

---



12/13

I/O

## 1. BufferedReader

1) 1 2 3 4 — Integers

→ StringTokenizer st =

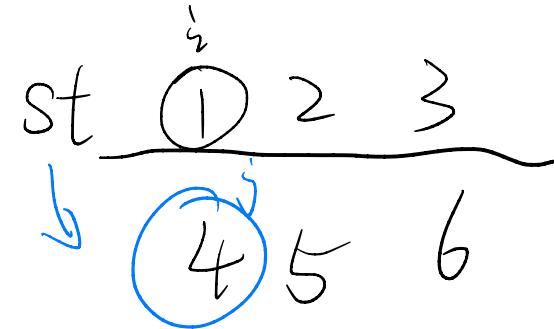
new StringTokenizer(f.readLine());

int i = Integer.parseInt(st.nextToken());

st =

→ new StringTokenizer(f.readLine());

int j = ...;



y

exe 8

→ 2) String {  $S = ?$   
~~1 2 3 4 5 6~~ }  
 $S \rightarrow abc\ de\n$   $S_1 = ?$   
 $S_1 \rightarrow ij\ ii\n$

$S.length() \rightarrow 6$  charAt( )

$S_1.length() \rightarrow 5$

3) char

a b c  
st = new S.T.(f.readLines())

char c = Character. Parsechar(st.)

## 2. PrintWriter.

BufferedWriter bw = new BufferedWriter  
(new FileWriter(" "));

PrintWriter out = new  
PrintWriter( bw );

out.print(" "); → to file.

out.close();

# ASCII

A - Z

$$\alpha = \bar{z}$$

— 9

1 2

ABC II

char A > int

(int) A → 65

int  $A - i = \underline{(\text{int})}^{\text{of}} A$   
↓  
- 4.

0, 1 2 3 4 5  
COMET Q.

65 - 90

A - Z

↓

① char At.      1 - 26.

② (int) char - 64      1 - 26.

③ Calc.

1.

1 2 3 4 | 5 6 7 8 9 10

G G H G H G H H G G

(C) (H) (G) (H) (G) (H) (H) (G) (G)

↓ ↓ ↓ ↓ ↓ ↓

1 2 3 4 5 6 7 8 9 10

H H G H G G F G G H

(H) (G) (H) (G) (G) (F) (G) (G) (H)

X

~~20000~~

1 2 3 4 5 6 > 7 9 10 11 12 13 14

→ G G G H G H H H G A A H G H G

G H G H H H G H G H G H G

~~A~~

j=4. 6. 8. 10. 12. 14.

max 24 (6) 3. 3. 2. 1.

for ( $i=0$ ;  $i < \text{len}$ ;  $i++$ )  
    for ( $j=0$ ;  $j < \text{len}$ ;  $j+=2$ )  
        reverse  
 $\text{lmax}_i$  matory. even G.

)  
,

print l<sup>'</sup>)

Time  
exceeds  
the  
req

Time →

Space.

Map < Integer, String >

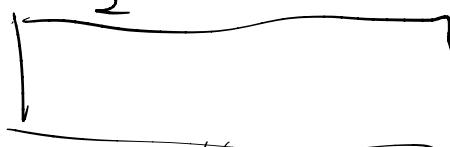
map.put(k, str)

(1) str  
(2)

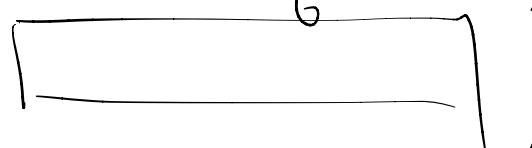
for (i = 0; i < n; i++)

→ (1) k

Map.put(k, v)

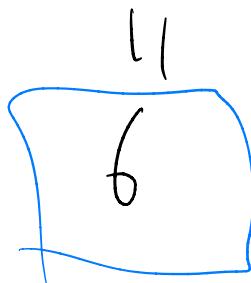


ans →



String

max-matching



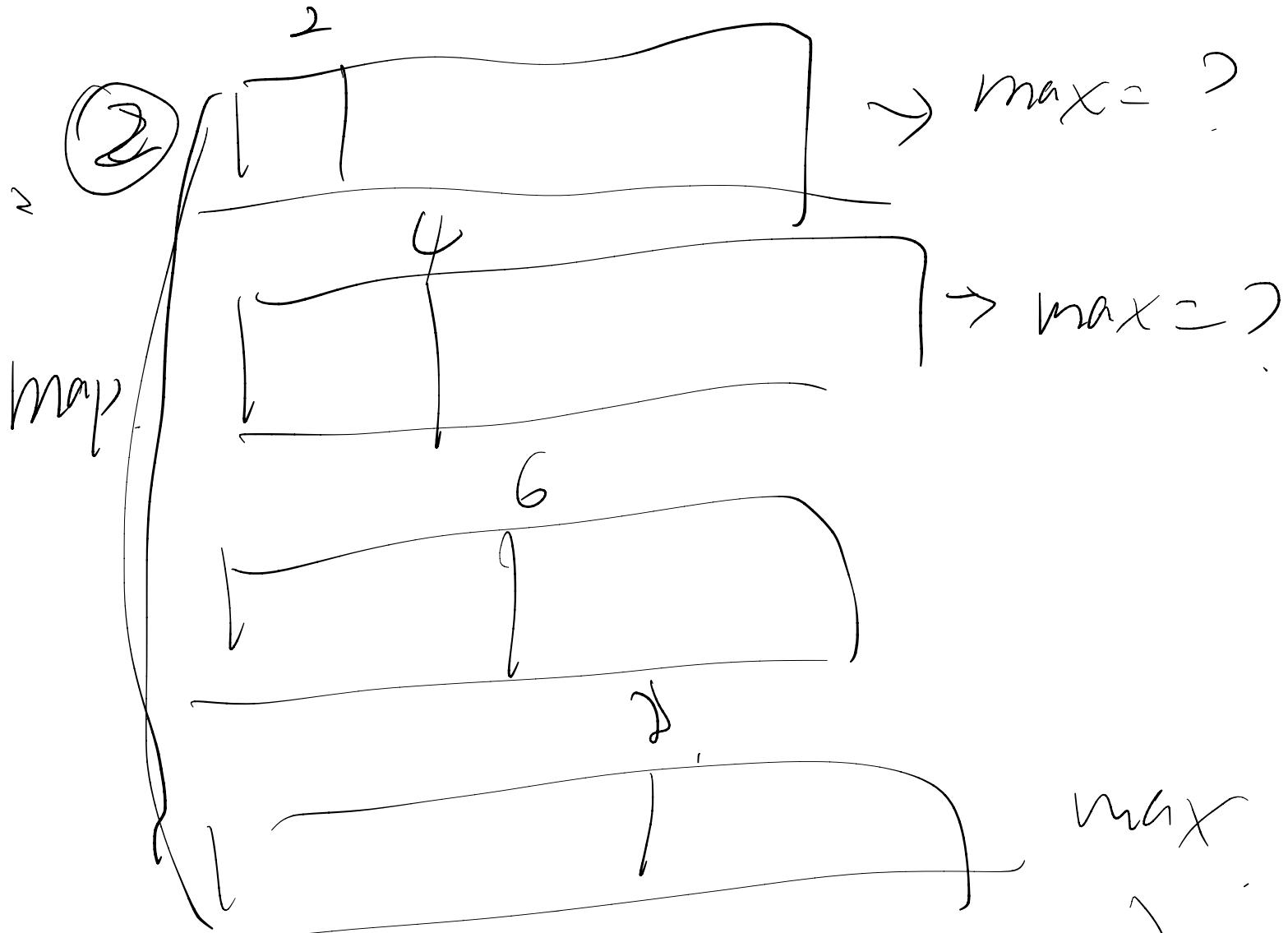
→ (2)

map.put



max-matching

```
for ( l → ∞ ) { ← 2,  
| ① max_match = 0,
```



```
| ② map.put( k, max ).  
| } if ( max_new <= max_old ) {
```

```
break;  
return key.
```

~~G G G H G H G H G H G H G H G~~

~~G H H | G H H | G~~

even not G  $\rightarrow$  G (max)

ans = 2

2)

ans = 0

for(j = n-2; j >= 0; j -= 2)

(if str[j] == str[j+1]) {  
    ans++;

    str[j] == G ) == 0

    ans / 2 == 0.)

    ans++; j == 1

12/14

$O(n)$   $\rightarrow$

while ( $i=0 \rightarrow n$ )  
For ( $i=0 \rightarrow n$ )

$O(n^2)$   $\rightarrow$

while ( $i=0 \rightarrow n$ )  
 $\hookrightarrow$  For ( $i=0 \rightarrow n$ )

## Data Structure

arrays.

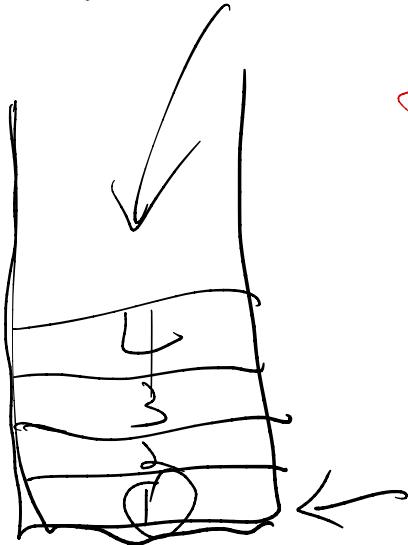
ArrayLists



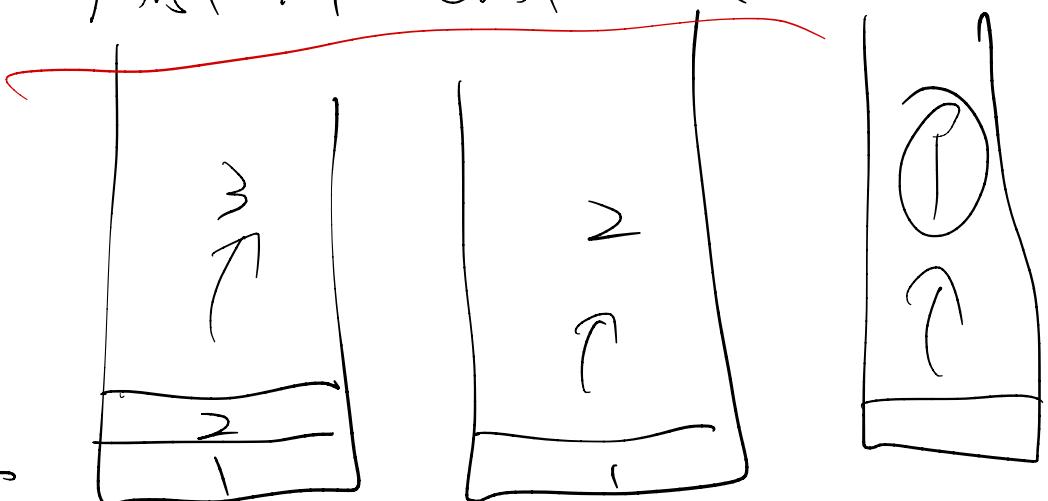
stack  
queue  
double queue  
linked list

map / set

Stack

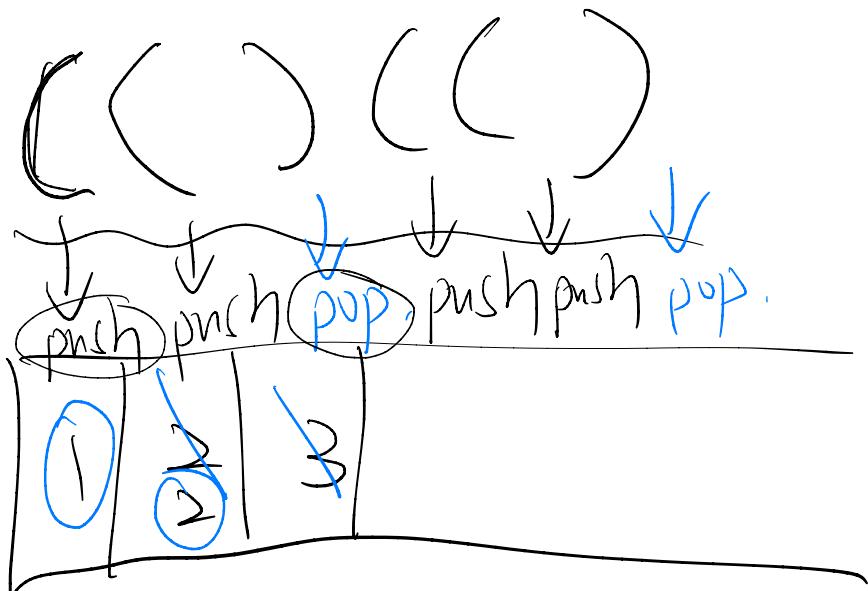


First in Last out.



Stack → pop()      push()

Stack s = new Stack();  
s.pop();      s.push();

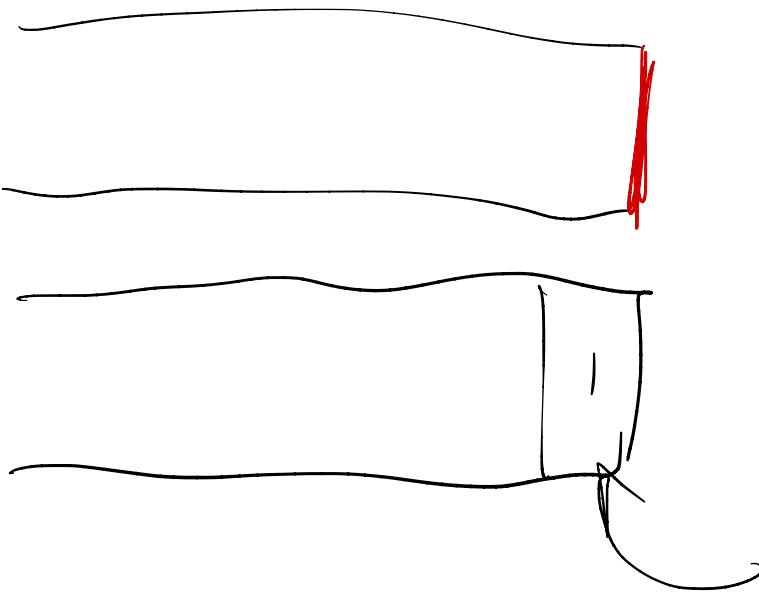


s.size()



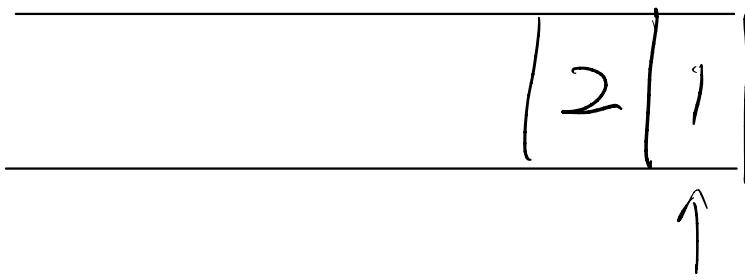
O(n)

queue / DR

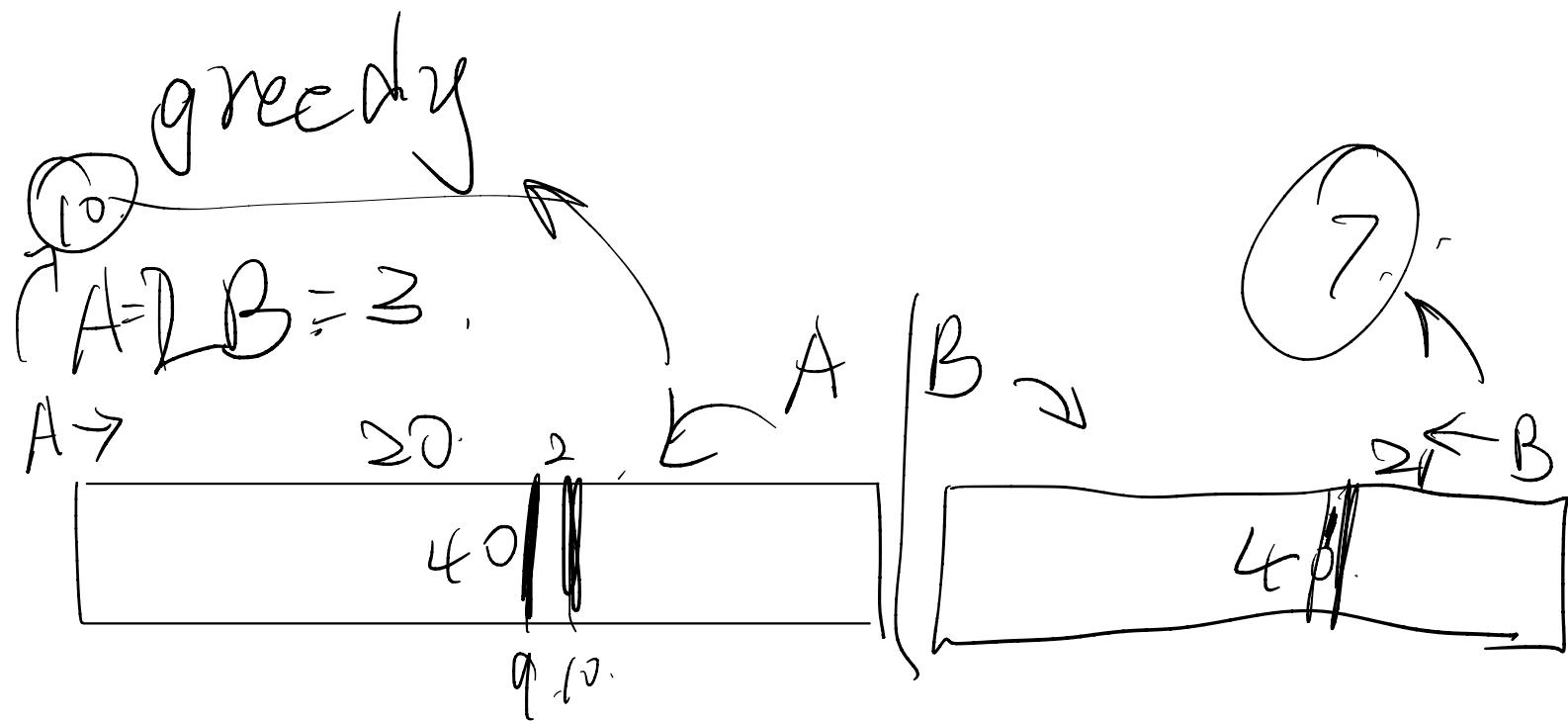


single-sided

q.insert()  
q.delete()



Double-Ended  
queue



Map / Set → tuple

dict (key : value) → Map.

HashMap ↪ key ↪ value ↪ hm

String,  
Integer, Integer  
Character Character

= new HashMap<key, value>();

String → Integer

Dave	0
L	0
B.	0

For ↓  
⇒ L

- Dave 200 >  
L B.  
1. map.get(name)  
2. map.put(name, a)  
3. C B,

12/15

① naive record  
max match

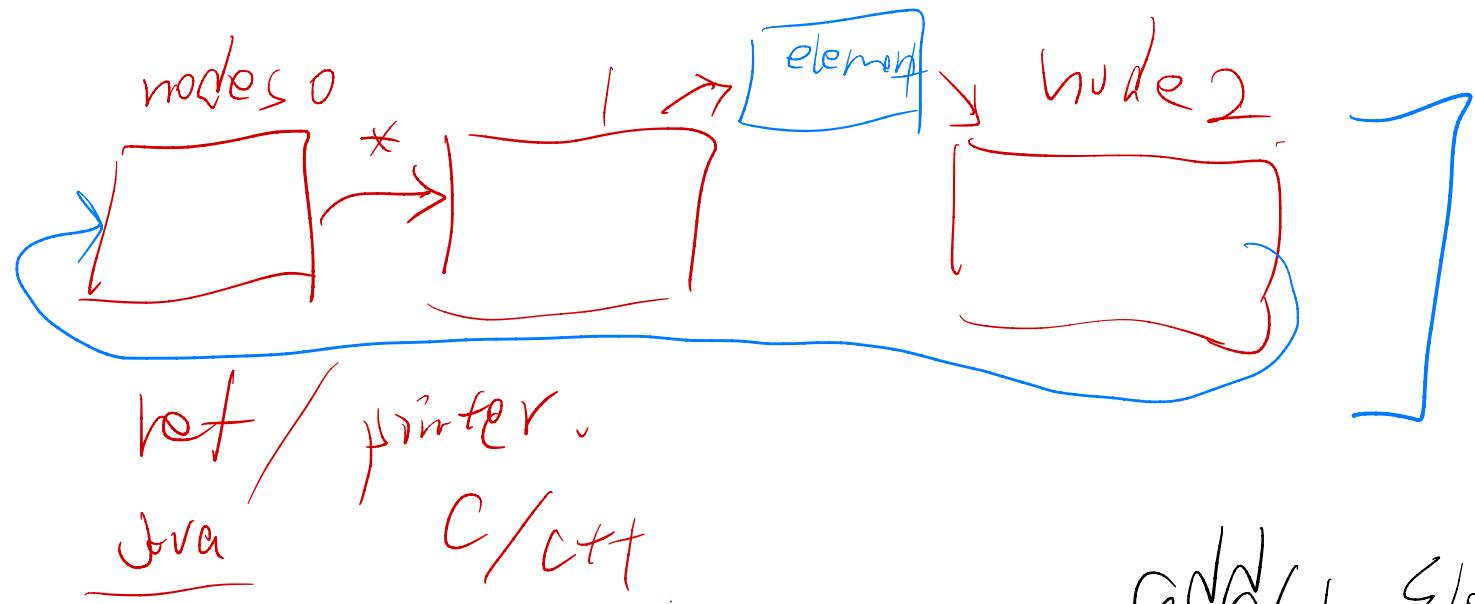
1. I/O., simple algo.  $O(n) = \underline{\text{even}}$

2. data structure.  $\rightarrow$  Map.  
① Ubiquitous  
Time  $\rightarrow$  mem.

3. array  $\leftrightarrow$  ArrayList.  
\* func. / methods.

# Array Vs. ArrayList. LinkedList.

list list = ArrayList<Integer>()



array[ ]. ① primitive,

ArrayList.

[ 1, 2, 3, 4 ]

{ 1, 2, 3, 4 }

ArrayList

Collections

ArrayList<Integer> a1 = new ArrayList<Integer>()

< 3 , 2 , 1 >  
ascending.

Collections.sort(a1);

< 1 , 2 , 3 >

bubble sort ←  
merge sort ←

$\langle 1000, 100, 8, 9, \dots \rangle$

$10^3$ .

Collections,  $\text{sort}(\text{a1})$  ;  
↓

$O(n \log n)$ .  $A^*$



function      arraylist

(private      int) func-name (int a, intb){

int ans)      ans 2 .

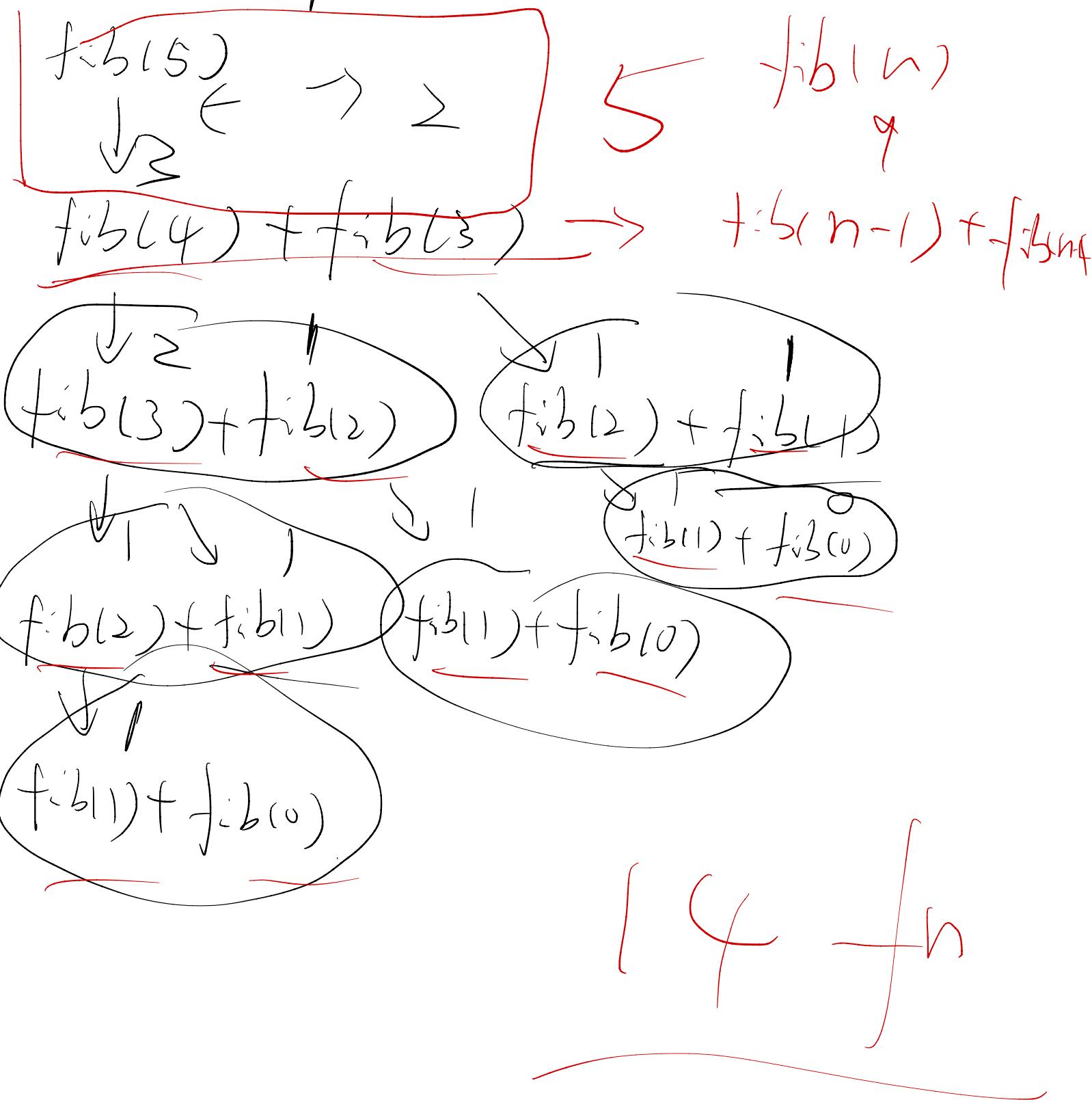
}

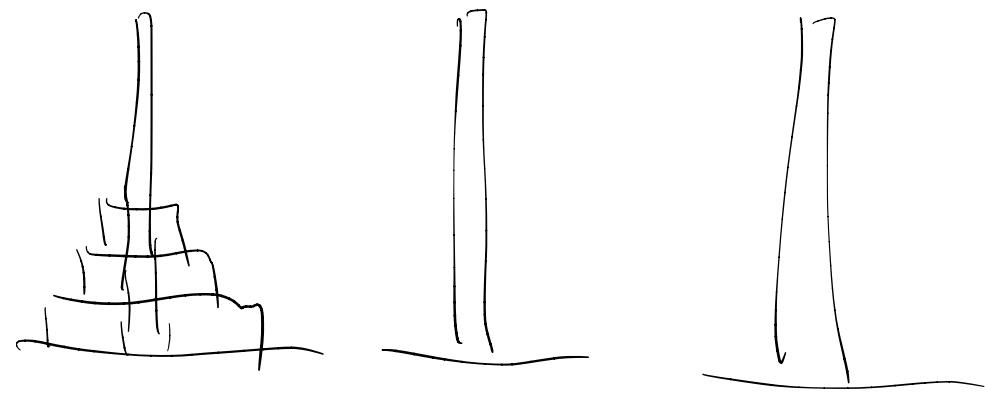
    return ans;

}

# recursion

Time → Space

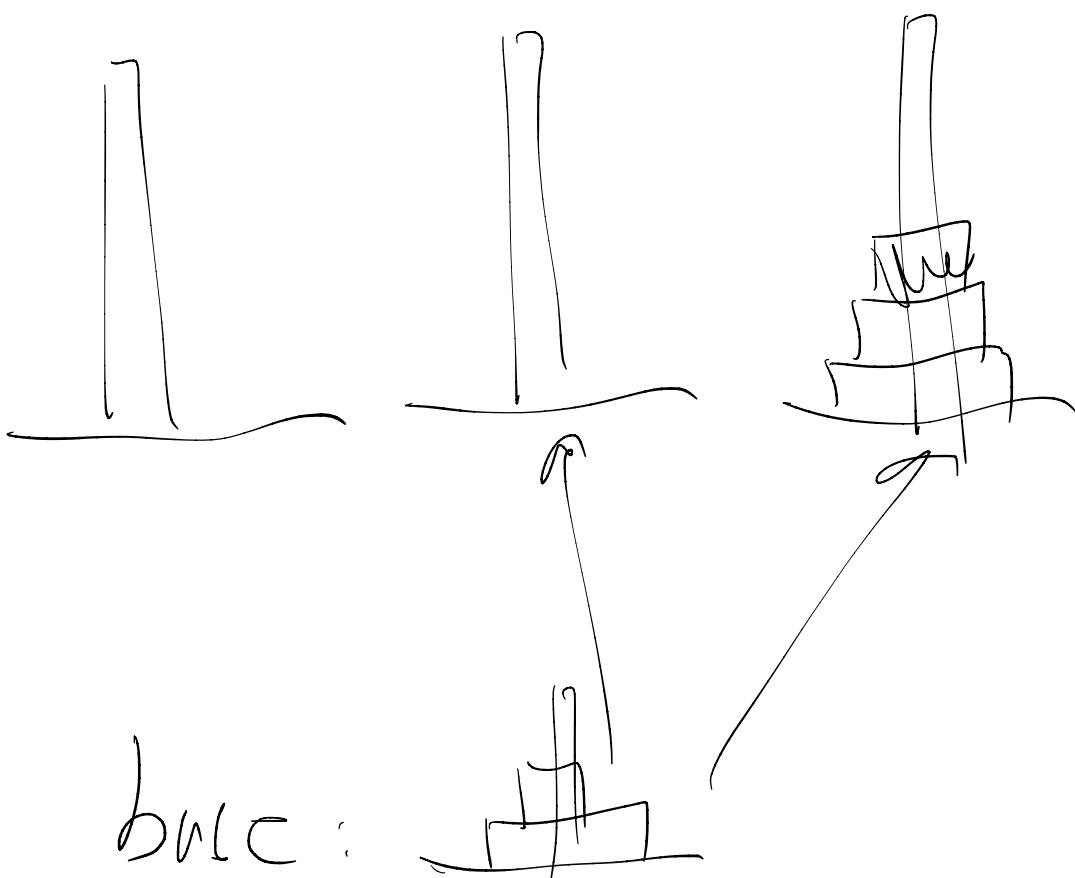




rule:



always here  
small ones on  
large ones



rec<sub>f(h)</sub> ( $n-2$ )  $\rightarrow$  3rd  
 $(n-1) \rightarrow$  2nd

func:

① separate.

$L \rightarrow \leq$      $G \rightarrow \geq$     less

$L \quad 3$   
 $G \quad 5$

$\times 3$      $\leftarrow$

$L \quad 5$

$2 \quad 6$

$G \quad 2$

$G-1$   
 $G_0$

$L_1$

$L_2$

$L_3$

$L_4$

$L_5$   
 $L_6$

$G_2?$

$G_3?$

$G_4$

$O$

$3$

Data Struct ?

a algo ?

ArrayList L, g

① read n,

2 → N+L; add. L or G.  
↑

② Collections. sort()

③ find out max - G  
min - L

Or if(max - G < max L)  
if(min - G < min - L)

Context

brainstorm

concepts

algo.

math / living

func

static var. ↙

Data Ptn

write

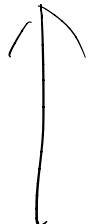
machine / program

debug

Test Cases?

\* sample ← obvious

\* Test



unit test

10.  
L 5

9 6.

{ 3

9 10.

Debug.

① Write down. ( Problem solving )

② techniques.

\* check out → System.out.println()

class test {

{ static int[] a = [0, 1, 2, 3].

st main()

{ helper1();  
helper2();

}

a[

[0, 2, 4, 3],

at void helper1() {

a[n] += 1;

}

st void helper2() {

a[n] += 2;

}

3

num

If  $(\zeta(1) = \infty)$ :



0 0 0

0 0 0 |

0 1 0

| 0 0

| 1 1  
| 0 0 |

5 2

1 2 3 4 5  
G H H G G  
1 G G G  
2 H H H H  
3 H H H H H  
4 G G G G  
5 G G G

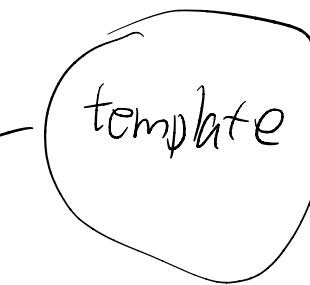
5 1

1 2 3 4 5  
G H H G G  
5 G  
1+ H H  
2 H H H  
3 G G G  
4 G G G  
5 G G G

· H H ·

Beginning - #1

I/O - Scanner



One  
two  
three  
class one

main  
Scanners

Data Structures / Basic

Stack  
queue  
DQ  
linked list  
Map.

) - ID

| C:V

Recursion ←

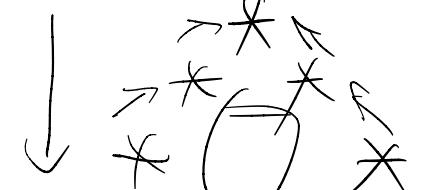
Iteration

For loops

If-else

nested - for loops

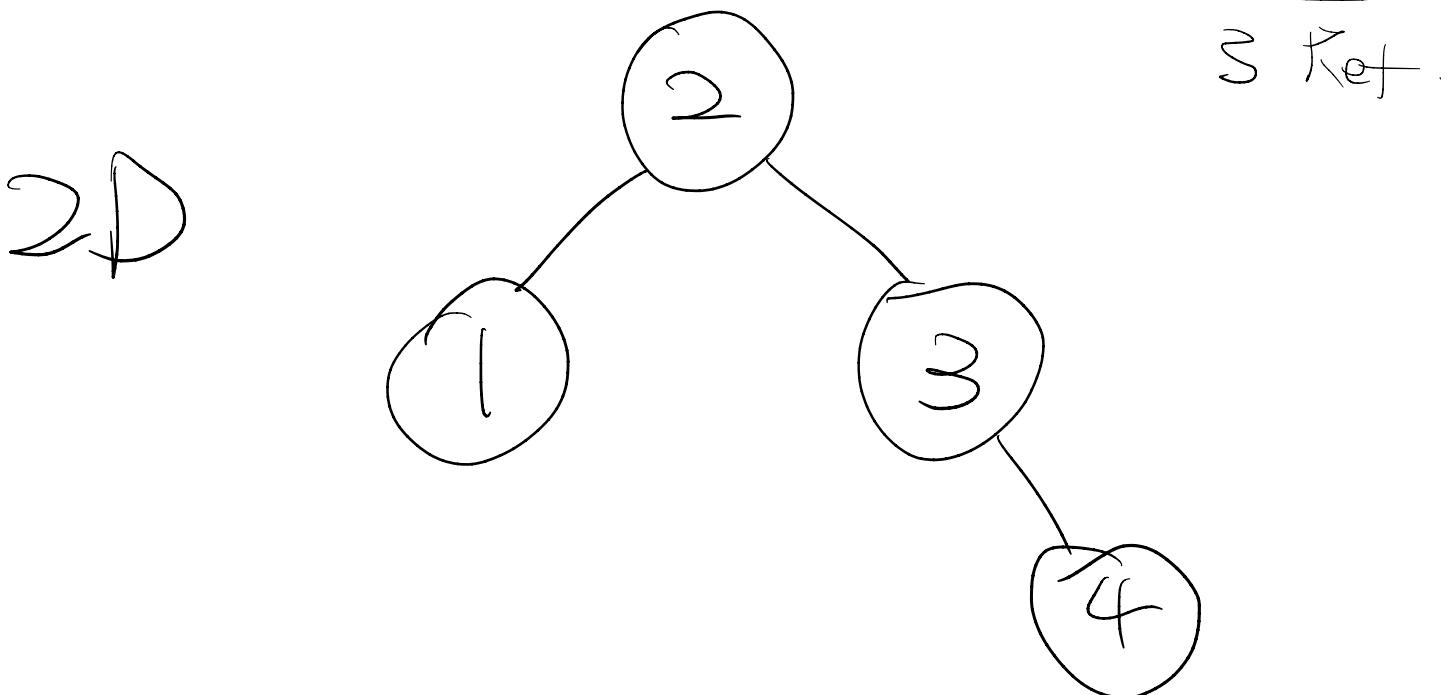
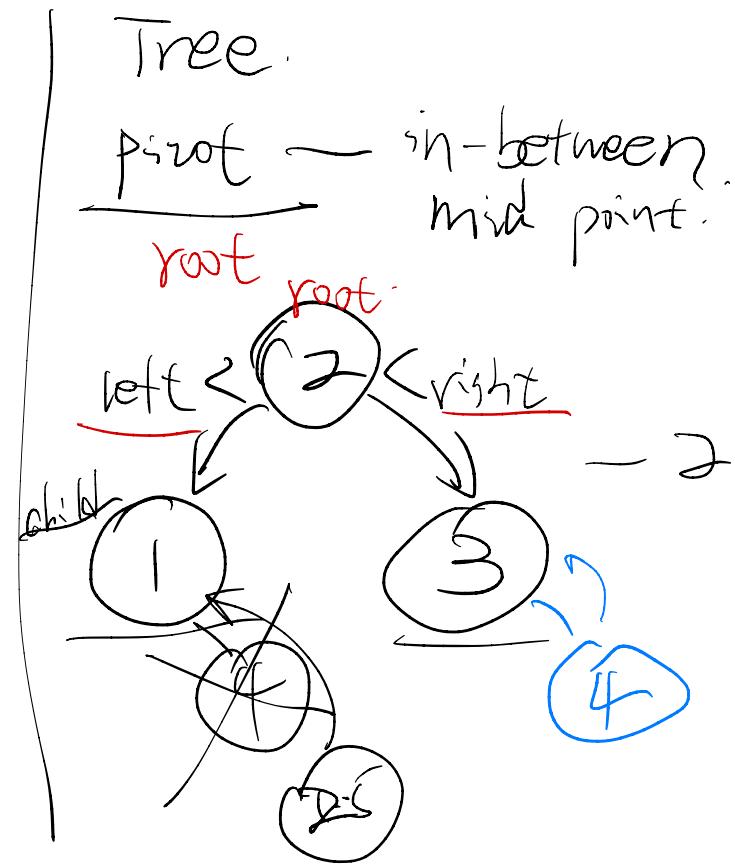
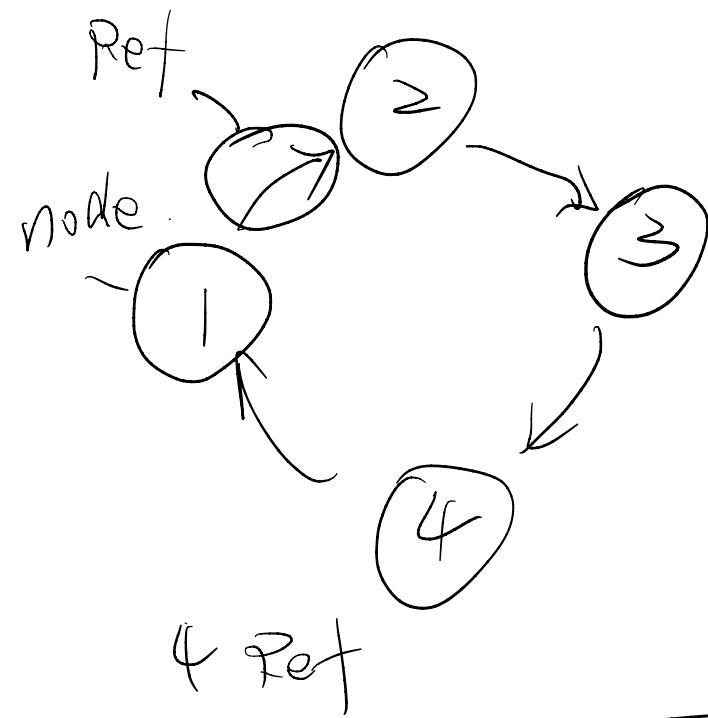
$O(n^m)$

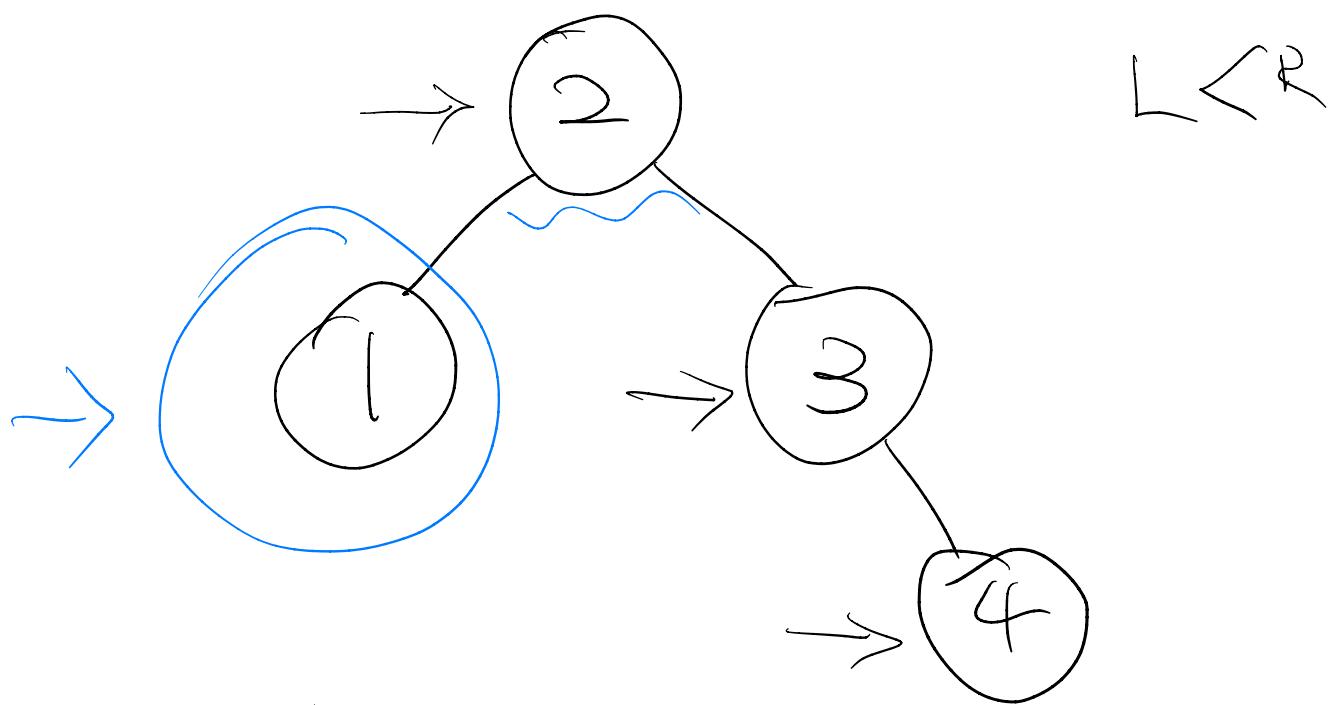


Tree - Binary Tree

○ - node.

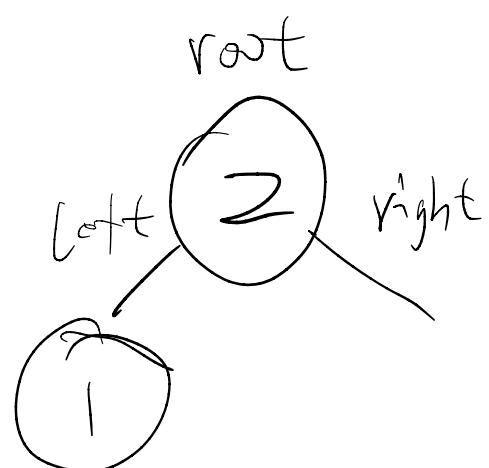
Linked List.





1. find an element.

Search — binary Search



1. addNode.

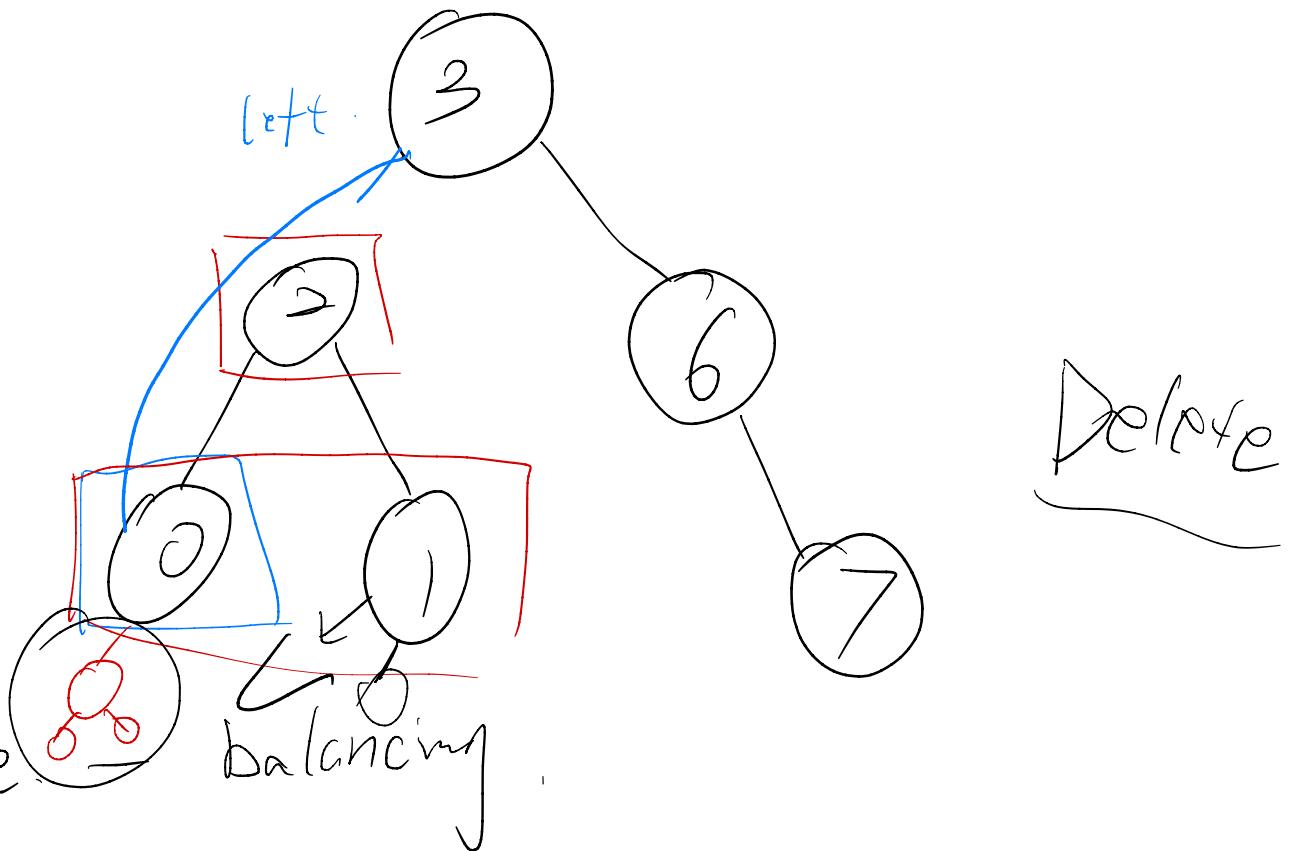
(root, 1);



3. addNode (current, 1) → 1

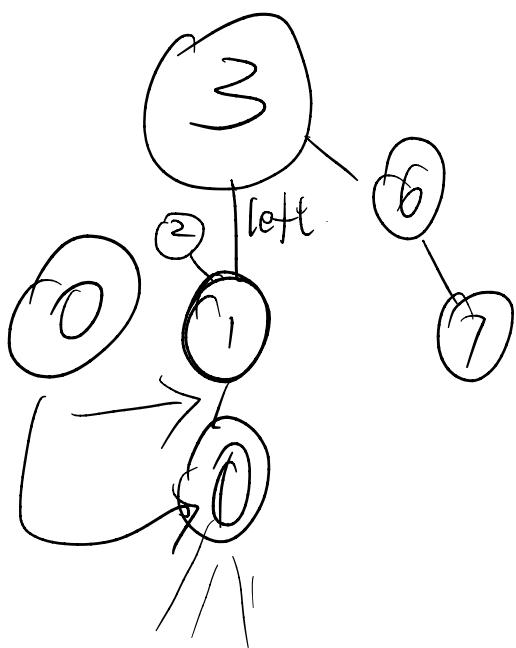
```
class Node {  
    int value;  
    Node left;  
    Node right;  
  
    Node(int v) {  
        this.value = v;  
        r = null; L = null;  
    }  
}
```

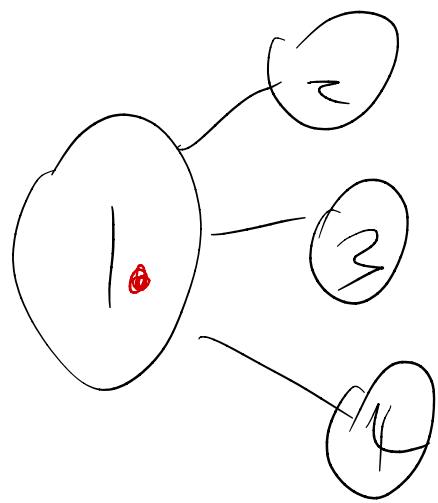
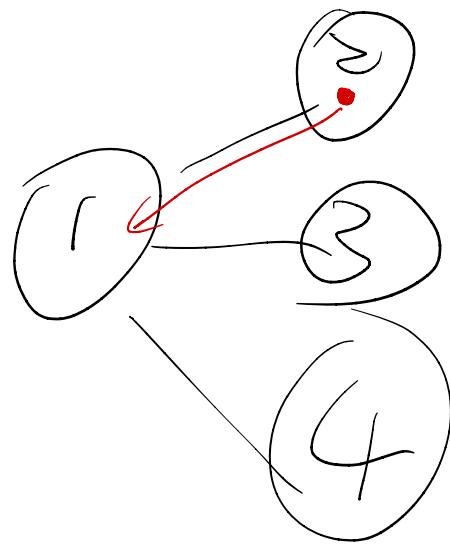
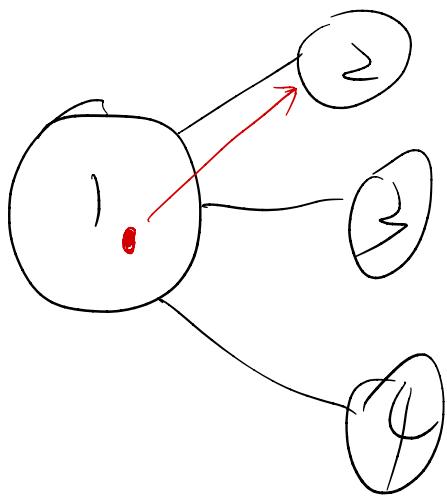
```
class Binary Tree {  
    Node root;  
}
```

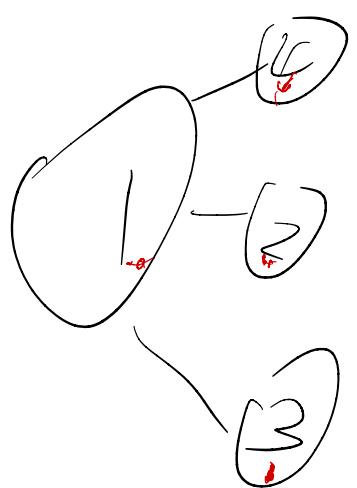
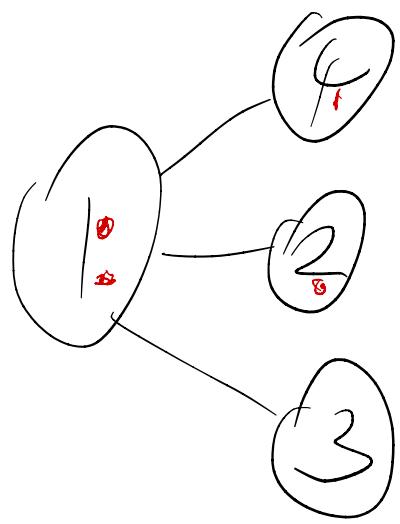
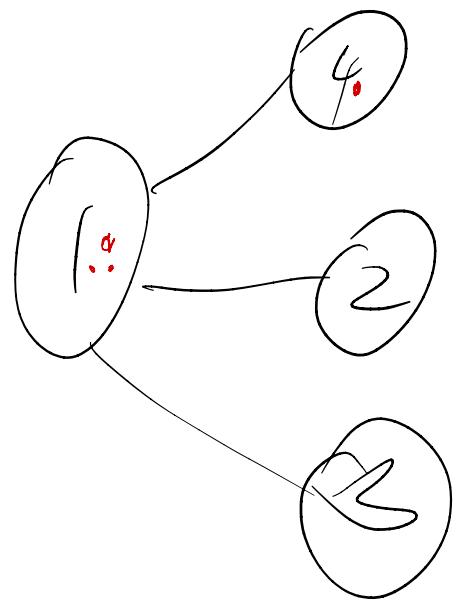
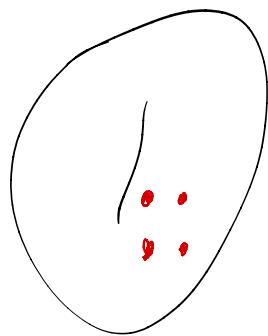
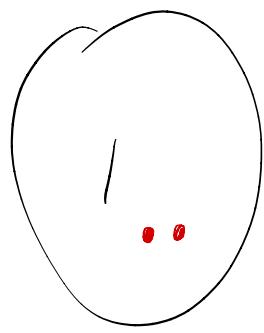


2. 1) unlink , upper  
 => left most / right most

3. balancing ✓



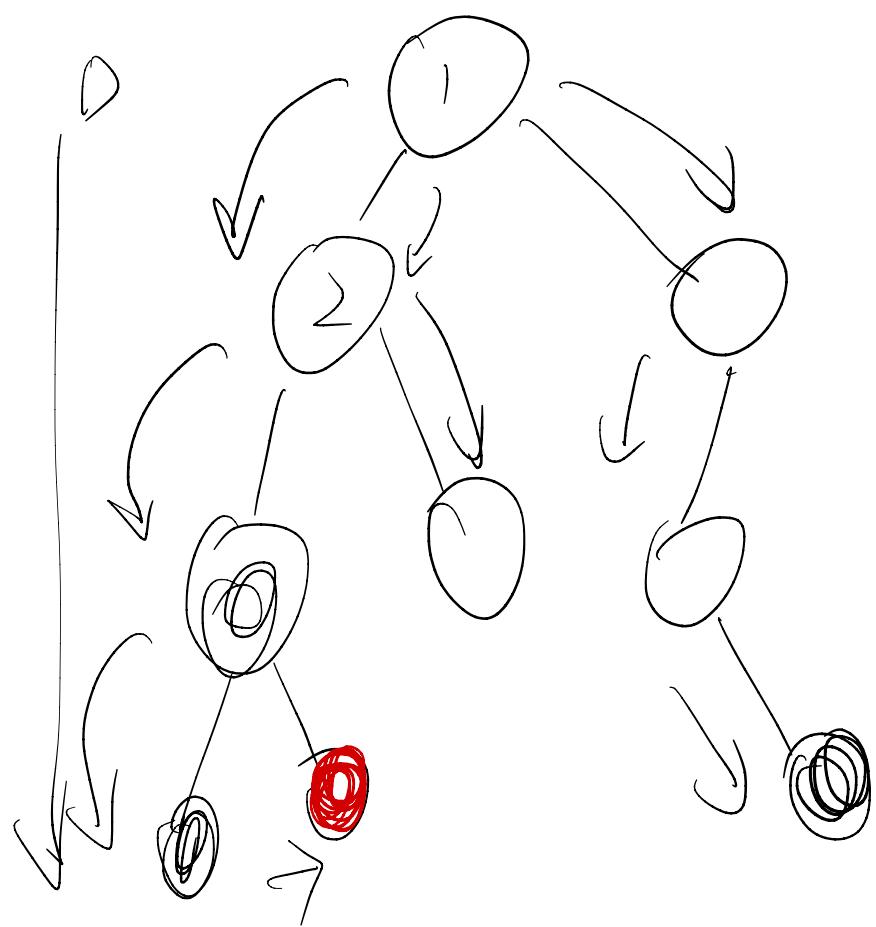




# DFS

## Depth First Search

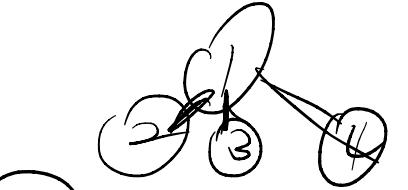
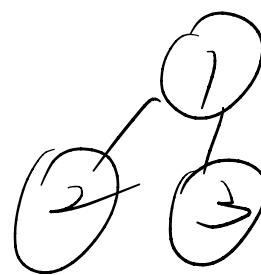
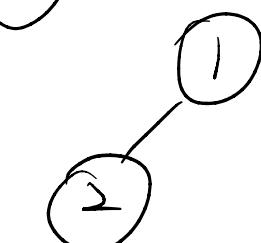
graph node



4

$$\begin{array}{r} 1 \quad 2 \\ \hline & 3 \\ \hline 1 & 2 & 3 \\ \hline \end{array}$$

T 0 5

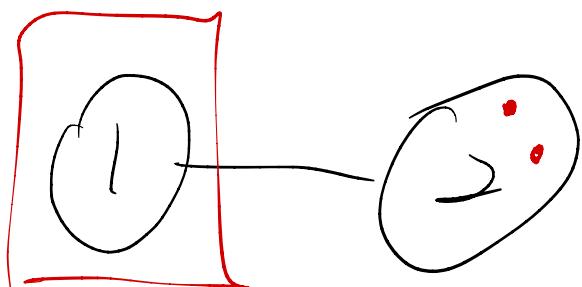
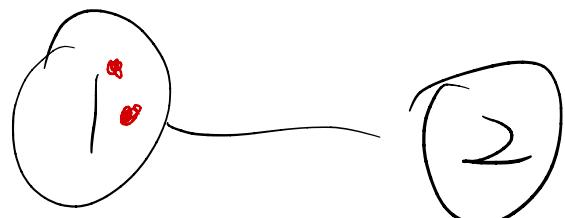
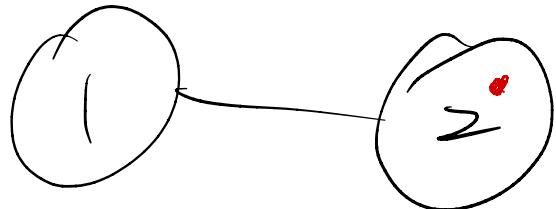
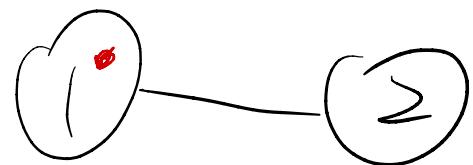
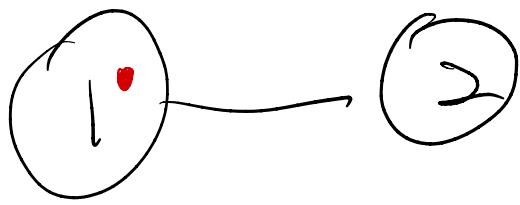


T 0 3 → 4 - 3

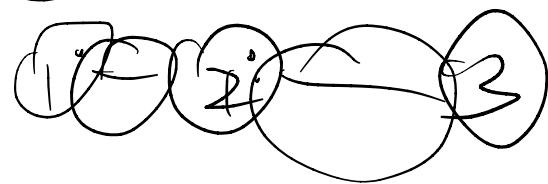
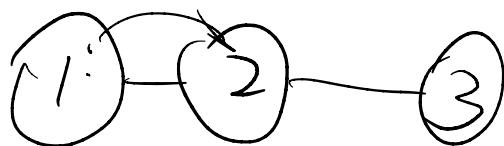
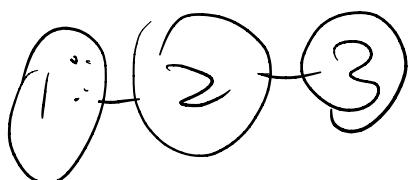
$$1 \times 2 \times 2$$

$$1 + 1 = 2$$

$$3 + 2 = 5$$



3



2+2

2 + 1 + 1 + 1

1 + 1 + 1 + 1

4