↓↓↓↓

| 0 | 1 | 2 | 3 | 4 | 5 | → Amount |
|---|---|---|---|----|----|----------|
| 0 | 1 | 2 | 3 | 4 | 5 | ∞ |
| 6 | 7 | 8 | 9 | 10 | 11 | ∞ |

1 TL $\min(0+1, \infty)$ $\rightarrow \min(1+1, \infty)$ $\rightarrow ?$

| 0 | 1 | 2 | 3 | 4 | 5 | → Amount |
|---|---|---|---|----|----|----------|
| 0 | 1 | 1 | 2 | 2 | 3 | ∞ |
| 3 | 4 | 4 | 5 | 5 | 6 | ∞ |
| 6 | 7 | 8 | 9 | 10 | 11 | ∞ |

5 < 1 $\min(0+1, 3) \rightarrow ?$

| 0 | 1 | 2 | 3 | 4 | 5 | → Amount |
|---|---|---|---|----|----|----------|
| 0 | 1 | 1 | 2 | 2 | 1 | ∞ |
| 2 | 2 | 3 | 3 | 2 | 2 | ∞ |
| 6 | 7 | 8 | 9 | 10 | 11 | ∞ |

$$\min(1+1, 3) \rightarrow \min(1+1+1, 3) \rightarrow \min(1+1+1+1, 3) \rightarrow \min(1+1+1+1+1, 3) \rightarrow ?$$

$$d[2][6] = \min(d[2][5]+d[2][1], d[2][6])$$

$$d[2][7] = \min(d[2][6]+d[2][1], d[2][7])$$

$C[P]$: minimum number of coins

x : first coin

$C[P] = 1 + C[P-x]$

$$C[P] = \begin{cases} \min\{C[P-1] + 1 : d_i < P\} \\ 0 \quad \text{if } P=0 \end{cases}$$

$$C[8] = \min\{C[7] + 1, C[6] + 1, C[5] + 1\}$$

$$C[0] = 0$$

for $p=1$ to n do {

$$\min = \infty$$

for $i=1$ to k do {

if $\mathcal{L}[i] \leq p$ then

$\min = 1 + C[p - \mathcal{L}[i]]$

$$C[p] = \min$$

$O(M * k)$

3

coin says

{1, 5, 2}

}

Knapsack Problem

subset sum problem

abandon
labor

select

$W = \max \text{ weight}$
 $N = \text{number of items}$
 $i^{\text{th}} \text{ item} \rightarrow \text{value, weight}$

maximize $\sum v_i$ subject to w

| item | 1 | 2 | 3 |
|------|---|---|---|
| | | | |
| | | | |
| | | | |

maximum profit

from item 1 to item i

from weight 1 to weight j

| item | 1 | 2 | value | $w = 5 \text{ kg}$ |
|------|---|---|-------|--------------------|
| 1 | | | 12 | |
| 2 | | 1 | 10 | |
| 3 | | 3 | 20 | |
| 4 | | 2 | 15 | |

| item | 1 | 2 | value | $w = 5 \text{ kg}$ |
|------|---|---|-------|--------------------|
| 1 | | | 12 | |
| 2 | | 1 | 10 | |
| 3 | | 3 | 20 | |
| 4 | | 2 | 15 | |

0 kilo 2 kilo exceeding item

weight

| | | 0 | 1 | 2 | 3 | 4 | 5 |
|------|---|---|----|----|----|----|----|
| | | 0 | 0 | 0 | 0 | 0 | 0 |
| item | 1 | 0 | 0 | 12 | 12 | 12 | 12 |
| | | 1 | 0 | 12 | 12 | 12 | 12 |
| 2 | 2 | 0 | 10 | 12 | 22 | 22 | 22 |
| | | 1 | 10 | 12 | 22 | 30 | 32 |
| 3 | 3 | 0 | 10 | 12 | 22 | 30 | 32 |
| | | 1 | 10 | 12 | 22 | 30 | 37 |
| 4 | 4 | 0 | 10 | 15 | 25 | 30 | 37 |
| | | 1 | 10 | 15 | 25 | 30 | 37 |

Case 1: thief takes item i

item

Case 2: thief doesn't take item i

Rekürans haryatı

$$P(i, w) = P(i-1, w)$$

not include
item

Before item i

10. Haufta

O-1 Knap sack Problem

$W = 11$ item → value
item → weight

item value weight

1 1 1

2 6 2

3 18 5

4 22 6

5 28 7

item weight
 \downarrow $\begin{cases} Q \leq \text{value} \\ P_{\text{opt}} \end{cases}$

$P[i, w]$

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 3 | 0 | 1 | 6 | 7 | 7 | 18 | 19 | 24 | 25 | 25 | 25 | 25 |
| 4 | 0 | 1 | 6 | 7 | 7 | 18 | 22 | 24 | 28 | 29 | 29 | 40 |
| 5 | 0 | 1 | 6 | 7 | 7 | 18 | 22 | 28 | 29 | 34 | 35 | 40 |

Case 1: take item :

$$P[i, w] = P[i-1, w - w_i] + \text{value}_i$$

\downarrow now weight of item

Alnigilirini aldm?

Case 2: not take item:

$$P[i, w] = P[i-1, w]$$

\downarrow $w_i > w$

$\rightarrow P[5, 3] = P[4, 3] \rightarrow \text{weight}[5] > 3$

Take $\max \{v_i + P[i-1, w-w_i], P[i-1, w]\}$

$$P[5, 11] = P[4, 11]$$

almdam

$$\text{else aldm } P[4, 11 - k_3] = P[3, 11 - k_3]$$

$O(n \cdot w)$

↳ top. arg'lz

Recurrence Relation

$$P[i, w] = \begin{cases} P[i, w] = 0 & \text{if } i=0 \text{ or } w=0 \\ P[i-1, w] & \text{if } w_i > w \text{ not taken} \\ \max(v_i + P[i-1, w-w_i], P[i-1, w]) & \text{otherwise} \end{cases}$$

↳ satz

↳ sätz

↳ kazanç

↳ alısmam

↳ halmazsam

Edit Distance

① graffe

graph, graft, grain, giraffe

② Computational Biology

AC GTGGA

AGTAGAA

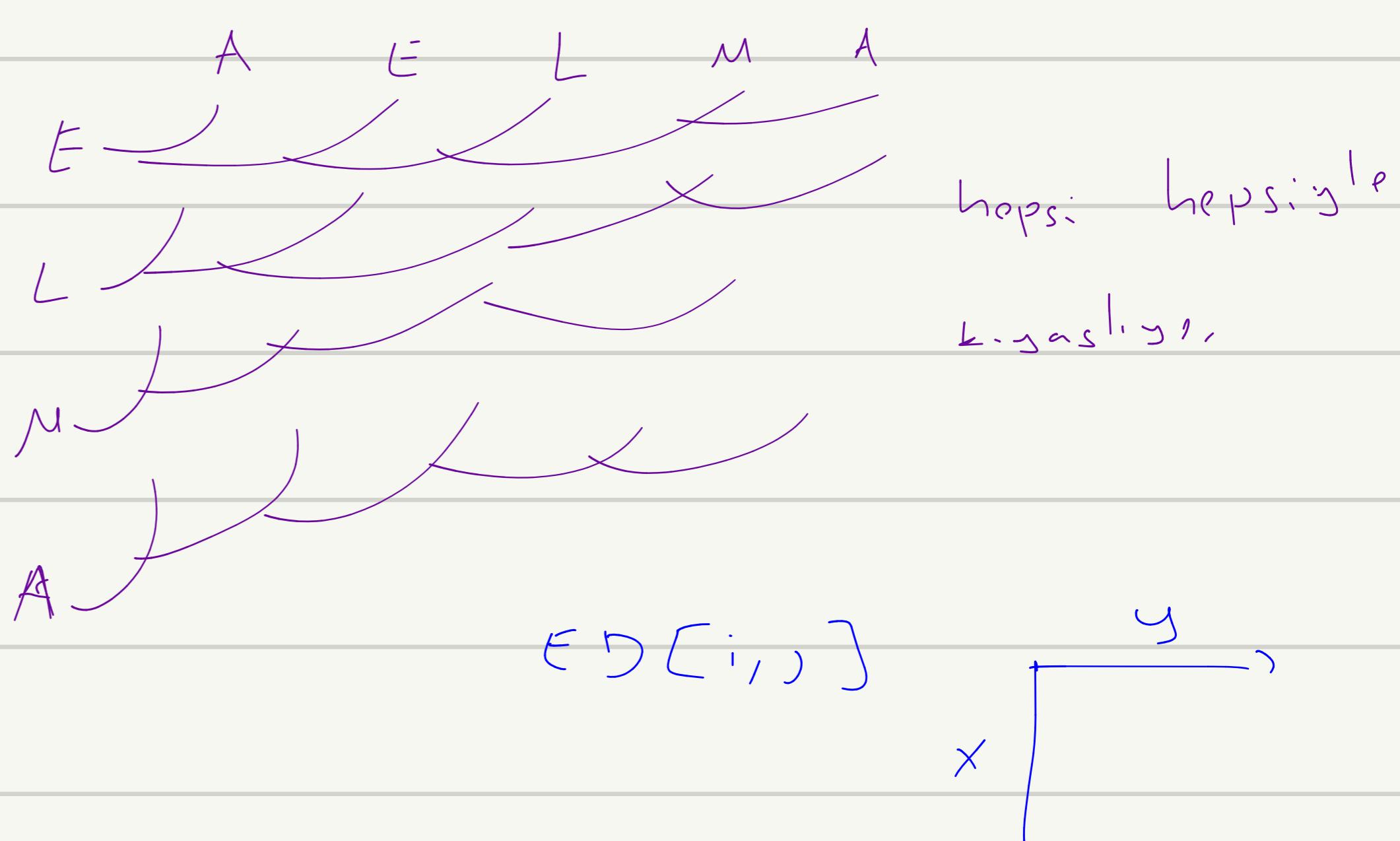
S N T E * N T I O N
 ↓ ↓ X E C U T I O N
 S! dog's eld. degis

top-left distance $\rightarrow 5$

(a) Substitute $\begin{cases} \text{same} \\ \text{change} \end{cases}$ $\text{change}(x_i, y_j, a) \rightarrow x_1 \dots x_{i-1} a x_{i+1} \dots n$
 $x_i = y_j \rightarrow ED[i, j] = ED[i, i]$
 $x_i \neq y_j \rightarrow ED[i, j] = ED[i-1, j-1]$

(b) Delete: $x_1, x_2 \dots x_{i-1}, x_i, \dots x_n$
 $x_i \neq y_j \quad ED[i, j] = ED[i-1, j-1]$

(c) Insert: $x_1, x_2 \dots x_{i-1}, a, x_{i+1} \dots x_n$
 $ED[i, j] = ED[i, j-1] + 1$



Rekurrenz Beziehung

$$ED[i, j] = \min(ED[i-1, j] + 1, ED[i, j-1] + 1, ED[i-1, j-1] + 1)$$

↳ eiger fall für a

$$\min \left\{ \begin{array}{l} ED[i-1][j] + 1 \quad \uparrow \text{deletion} \\ ED[i][j-1] + 1 \quad \leftarrow \text{insertion} \\ ED[i-1][j-1] + \text{cost} \rightarrow x_i = y_j \quad \text{cost} = 0 \\ \quad \quad \quad \text{else} \quad \text{cost} = 1 \end{array} \right. \quad \text{Substitution}$$

$x = ARTS$

$y = MATHS$

| | A | R | T | S | |
|---|---|---|---|---|---|
| O | 1 | 2 | 3 | 4 | |
| M | 1 | 1 | 2 | 3 | 4 |
| A | 2 | 1 | 2 | 3 | 4 |
| T | 3 | 2 | 2 | 2 | 3 |
| H | 4 | 3 | 3 | 3 | 3 |
| S | 5 | 4 | 4 | 4 | 3 |

$\mathcal{O}(N+k) \approx \mathcal{O}(N^2)$

$ART \neq S$
 $MATHS$ testen
gleichl.

Largest Common Subsequence

subsequence of a word

common subsequence

yakamoz

↓

yak

ya²

yakamoz

yo²gaz

yo²

ya²

Brute force $\rightarrow \mathcal{O}(n! 2^m)$

$$\begin{array}{l} X: ATCTGAT \\ y: TGCATA \end{array} \left\{ \begin{array}{l} LCS(x, y) = TCTA \\ \text{or} \\ TGAT \end{array} \right.$$

$LCS[i, j]$ $x[i:j]$ vs $y[i:j]$ for longest common subsequence

Case 1: $x_i = y_j$ $LCS[i, j] = LCS[i-1, j-1] + 1$

$\cancel{x}: ACGT$ $\cancel{y}: GCT$ $\rightarrow LCS[i, j] = 1$

Case 2: $x_i \neq y_j$ $\left[\begin{array}{l} x_i \text{ is not part of } s \\ y_j \text{ " " " " } \end{array} \right] \Rightarrow LCS[i, j] = \max(LCS[i-1, j], LCS[i, j-1])$

$$LCS[i, j] = \begin{cases} 0 & \text{if } i=0, j=0 \\ \max(LCS[i-1, j], LCS[i, j-1]) + 1 & \text{if } i>0, j>0 \text{ and } x_i = y_j \end{cases}$$

$$\max(LCS[i-1, j], LCS[i, j-1]) \quad \boxed{i>0, j>0} \quad \boxed{x_i \neq y_j}$$

$X = BACD\bar{B}$ $y = BDC\bar{B}$

| | | | | | |
|-----------|---|---|---|---|-----------|
| | B | A | C | D | \bar{B} |
| B | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 1 | 1 | 1 |
| C | 0 | 1 | 1 | 2 | 2 |
| D | 0 | 1 | 2 | 2 | 2 |
| \bar{B} | 0 | 1 | 2 | 2 | 3 |

diagoneller

algorithm

$\bar{B} \subset B$

11. HAFTA

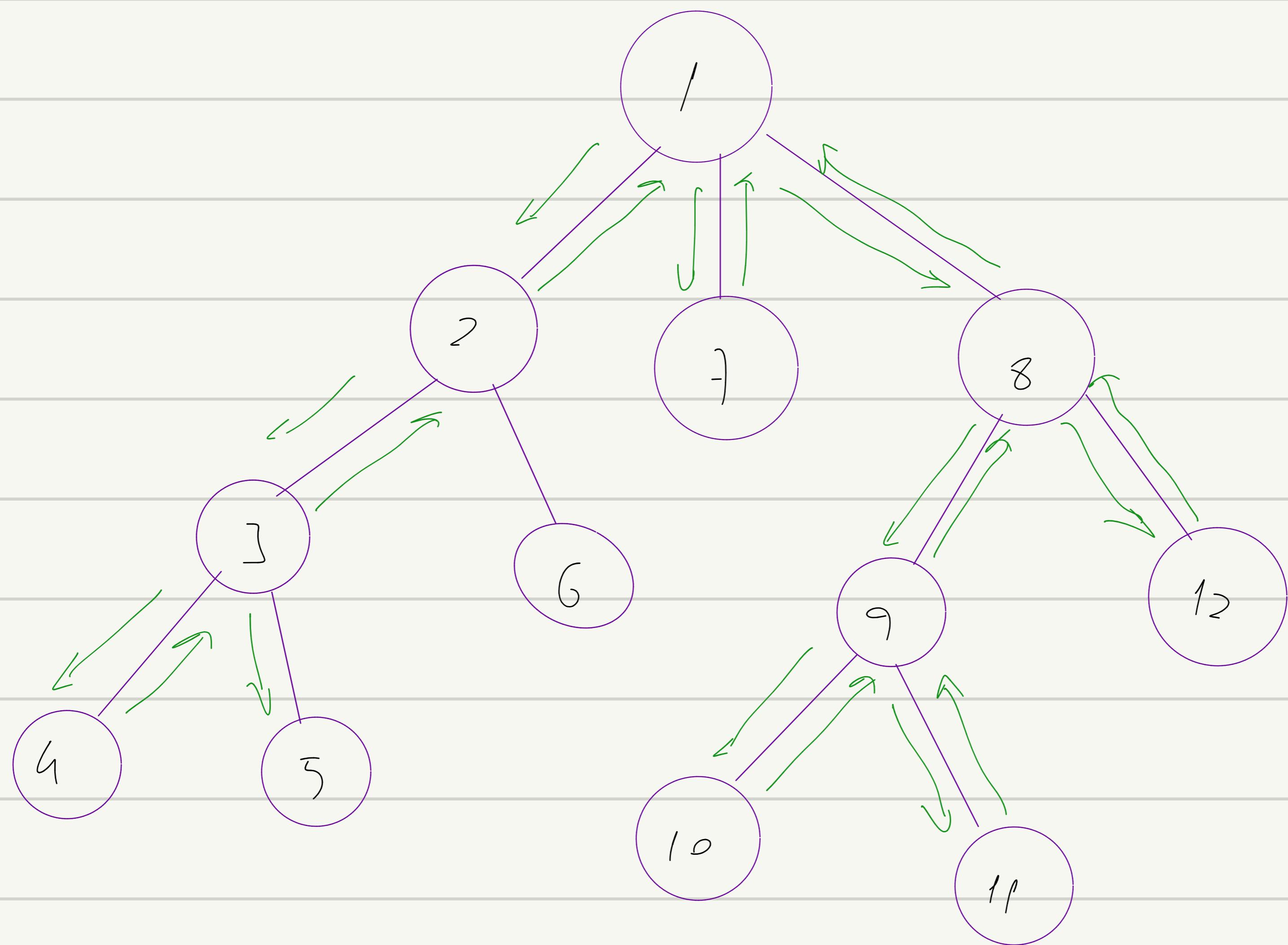
Graph traversal

↳ komşuluğunu matrisi:
 ↳ komşuluğunu listesi:

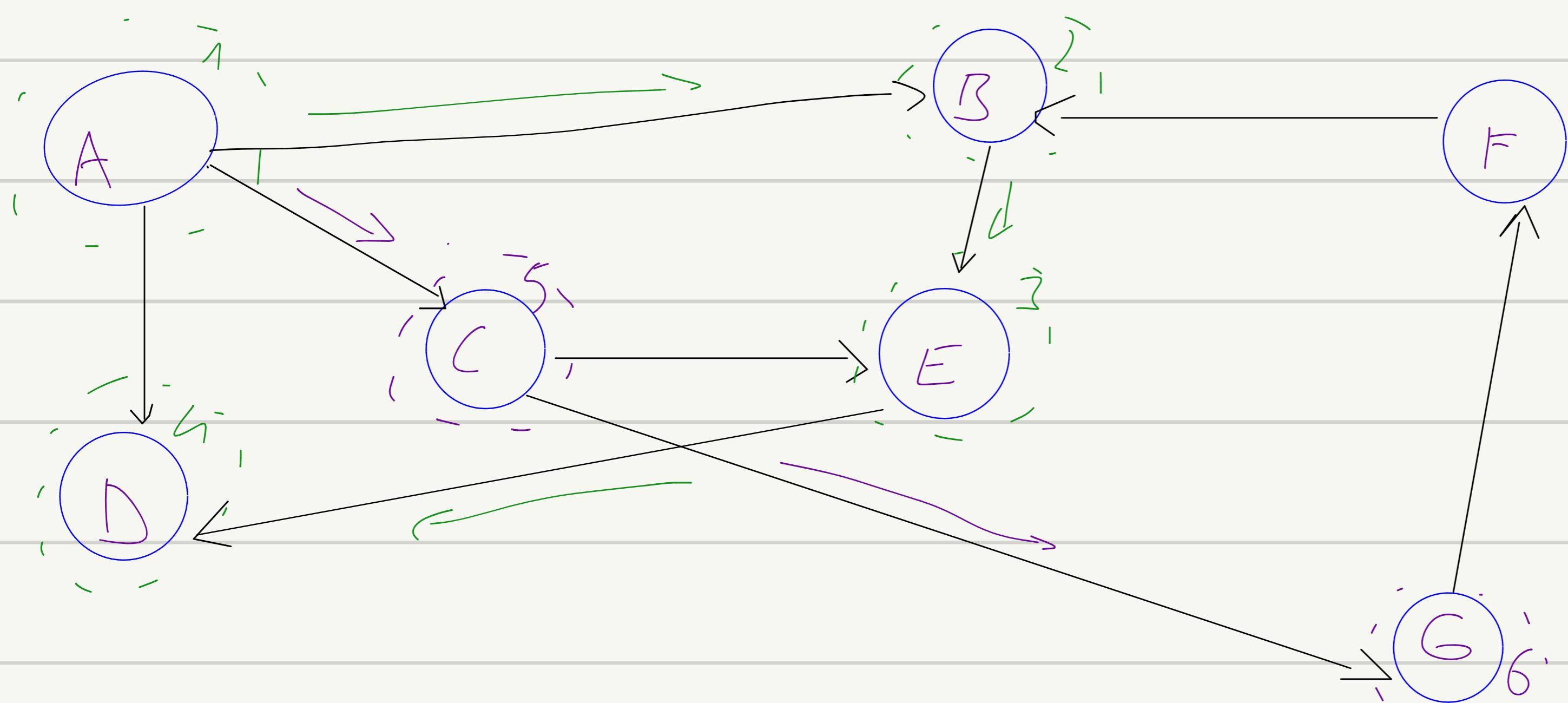
DFS → Queue

DFS

DFS → STACK



A'dan G'ye gol var mı?



| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ |

visited

Recursive DFS

$\text{DFS}(G, S) \left\{ \begin{array}{l} \xrightarrow{\text{Graph}} \text{initial point} \\ \text{mark } s \text{ as visited} \end{array} \right.$

mark s as visited

for all neighbours w of s in graph G

if w is not visited

$\text{DFS}(G, w) \}$

Iterative DFS

$\text{DFS}(G, S)$

push(s)

mark s as visited

while stack is not empty

$v = \text{pop}()$

for all neighbours w of v in graph G

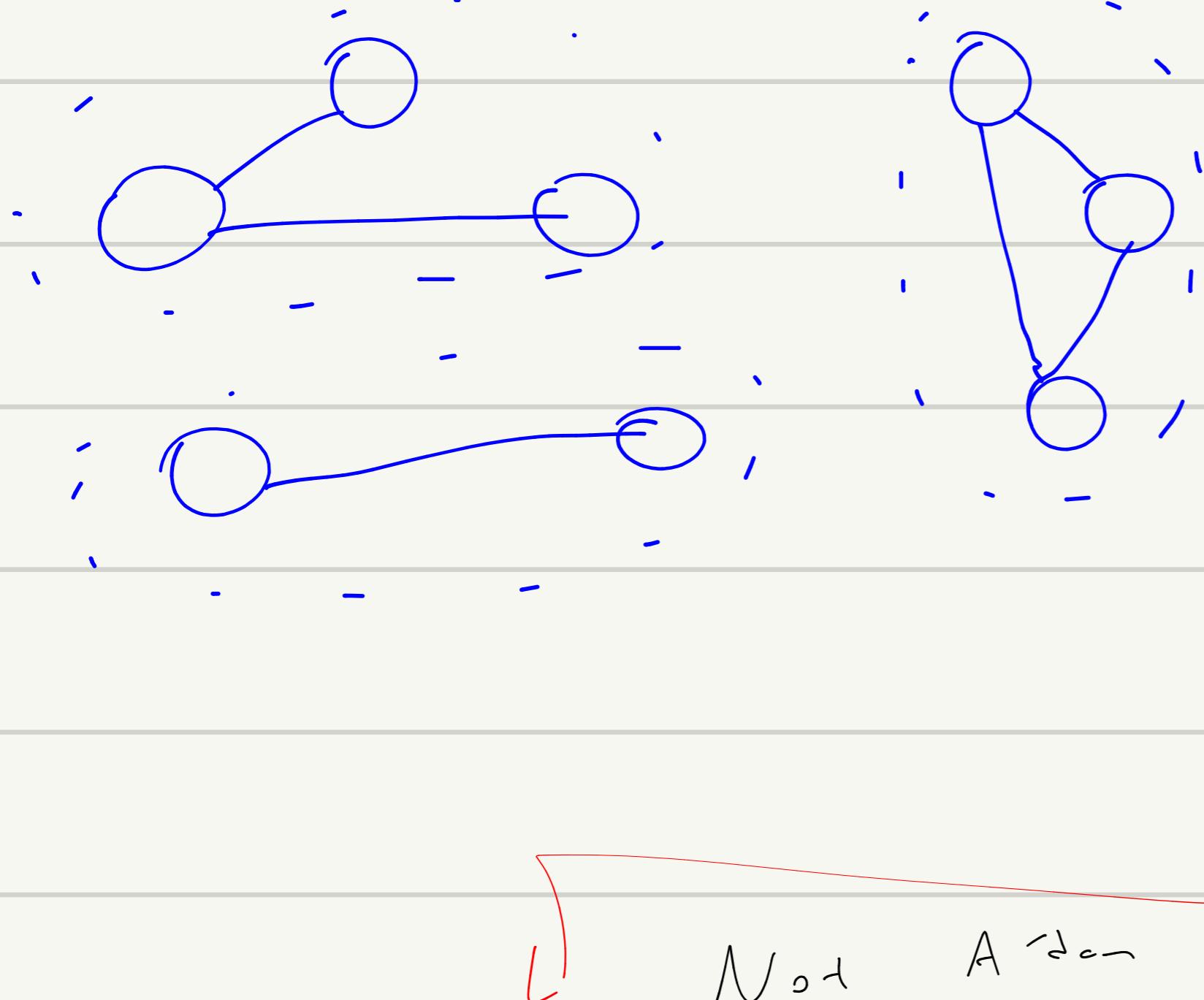
if w is not visited

push w

mark w as visited

Connected Component

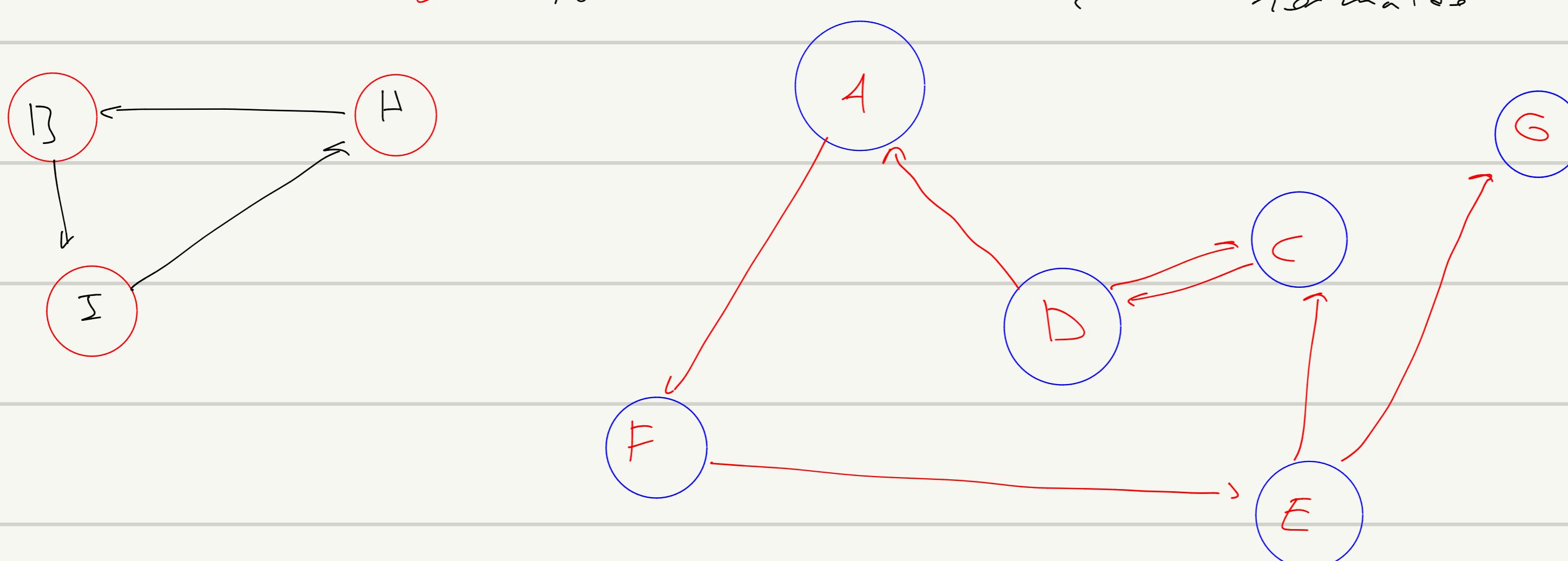
Yonko Graph



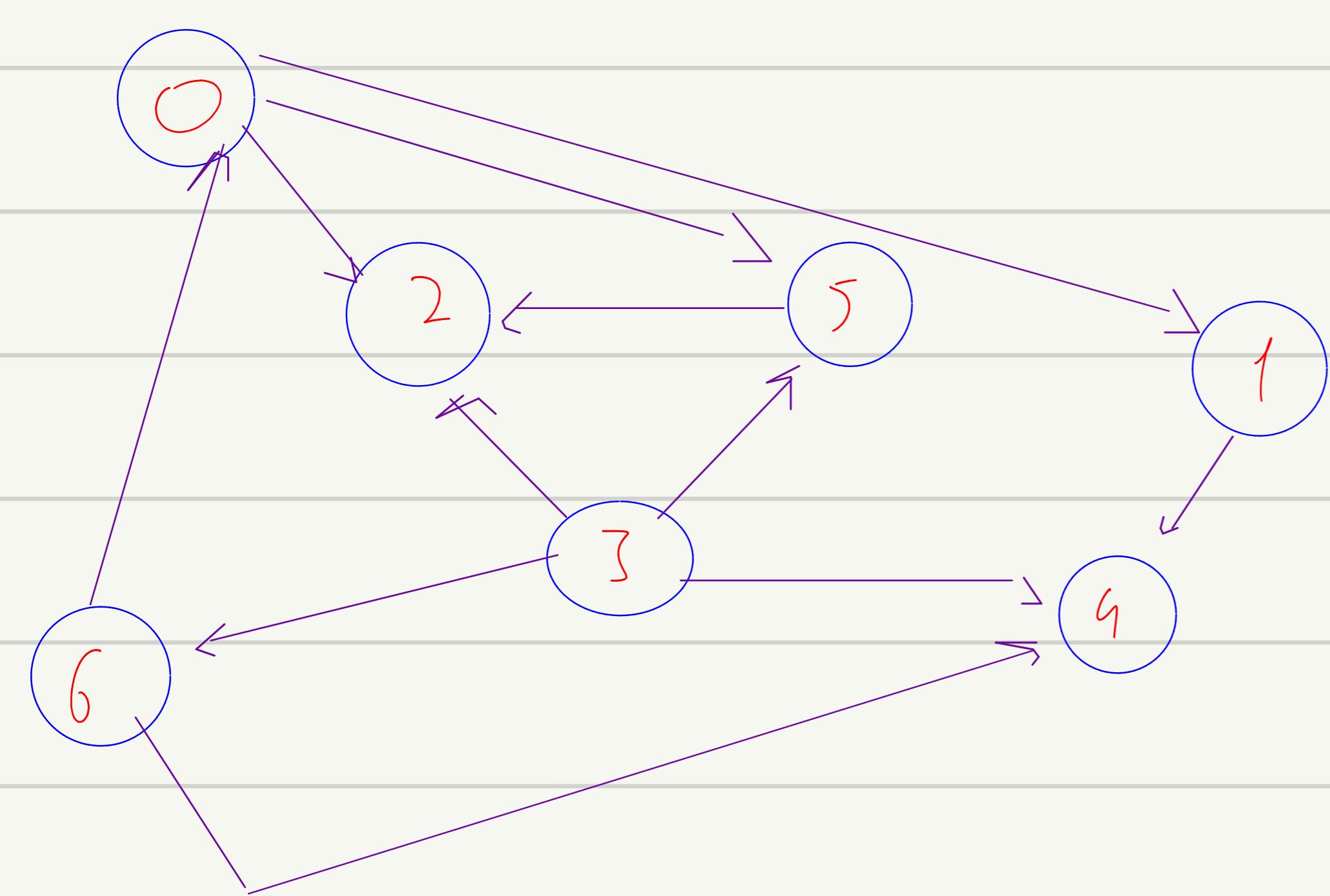
Not A disconnected

normal

component



Topological Sort \rightarrow Acyclic & Directed group



Adjacency List

0 → 1 → 2 → 5

$$1 \rightarrow 6$$

$$2 \rightarrow x$$

3 → 2 → 4 → 5 → 6

↳ X

$$5 \rightarrow 2$$

indegree \rightarrow 0 in 1 (fan)

$$6 \rightarrow 0 \rightarrow 4$$

out degree \rightarrow 0 "] (1, 2, 5)

$\downarrow f \circ (0)$

dfs(1)

$$f \in S(h)$$

h done

1 2 3 4

$\downarrow f s(z)$

2 done

$\text{dfs}(\varsigma)$

check 2

3 done

O done

$\downarrow f s(3)$

C 2

C h

C 5

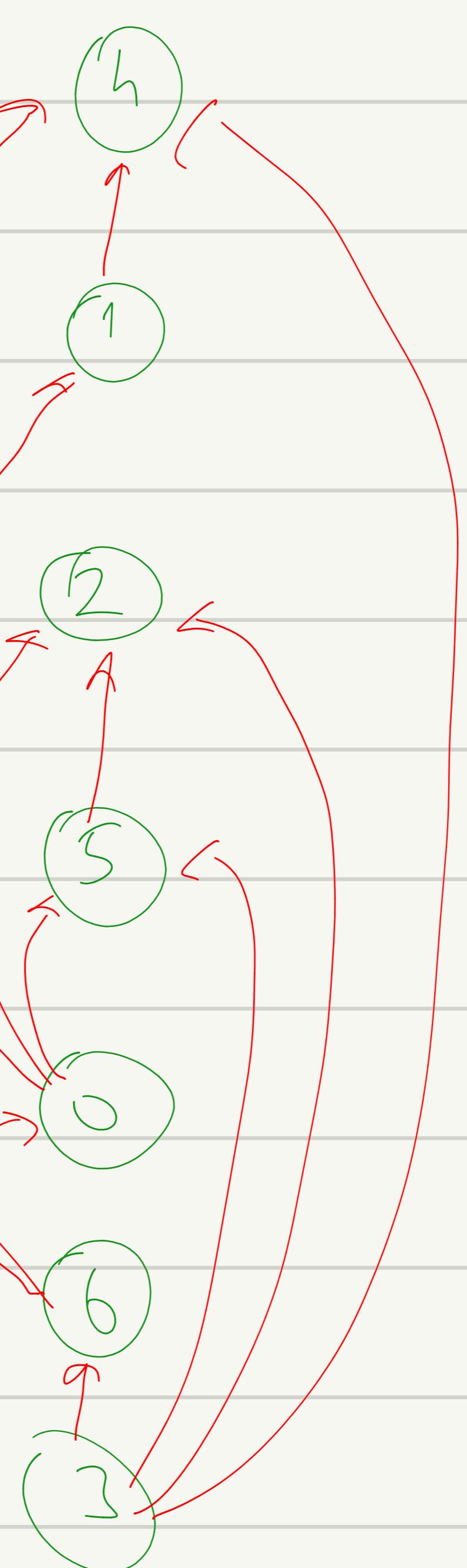
dfs(6)

1

6

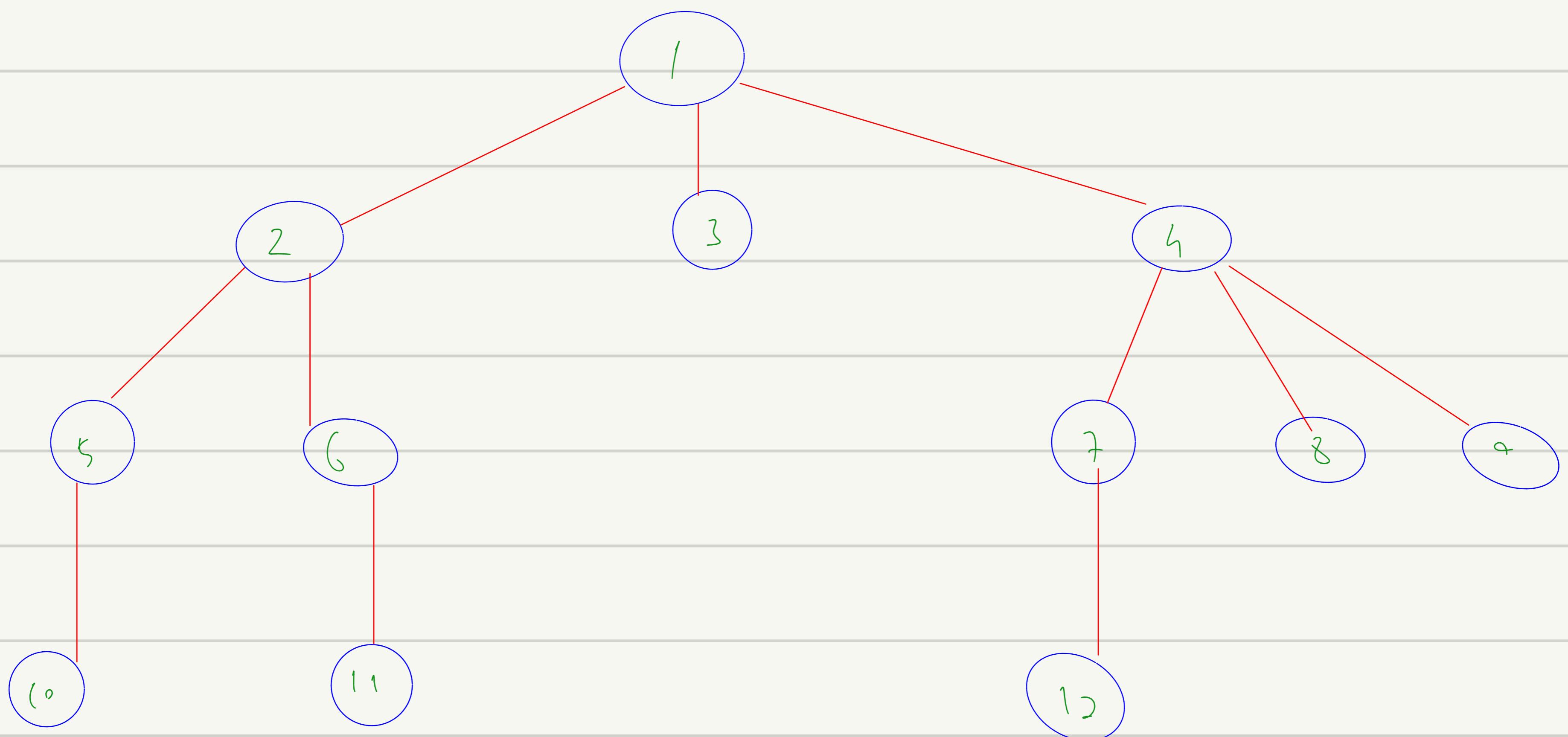
6 do

done



flood fill, cycle detection, topological sort, glib, problem, DFS
like graphs, connected component

BFS $S \rightarrow \text{Queue}$



BFS(G, s) {

enqueue(s)

mark s as visited

while (Queue is not empty)

$v = \text{dequeue}(s)$

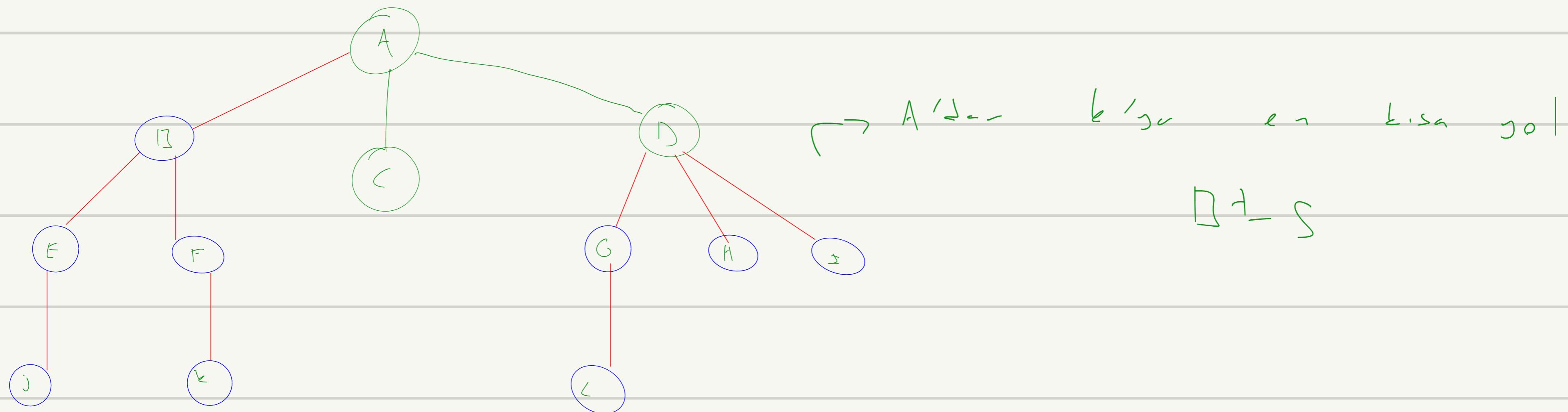
for all neighbours w of v in G

if w isn't visited

enqueue w

mark w as visited

}



| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

→ Visited

| A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 0 |

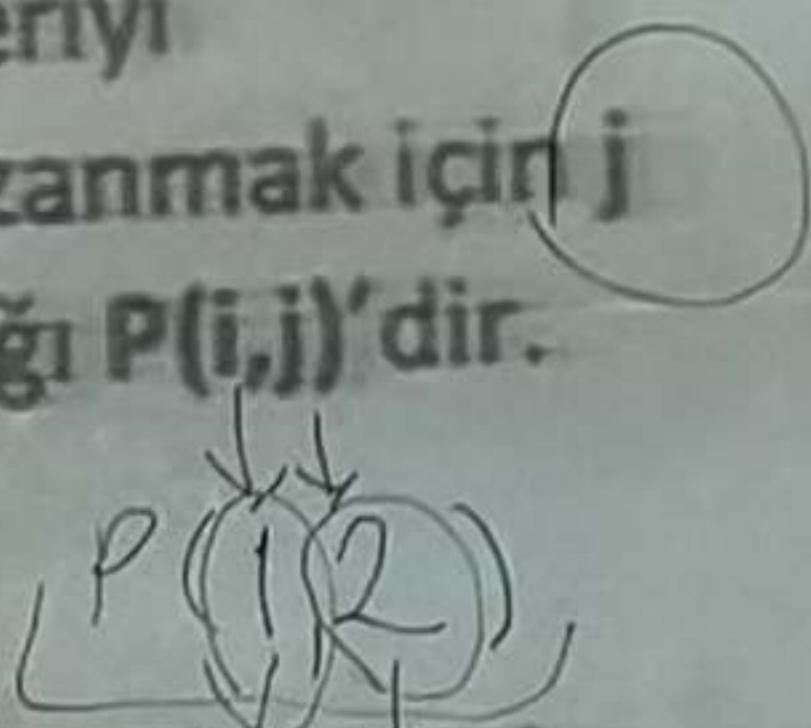
→ Path

Google hierarchy, P2P network, global visualization and BFS traversal.

Dinamik Programlama Örnek:

LAR

A ve B takımları, içlerinden biri n galibiyet alana kadar maç yapacaklardır. A takımının bir maçı kazanma olasılığı her maç için p , kaybetme olasılığı ise $1-p$ dir. Dolayısıyla beraberlik ihtimali mevcut değildir. A takımının seriyi kazanmak için i tane daha maç kazanması, B takımının da seriyi kazanmak için j tane maç kazanması gereken durumda A'nın seriyi kazanma olasılığı $P(i,j)$ 'dir.



- $P(i,j)$ 'yi kullanarak dinamik programlama ile algoritma tasarlayabilemek için rekürans bağıntısını yazınız.
- A takımının bir maçı kazanma olasılığının 0.4 olduğu durumda 7 maçlık seride (4 alan kazanır) A takımının kazanma ihtimalini hesaplayınız. (Dinamik programlama yaklaşımını kullanınız.)

$$P(i,j) = P \leftarrow P(i-1, j) + (1-P) * P(i, j-1) \quad i, j > 0$$

$$P(0, j) = 1 \quad j=0$$

$$P(1, 0) = 0 \quad j=0$$

| | | j | | | | | |
|--|--|-----|--------|---------|---------|----------|---|
| | | 0 | 1 | 2 | 3 | 4 | |
| | | 0 | 0 | 1 | 1 | 1 | 1 |
| | | 1 | 0.4 | 0.64 | 0.784 | 0.8704 | |
| | | 2 | 0.16 | 0.352 | 0.5248 | 0.66304 | |
| | | 3 | 0.064 | 0.1792 | 0.31744 | 0.45568 | |
| | | 4 | 0.0256 | 0.08704 | 0.1792 | 0.289792 | |

$P(\tau)$

for ($i=0$; $i < n$; $i++$)

$\text{dizi}[i][0] = 0$,

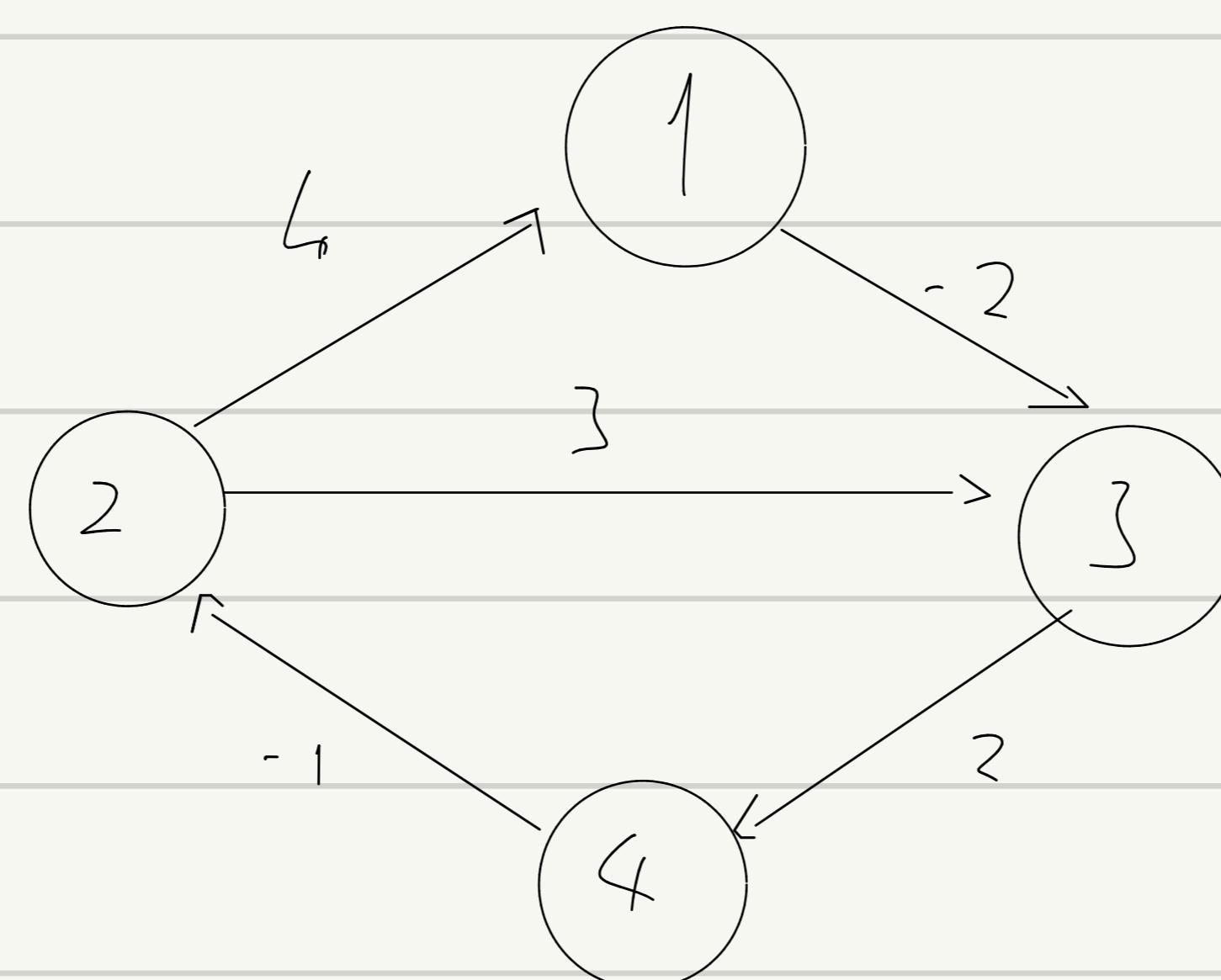
$\text{dizi}[0][i] = 1$,

for ($i=1$; $i < n$; $i++$)

for ($j=0$; $j < n$; $j++$)

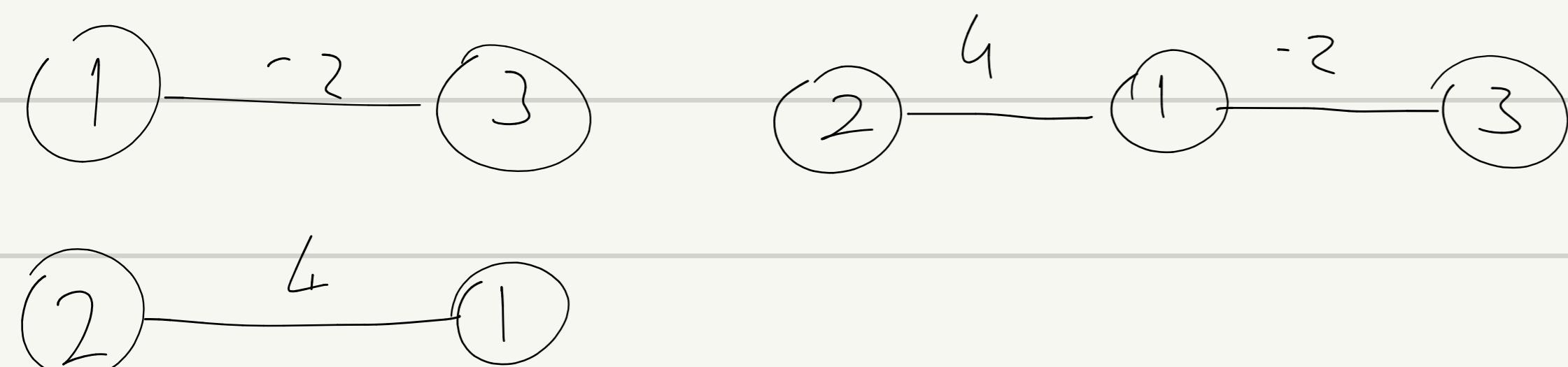
$\text{dizi}[i][j] = \text{dizi}[i-1][j] + P + \text{dizi}[i][j-1] + (1-P)$

Graph - Floyd Warshall



$k = 0$

$k = 1$



$$\min(\text{distance}[i][j], \text{dist}(i, k) + \text{dist}(k, j))$$



$$\begin{aligned} \text{dist}[2,3] &= 3 \\ \text{dist}[2,1] + \text{dist}[1,3] & \end{aligned}$$

| $k=2$ | 1 | 2 | 3 | 4 |
|-------|----------|----------|----------|----------|
| 1 | 0 | ∞ | -2 | ∞ |
| 2 | 4 | 0 | 3 | ∞ |
| 3 | ∞ | ∞ | 0 | 2 |
| 4 | ∞ | -1 | ∞ | 0 |

| $k=0$ | 1 | 2 | 3 | 4 |
|-------|----------|----------|----------|----------|
| 1 | 0 | ∞ | -2 | ∞ |
| 2 | 4 | 0 | 3 | ∞ |
| 3 | ∞ | ∞ | 0 | 2 |
| 4 | ∞ | -1 | ∞ | 0 |

| $k=1$ | 1 | 2 | 3 | 4 |
|-------|----------|----------|----------|----------|
| 1 | 0 | ∞ | -2 | ∞ |
| 2 | 4 | 0 | 3 | ∞ |
| 3 | ∞ | ∞ | 0 | 2 |
| 4 | ∞ | -1 | ∞ | 0 |

$W(\text{graph } L, V, E)$

```

for i=0; i< v; i++
  for j=0; j< v; j++
    matrix[i][j]= $\infty$ 
  
```

```

for each edge (u,v)
  dist[u][v] =  $\infty$ 
  
```

```

for i=0; i< v; i++
  dist[v][v]=0
  
```

```

for k=0; k< v; k++
  for j=0; j< v; j++
    for i=0; i< v; i++
      if( dist[i][j] > dist[i][k] + dist[k][j] )
        dist[i][j] = dist[i][k] + dist[k][j]
  
```

k_y

final

$S_L g$

ϕ 1
 1 2
 2 3
 X 4

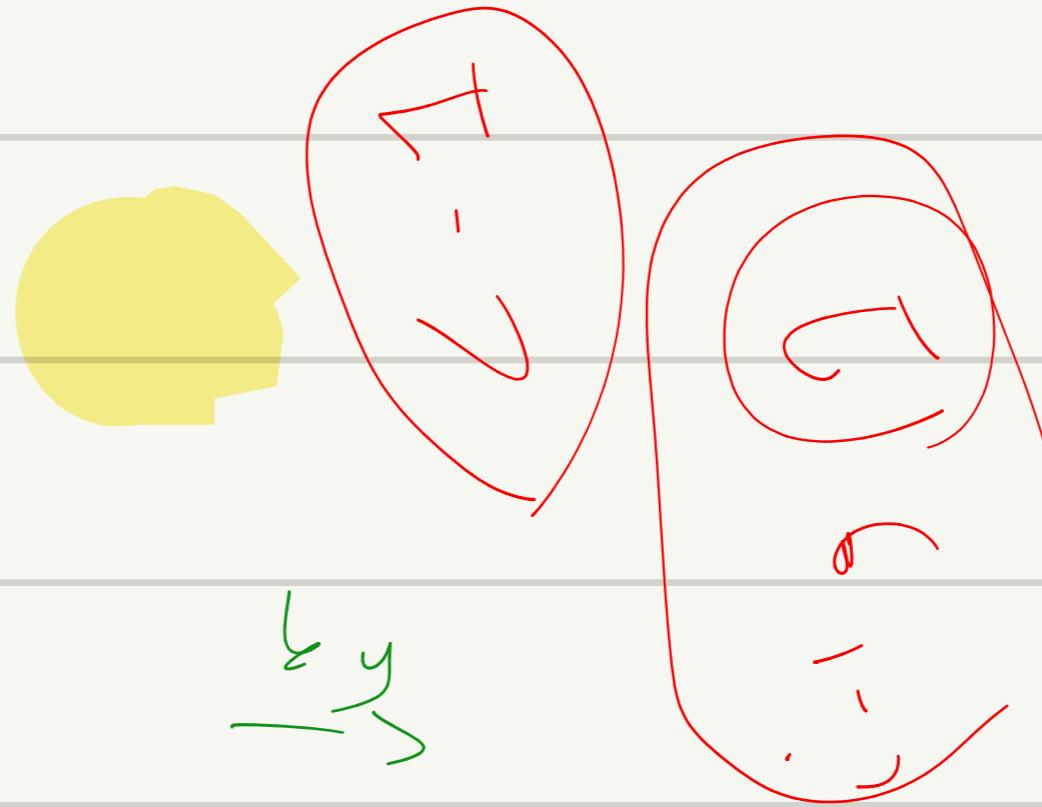
2
 1
 1
 1
 1

$2 \rightarrow f_1$

$35 E^1$

$13 E^1$

$12 E^1$



k_y

row index

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

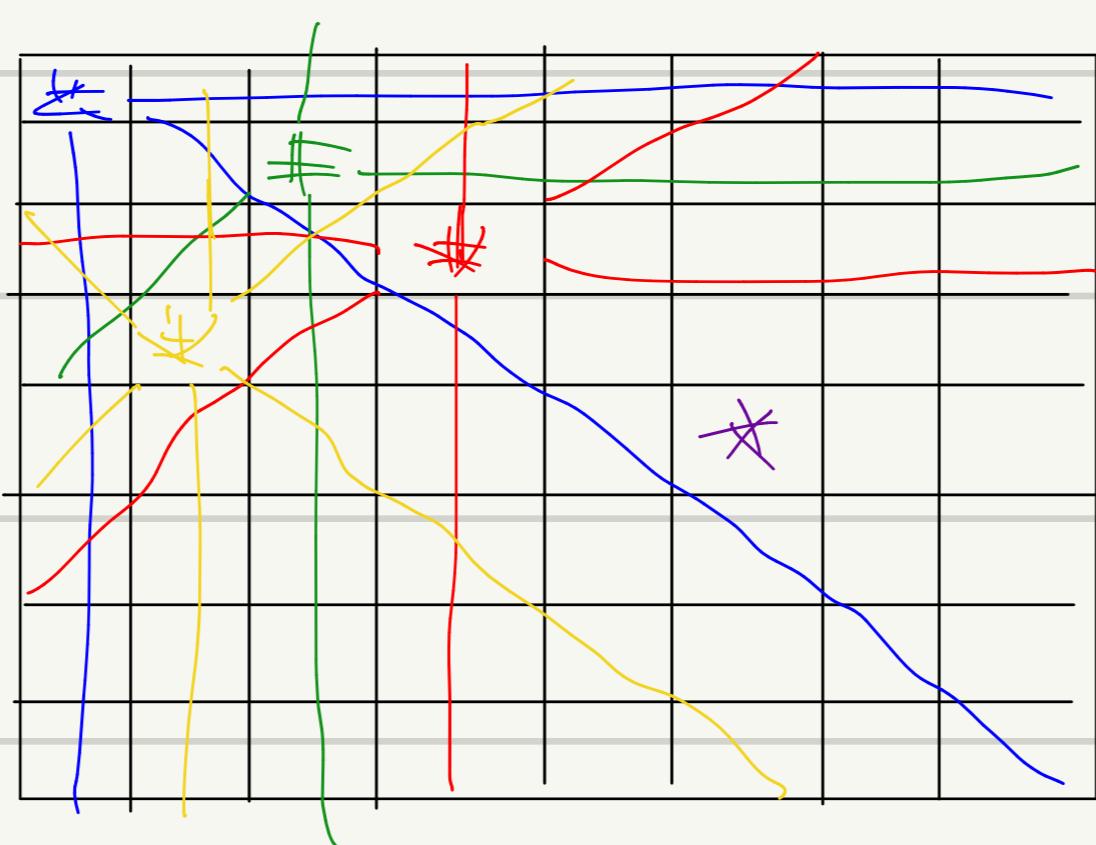
c_1
 c_2
 c_3
 c_4

6. ✓
3. ✓
2. ✓

12. Hafta

Backtracking

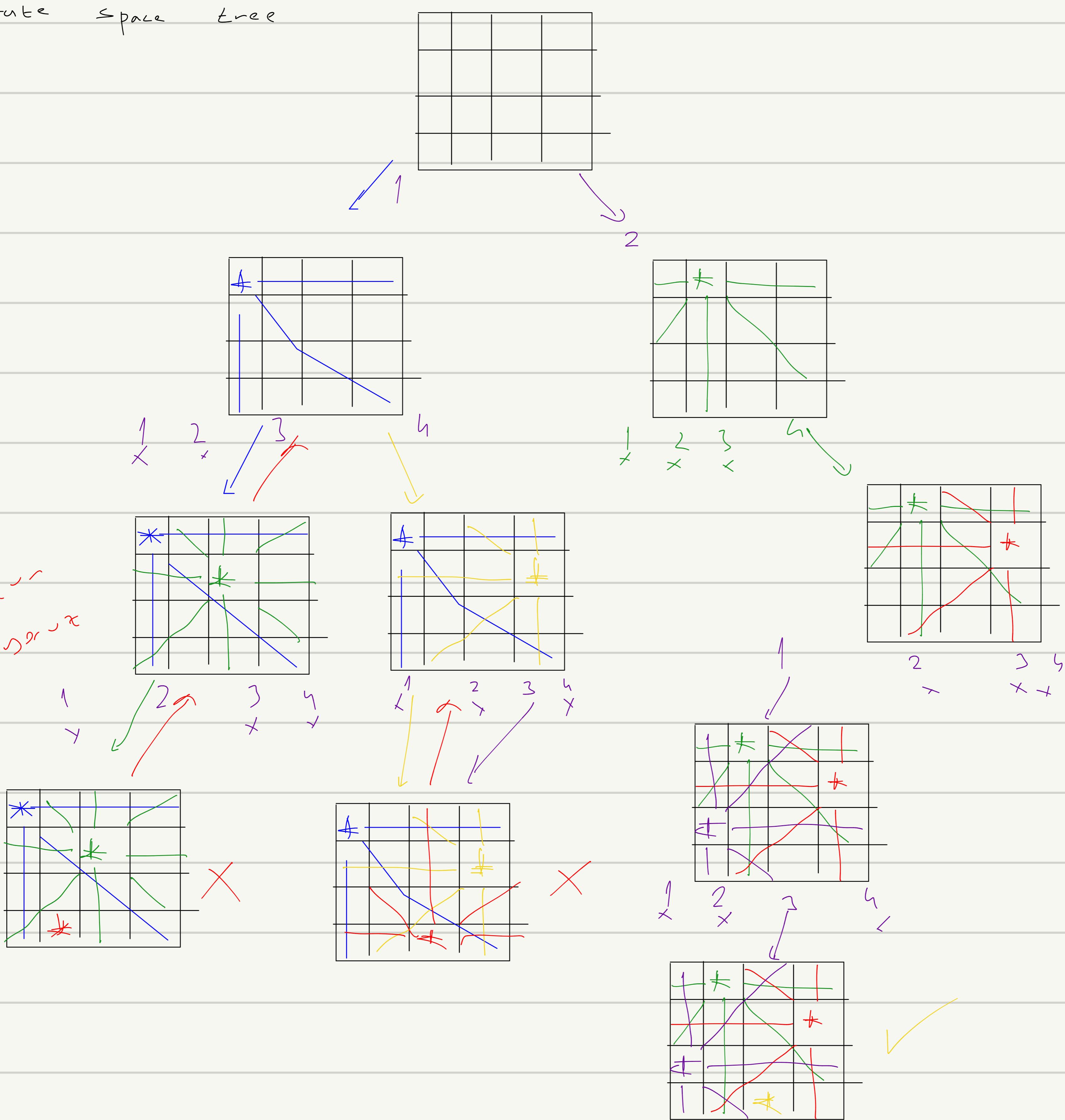
N-Queen Algorithm



ds1 montik 8 une vezisi
yerlestiriken, yerlesmesine goni
dinemek gerektigini.

- ① Brute force'da $(8^8) \rightarrow 6$ milyon deneme
- ② Satır Kontrolü yapsak $n^n \rightarrow 8^8 \rightarrow 16$ milyon
- ③ Satır ve sütun kontrolesi yaparsak $\rightarrow n! \rightarrow 40$ bin
- ④ Yerlestirmediğinde bir hizla: durum, donne islemi $\rightarrow 2$ bin
 \hookrightarrow back tracking

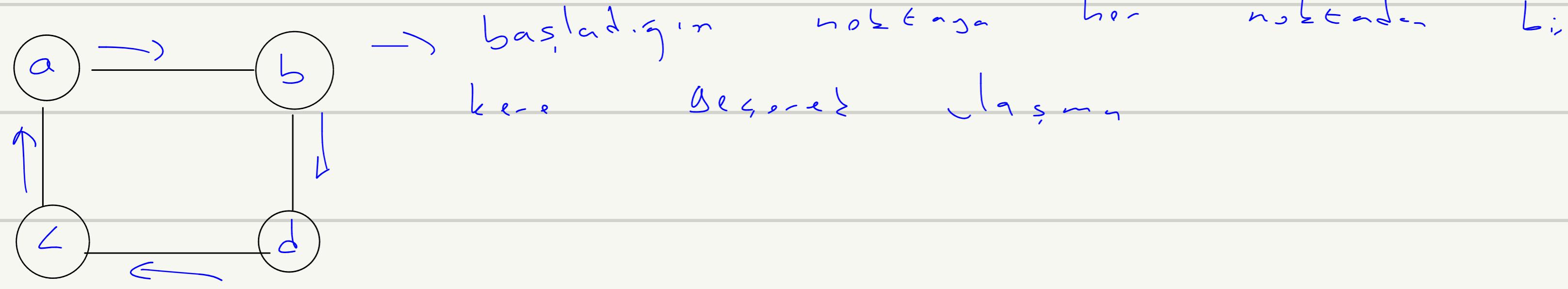
State space tree



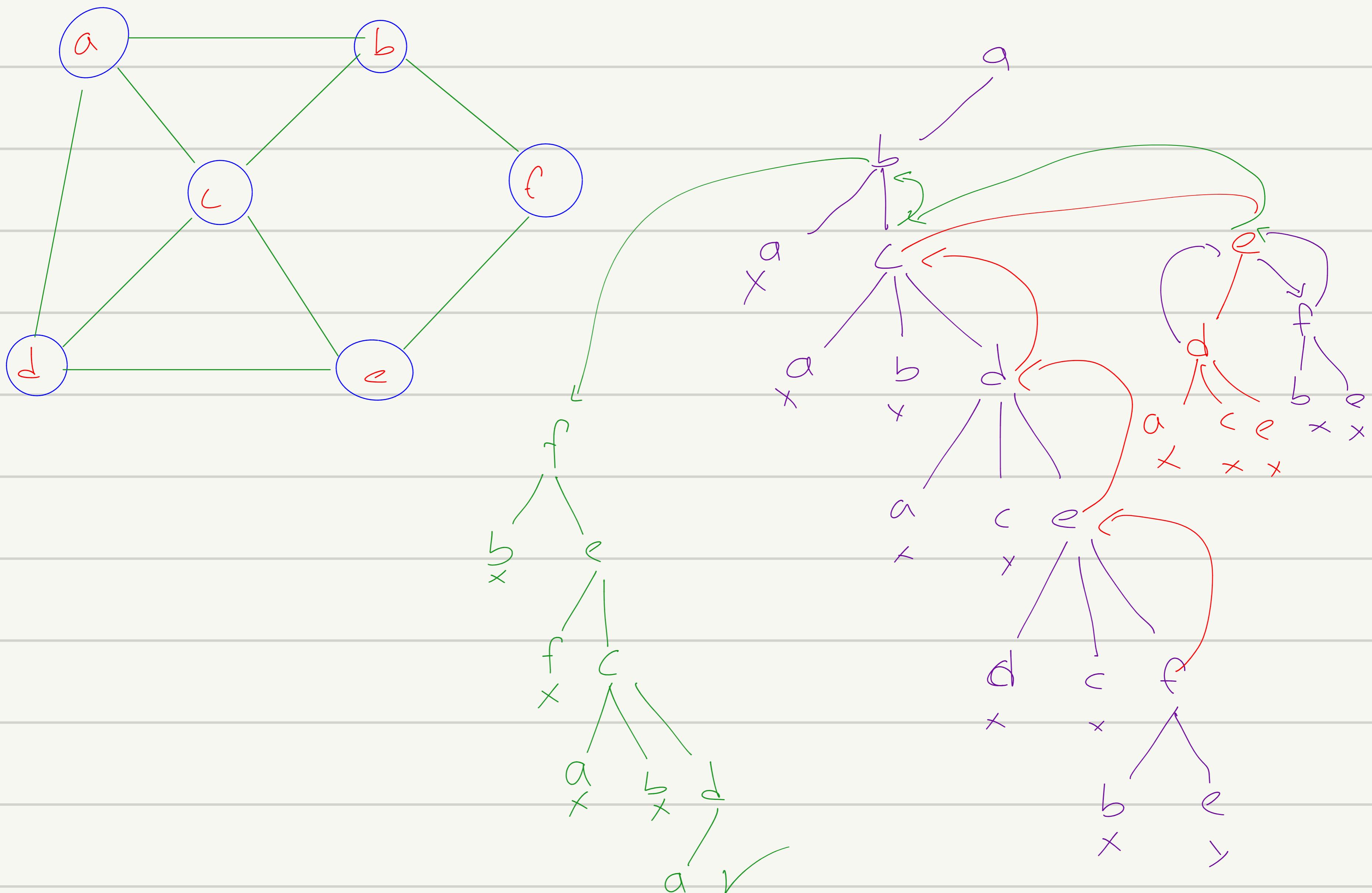
Fahmi Konan

Circuit \rightarrow DFS

6:5;



State Space tree

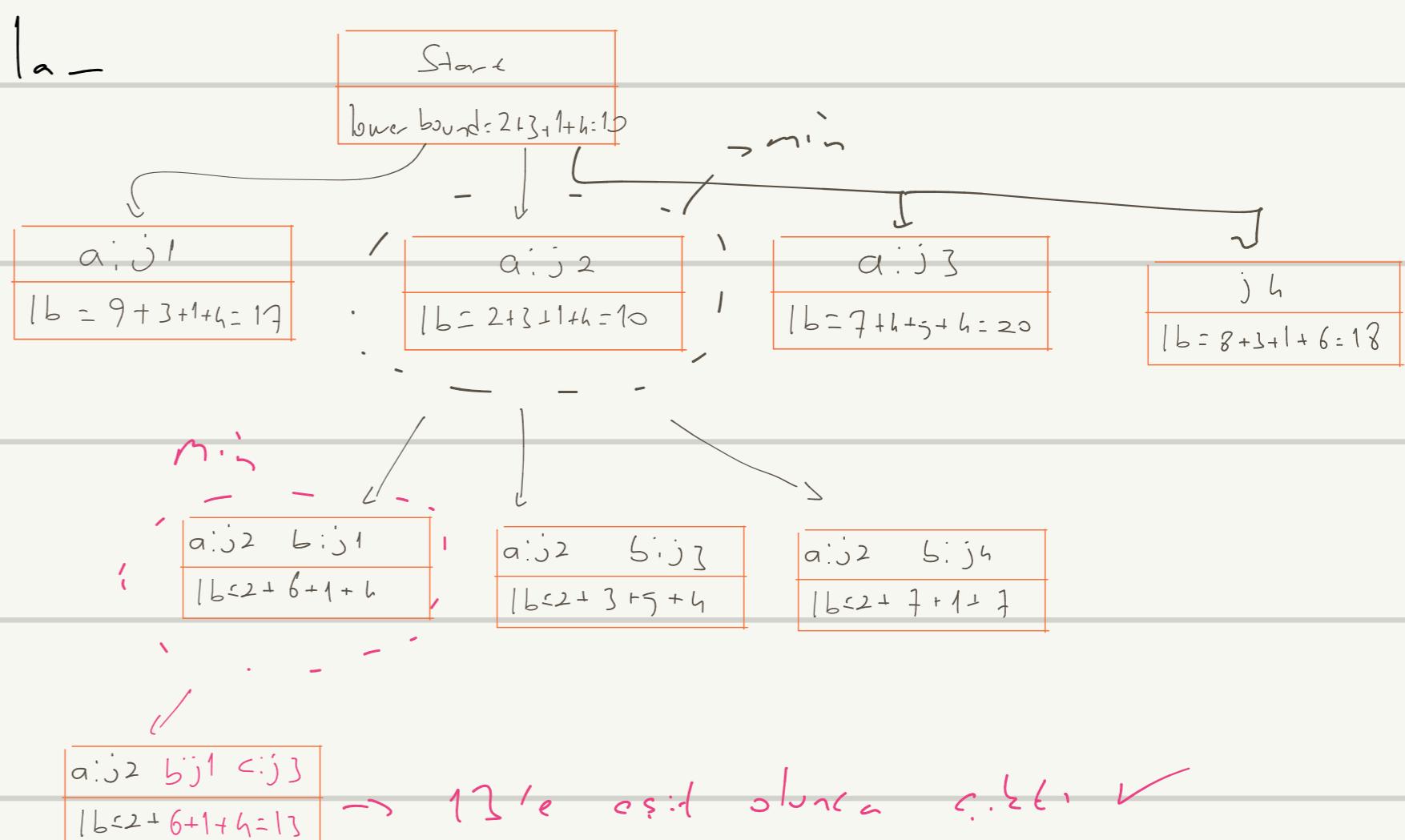


2

Branch and Bound \rightarrow BFS

State Space Tree

| Job 1 | Job 2 | Job 3 | Job 4 | Jobs |
|-------|-------|-------|-------|--------|
| 9 | 2 | 7 | 8 | ahmet |
| 6 | 4 | 3 | 7 | mehmet |
| 5 | 8 | 1 | 8 | cavdar |
| 7 | 6 | 9 | 4 | sevket |



Knapsack Problem

$$W = 10 \text{ kg}$$

| <u>item</u> | <u>weight</u> | <u>value</u> | <u>v_i/w_i</u> |
|-------------|---------------|--------------|-----------------------------|
| 1 | 4 | 10 | |
| 2 | 7 | 12 | 6 |
| 3 | 5 | 7 | |
| 4 | 3 | 12 | 4 |

$$v_1/w_1 > v_2/w_2 > v_3/w_3 > \dots$$

upper bound = $v + (W - w) \cdot \underbrace{(v_{i+1}/w_{i+1})}_{\substack{\text{gering} \\ \text{kalar} \\ \text{bistig}}}$

add knapsack

den zulässigsten

SST

| |
|------------------------------|
| $w=0, v=0$ |
| $Ub = 0 + 10 \cdot 10 = 100$ |

| | |
|---------------------------------|------------------------|
| $w=4, v=10$ | $w=0, v=0$ |
| $Ub = 10 + (10-4) \cdot 6 = 76$ | $Ub = 10 \cdot 6 = 60$ |

| | |
|----------------------------------|---------------------------------|
| $w=11, v=82$ | $w=4, v=40$ |
| $Ub = 82 + (10-11) \cdot 6 = 76$ | $Ub = 10 + (10-4) \cdot 5 = 70$ |

| | |
|---------------------------------|---------------------------------|
| $w=9, v=65$ | $w=4, v=40$ |
| $Ub = 65 + (10-9) \cdot 6 = 69$ | $Ub = 10 + (10-4) \cdot 4 = 64$ |

| | |
|------------|---------------|
| $w=12, v=$ | $w=9, v=65$ |
| $Ub =$ | $Ub = 65 + 0$ |

13. HAFTA SON

Polinomyal $\rightarrow O(n^{\text{maks}}) \rightarrow$ prefixif değil

efektifler $\rightarrow O(n^2), O(n^3), O(n \log n)$

efektif olmayanlar $\rightarrow O(2^n), O(n^n), O(n!), O(n^{1+\log n})$

$n^{10^{\text{maks}}}$ \rightarrow tractable
 $n^{\log n}$ \rightarrow in ..

gözdeğerler ana yerdeki düşüncelerdeki Jelgit'ler
 " " " " " "

et-tractable.

NP \rightarrow Non det. polynomia!

Traveling salesman \rightarrow Başladığın yerden her gerek 'e' doğrularak, her sonu: gerne!

Hamiltonian circuit \rightarrow Renk algıları. Traveling salesman

her dumur Jenevič Lazan
 {
 } \rightarrow Dynamic $\rightarrow 2^{n^n}$
 } \rightarrow Brute force $\rightarrow n! \rightarrow$

Problems

Decision
Problems

Optimization
Problems

Deterministic \rightarrow Çözüş Belli;

\hookrightarrow max elem

```
NSort(A,B&
for(i=0;i<N;i++&
B[i]=0;
for(i=0;i<N;i++&
j=rand(1..N&
if(B[j]!=0& then failure
B[j]=A[j]
endif
for(i=0;i<n-1;i++&
if(B[i]>B[i+1]& then failure;
endif
print(B&
success
0(1&
```

Nondeterministic \rightarrow Çözüş Bellisi

\hookrightarrow En yakın şe

```
NSearch(A,x&
j=choice(1..n&
if(A[j]=x&
print(j&
success
endif
print(0&
failure
```

Complexity $O(1^n)$

↑
NP

(1) guessing
 (2) verification
 (3) output

$P \subseteq NP$

$NP \rightarrow$ polinomial z maniera dogru lösbar algorithm,