Agenda

1. In place Merge sort
2. Quicksort
3. Comparator

Merge 2 sorted subarrays

SC = O(1)

s m m+1 e

```
int[] merge2sub(int[] A, s, m, e) {
/*
   i                  e-s+1          j
   s                  m  m+1         e
                      ↑
*/
int[] C = new int[e-s+1];
i = s      j = m+1      K = 0
while ( i <= m && j <= e) {        // j > e
    if ( A[i] < A[j]) {
        C[K] = A[i]
        K++ , i++
    } else {
        C[K] = A[j]
        K++  j++
    }
}
```

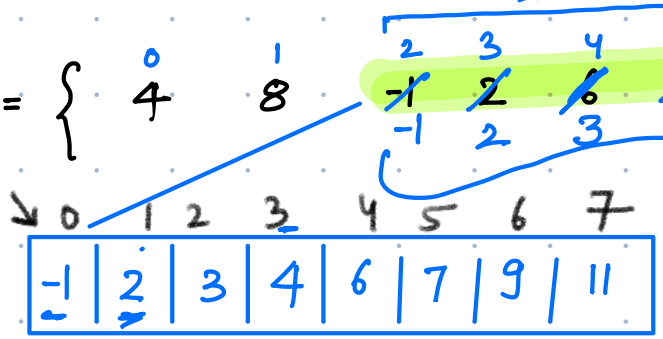
```
while ( i <= m) {        // j > e
    C[K] = A[i]
    K++ , i++

}
while ( j <= e) {        // i > m
    c[K] = A[j]
    K++
    j++

}
```

// C ⟶ A

```
for ( i=0   i < C.length   i++) {
    A[i+S] = C[i];

}
return C;
}
```

S=2

A[ ] = { 4 8 -1 2 6 9 11 3 4 7 13 0 }



A[i+s] = C[i]

$i = 0$ $\quad$ $s = 2$ $\quad$ $C[i]$

$A[0+2]$ $\quad$ $C[0]$

$A[2]$ $\quad$ $C[0]$

$S = 2 + 0$ $\quad$ $= 2$

$2 + 1$ $\quad$ 3

$2 + 2$ $\quad$ 4

$2 + 3$ $\quad$ 5

→ Space

$A[]$ $\quad$ s m e

s m m+1 e

sorted $C[]$

A1 $\quad$ A2 $\quad$ A3

I $\quad$ I $\quad$ I

EXTRA SPACE

0 $\quad$ 0 $\quad$ 0

Inplace Merge sort

	s				m	m+1			e
	0	1	2	3	4	5	6	7	8
$A[] =$	10	15	3	8	6	2	17	12	18

3 6 8 10 15 $\quad$ 2 12 17 18

2 3 6 8 10 12 15 17 18

```
void merge ( int A[], int s, int e){
{ if (s==e) { return; }
  m = (s+e)/2
  merge ( A , s, m);
  merge (A, m+1, e);
  merge2sub ( A, s, m, e);
}
```

Inplace ⟶ making sorting changes in the same array

HW TC

SC

TC : No of functions calls × TC of 1 function call

SC : max no of functions calls × SC of 1 func call

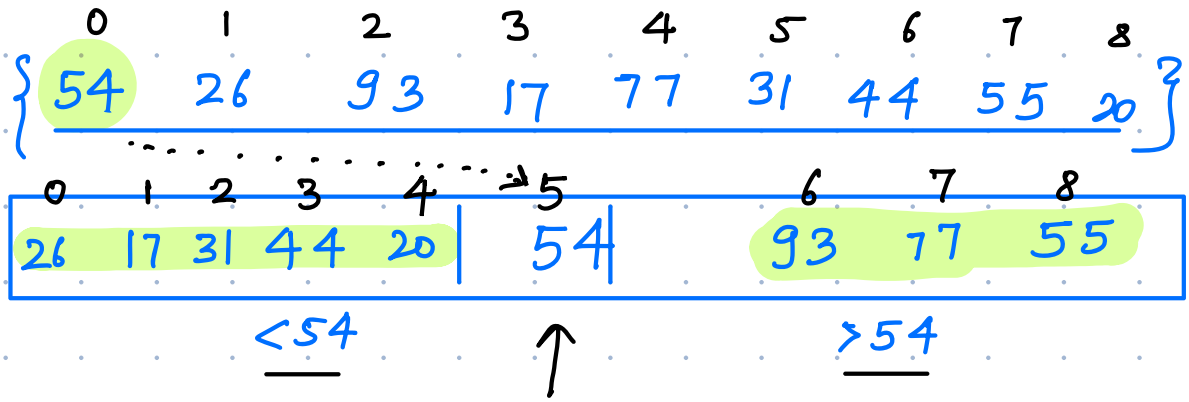
HW

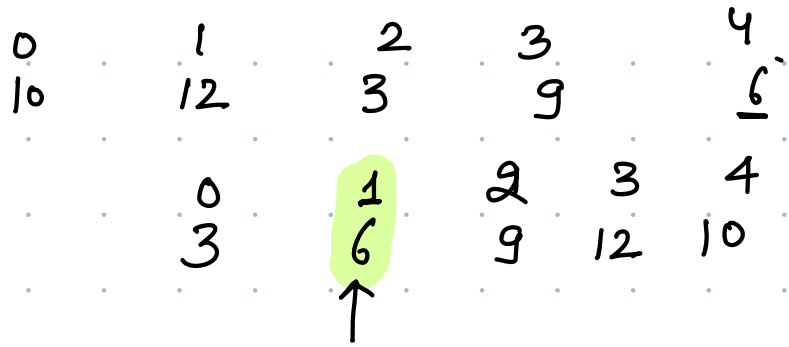
In merge2sub ⟶
Can we make changes in same array A, without use of C

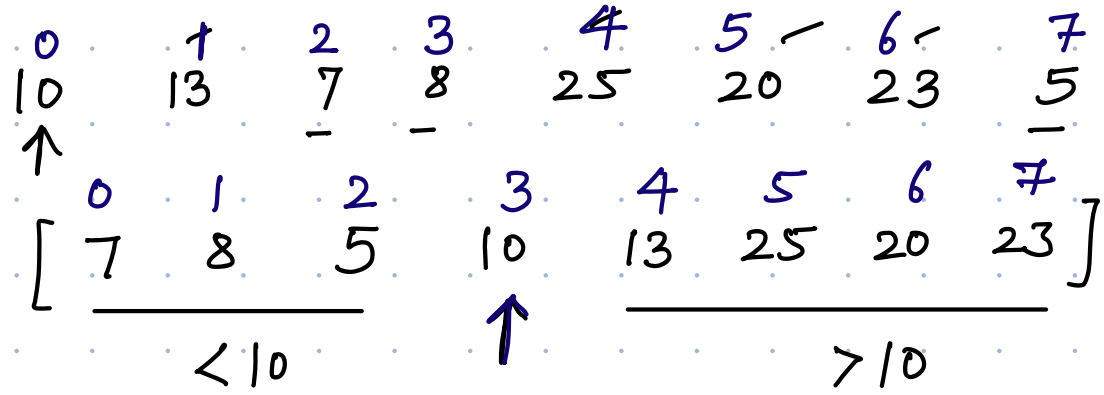
Helper Question :

Given an integer array, consider first element as pivot ⟶ P

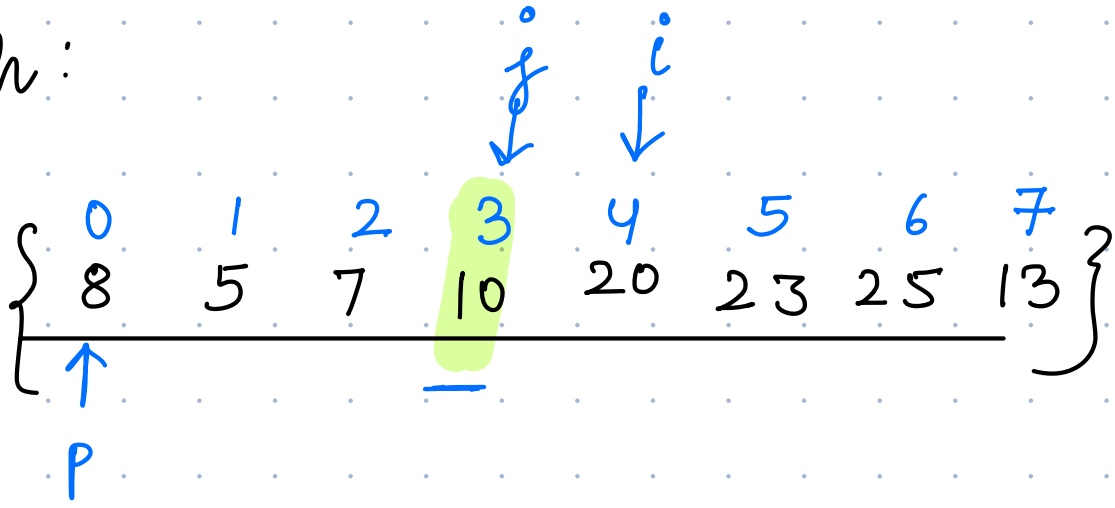
Rearrange the elements around pivot.







Approach :



P = ar[0]

i = 1
j = ar.length - 1

P < ar[i]

Put them in end

P > ar[i]

Put them in start

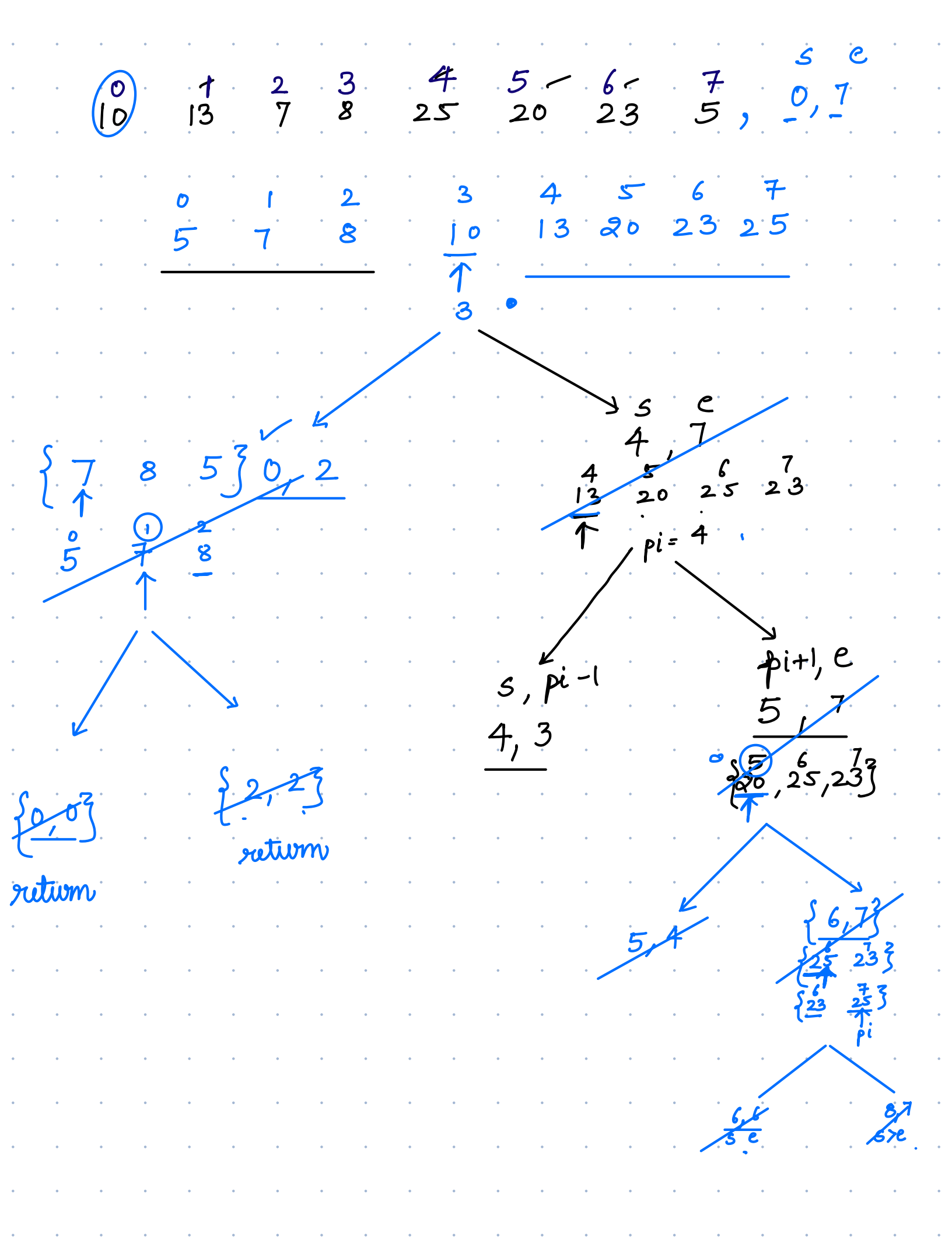
```
int partition ( int[] A, int s, int e ) {
    P = arr[s]

    i = s+1
    j = e

    while ( i <= j ) {
        if ( arr[i] > P ) {
            swap(ar, i, j)
            j--
        } else {
            i++
        }
    }

    → swap (ar, 0, j)
    return j
}
```

QuickSort : Divide and conquer

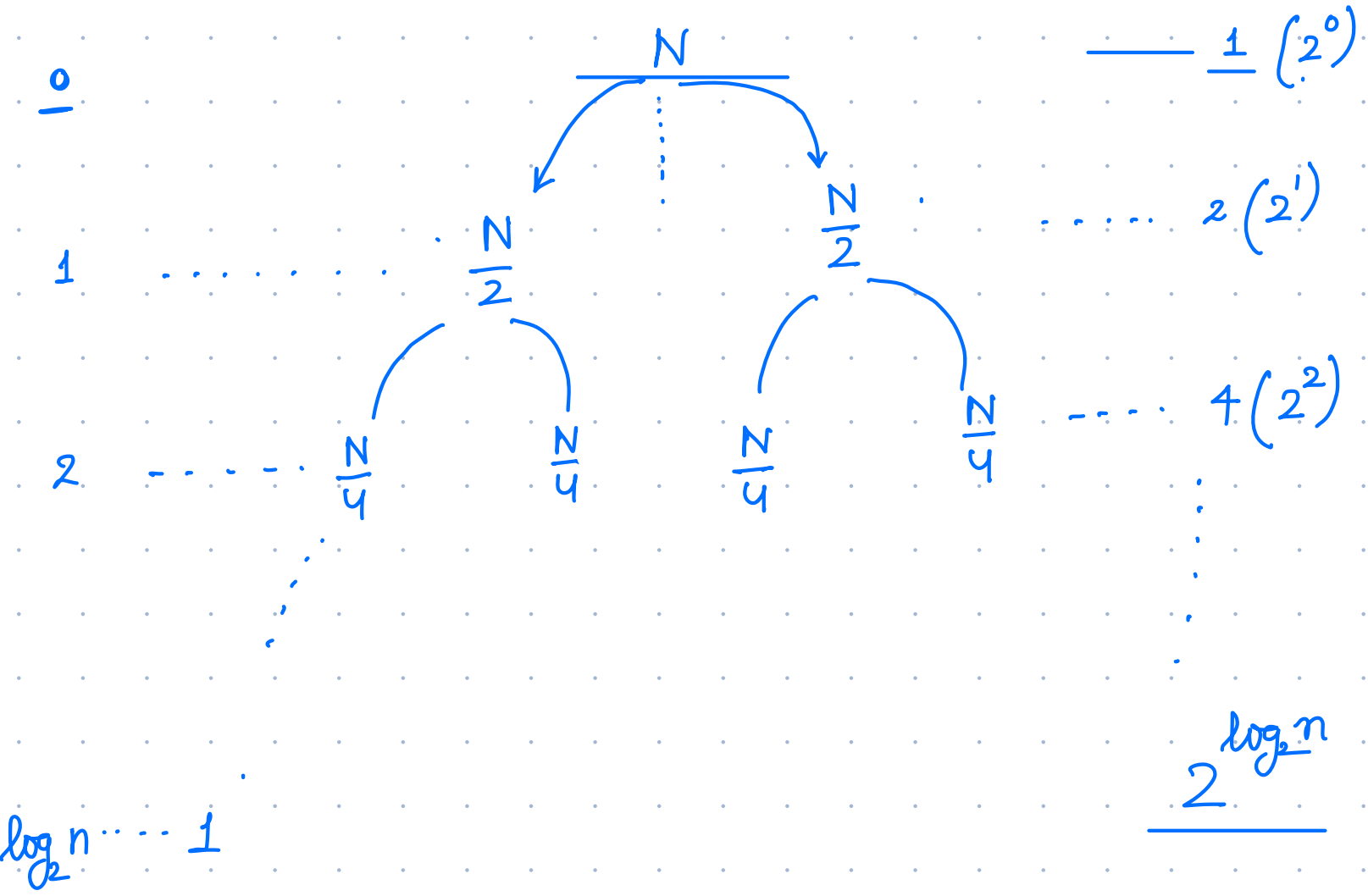
```
void quicksort ( int[] A , int s, int e)
{
    if (s >= e) {
        return;
    }
    pi = partition ( A , s , e)
    quicksort ( A, s , pi-1)
    quicksort (A, pi+1, e)
}
```

{base case} ↓

Main idea


0 1 2 3 4 5 6 7
10 13 7 8 25 20 23 5
s e
0, 7
0 1 2 3 4 5 6 7
5 7 8 10 13 20 23 25
3
7 8 5 0 2
0 1 2
5 7 8
s e
4, 7
4 5 6 7
13 20 25 23
pi = 4
s, pi-1
4, 3
pi+1, e
5, 7
5 6 7
20, 25, 23
0, 0
return
2, 2
return
5, 4
6, 7
25 23
23
25
pi

Quicksort



Total no of Recursive calls

$$= \left[2^0 + 2^1 + 2^2 + \ldots + 2^{\log_2 N} \right]$$

$$= \left[\underset{2^0}{1} + \underset{2^1}{2} + \underset{2^2}{4} + 8 + \ldots + 2^{\log_2 N} \right]$$

GP $= \log_2 N$

$$\text{Sum} = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{1\left(2^{\log_2 N} - 1\right)}{2 - 1} = N - 1$$

Time complexity of 1 recursive call

$$= O(N)$$

$$TC: \quad O(N^2)$$

HW

Time complexity of Quicksort

Comparator }

Java
C++
C#
JS
Python
Kotlin
Ruby

function — compares 2 values
which is bigger, which is smaller

↓

custom comparison

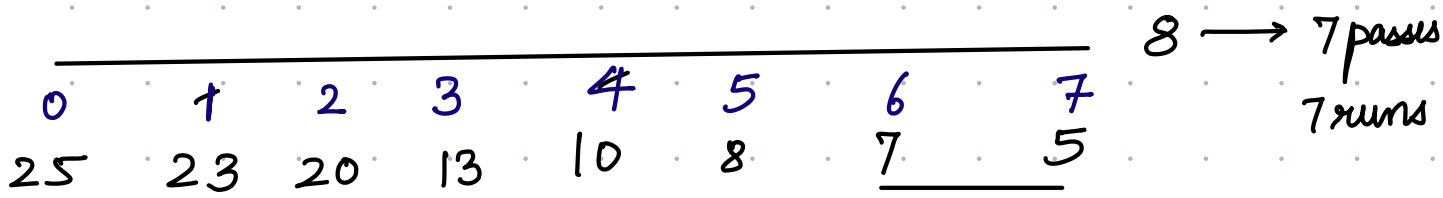
Descending order Sorting

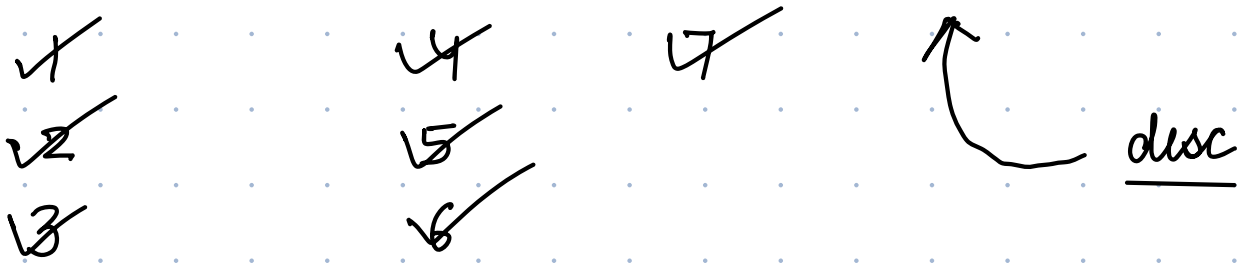
Algorithm :
Bubble sort

Decision : function

f, a →
s b

a < b return 1 → b a
b < a return -1 → a b





Comparator

Java, Python, JS, Ruby, C#

first and second → f s --> -ve value

s f --> +ve value

== --> 0 value

C++

first and second → f s --> true

s f --> false

Algorithm

Engine

Decision function

Driver

CQ 2

$$A[] = \{ \underset{3}{9}, \underset{2}{3}, \underset{4}{10}, \underset{4}{6}, \underset{3}{4} \}$$

Sort on the basis of count of factors, ascending order

If same count of factors, whosoever value is less → first

$$A[] \{ 3 \quad 4 \quad 9 \quad 6 \quad 10 \}$$

Q. $\frac{10}{4}$ $\frac{4}{3}$ $\frac{5}{2}$ $\frac{13}{2}$ $\frac{1}{1}$

$\frac{1}{1}$ $\frac{5}{2}$ $\frac{13}{2}$ $\frac{4}{3}$ $\frac{10}{4}$

Decision function : int a, int b

```
n1 = countfactors (a)
n2 = countfactors (b)
if (n1 < n2)
    return -1
else if ( n2 < n1) {
    return 1
} else {
    if (a < b) { return -1 }
    else { return 1 }
}
```

CQ 3 Ex1 $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ $\begin{bmatrix} -2 \\ 2 \end{bmatrix}$, B = 1

A $\sqrt{10}$ (0,0)

B $\sqrt{8}$ (0,0)

(-2, 2)

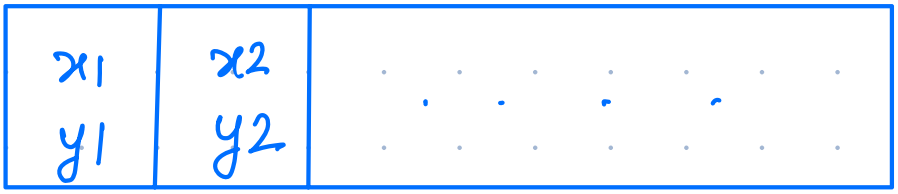
B points closest to origin

(2) $\begin{bmatrix} 3 \\ 3 \end{bmatrix}$ $\begin{bmatrix} 5 \\ -1 \end{bmatrix}$ $\begin{bmatrix} -2 \\ 4 \end{bmatrix}$, B = 2

$\sqrt{18}$ 0,0 $\sqrt{26}$ 0,0 $\sqrt{20}$ 0,0

x, y ——— 0,0

$\sqrt{(x-0)^2 + (y-0)^2} = \sqrt{x^2+y^2}$



Point {
int x
int y
}

Decision function : Point a , Point b

$$d1 = \sqrt{(a.x)^2 + (a.y)^2}$$

$$d2 = \sqrt{(b.x)^2 + (b.y)^2}$$

if (d1 < d2) return -1

else if (d2 < d1) return 1

else return 0

CQ4

Non negative nos in the array

↓

Sort

↓

when you Combine, you get biggest no.

[10, 2]

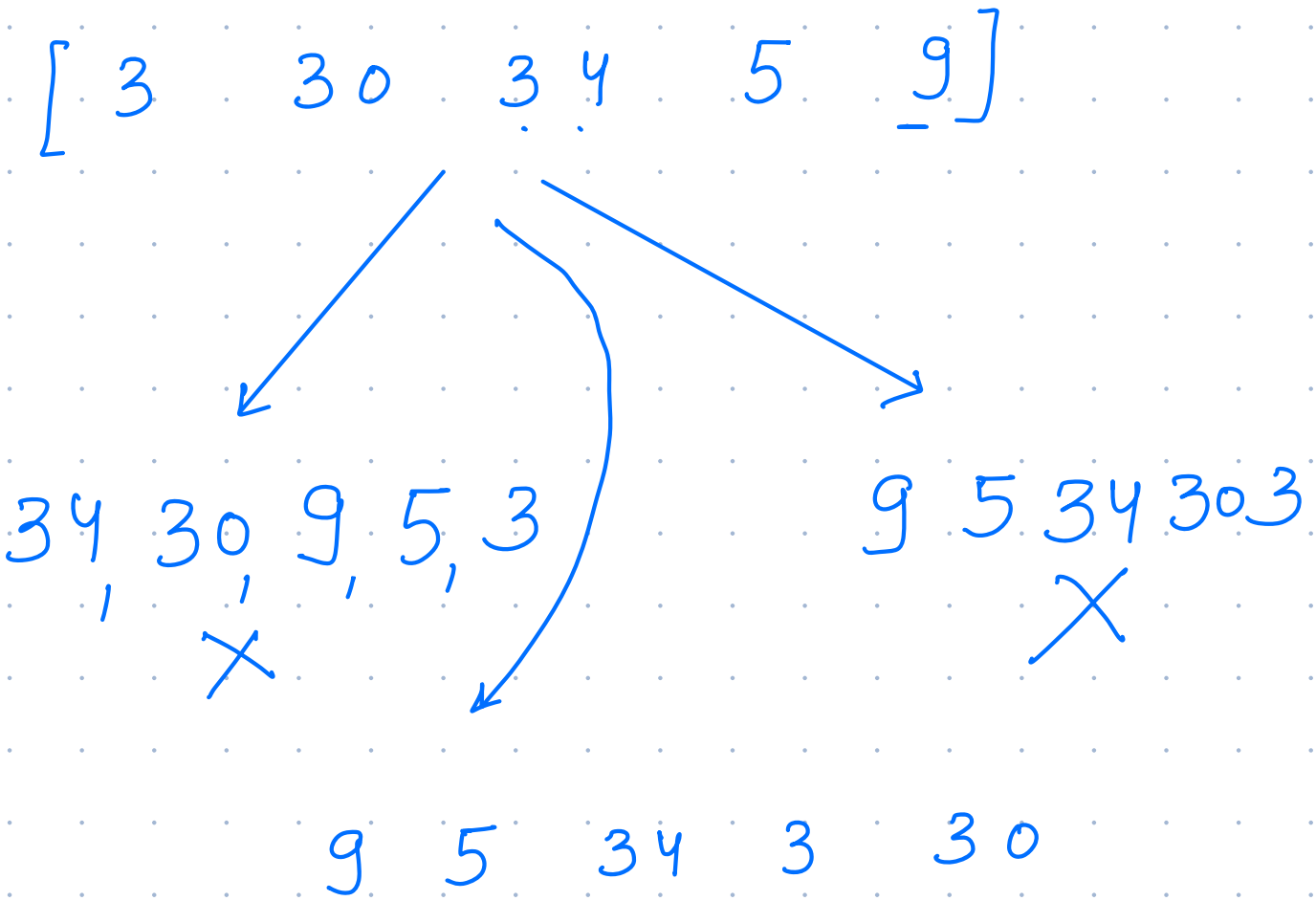
→ 10, 2

102

↓

✓ 2, 10 ans

210



Decision Making

int a int b

a + b > b + a

return -1

b + a > a + b

return 1