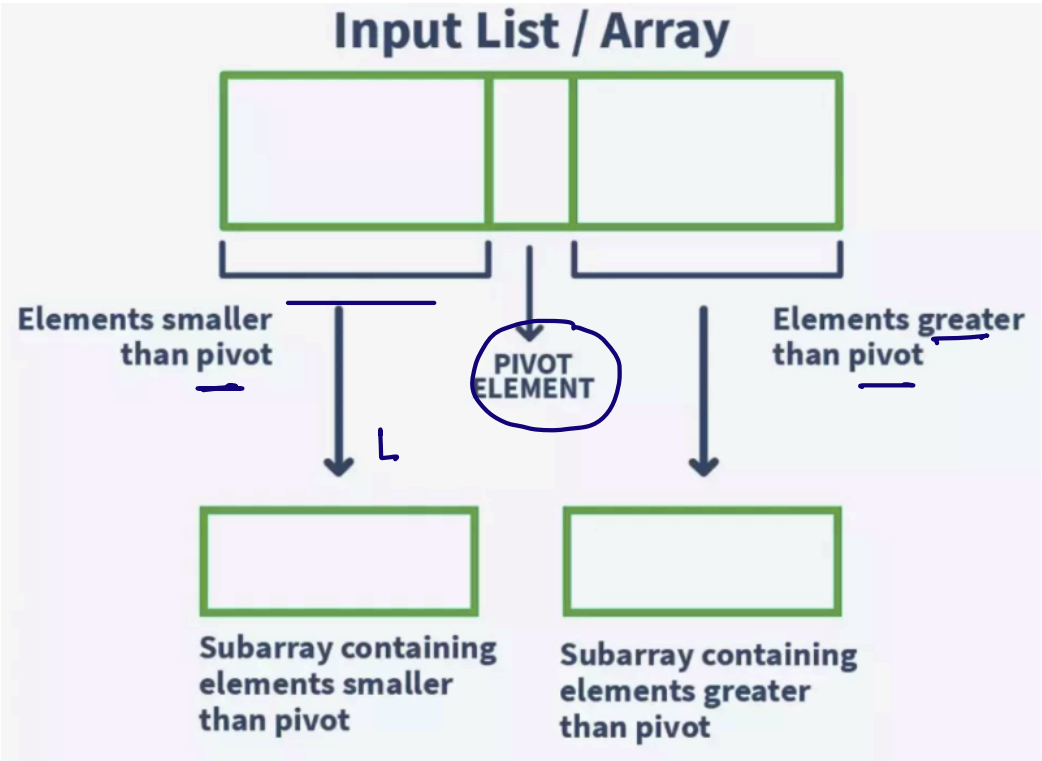
1. Partition of array
2. Quick sort
3. Comparator Problems

Imp

Given an integer array, consider first element as pivot, rearrange the elements such that for all `i`:

if `A[i] < p` then it should be present on left side
if `A[i] > p` then it should be present on right side

Note: All elements are distinct



0	1	2	3	4	5	6	7	8
54	26	93	17	77	31	44	55	20

$\underbrace{26 \quad 31 \quad 44 \quad 20 \quad 17}_{\text{Smaller}} \quad 54 \quad \underbrace{77 \quad 93 \quad 55}_{\text{bigger}}$

achieving it → Collections.sort / Arrays.sort] NlogN

10

if `A[i] > p` then it should be present on right side `A = [10, 13, 7, 8, 25, 20, 23, 5]`

58 users have participated

A	left side = [10, 7, 5, 8] right side = [10, 13, 25, 20, 23]	3%
B	left side = [10, 13, 7, 8] right side = [25, 20, 23, 5]	2%
C	left side = [13, 25, 20, 23] right side = [7, 8, 5]	3%
D	left side = [7, 8, 5] right side = [13, 25, 20, 23]	91%

0	1	2	3	4	5	6	7	8
31	26	20	17	44	54	55	77	93

r l

$\boxed{r+1 \text{ to } N-1}$ = Big

$\boxed{1 \text{ to } l-1/r}$ = Smaller if $p > A[l]$ $l{++}$

else $p < A[l]$

swap(l, r)

r--

left right

```
while (l <= r) {
    if ( p > A[l]) l++
    else {
        swap( l, r)
        r--
    }
}
```

$\boxed{r+1 \text{ to } N-1}$ = Big

$\boxed{1 \text{ to } l-1/r}$ = Smaller

swap(0, r)

$[8, 5, 7, 10, 20, 23, 25, 13]$

r l

$P = A[0] = 10$

```
while (l <= r) {
    if ( p > A[l]) l++
    else {
        swap( l, r)
        r--
    }
}
```

$\boxed{r+1 \text{ to } N-1}$ = Big

$\boxed{1 \text{ to } l-1/r}$ = Smaller

swap(0, r)

until l crosses r

✓ $\underbrace{1 \quad l-1}$ smaller

$\underbrace{r+1 \text{ to } N-1}$ bigger

unknowns

l r

TC = $O(N)$

SC = $\underbrace{O(1)}$ ⇐

Inplace algo

```
while (l <= r) {
    if ( p > A[l]) l++
    else {
        swap( l, r)
        r--
    }
}
```

$\boxed{r+1 \text{ to } N-1}$ = Big

$\boxed{1 \text{ to } l-1/r}$ = Smaller

swap(0, r)

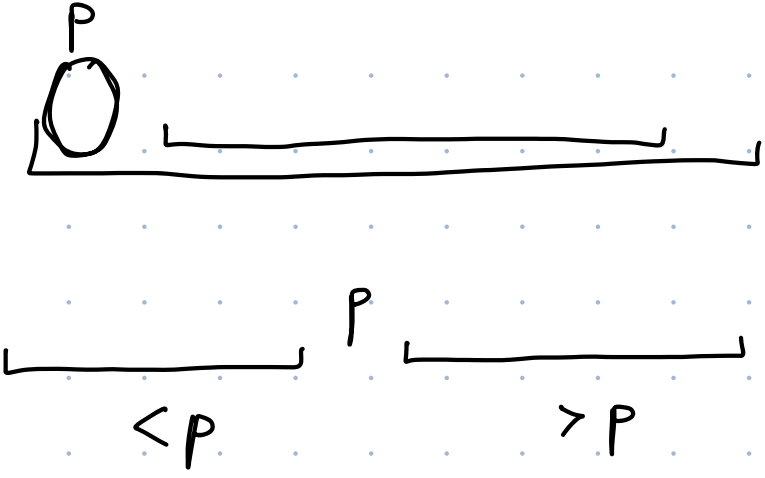
Questions tab

```
while (l <= r) {
    if ( p > A[l]) l++
    else {
        swap( l, r)
        r--
    }
}
```

$\boxed{r+1 \text{ to } N-1}$ = Big

$\boxed{1 \text{ to } l-1 / r}$ = Smaller

swap(0, r)



5	1	6	3	7	9	8
p				r	l	

0	1	2	3	4	5	6	7	8	9
11	12	1	5	8	9	16	6	77	99

i r l j

0 ⟶ i

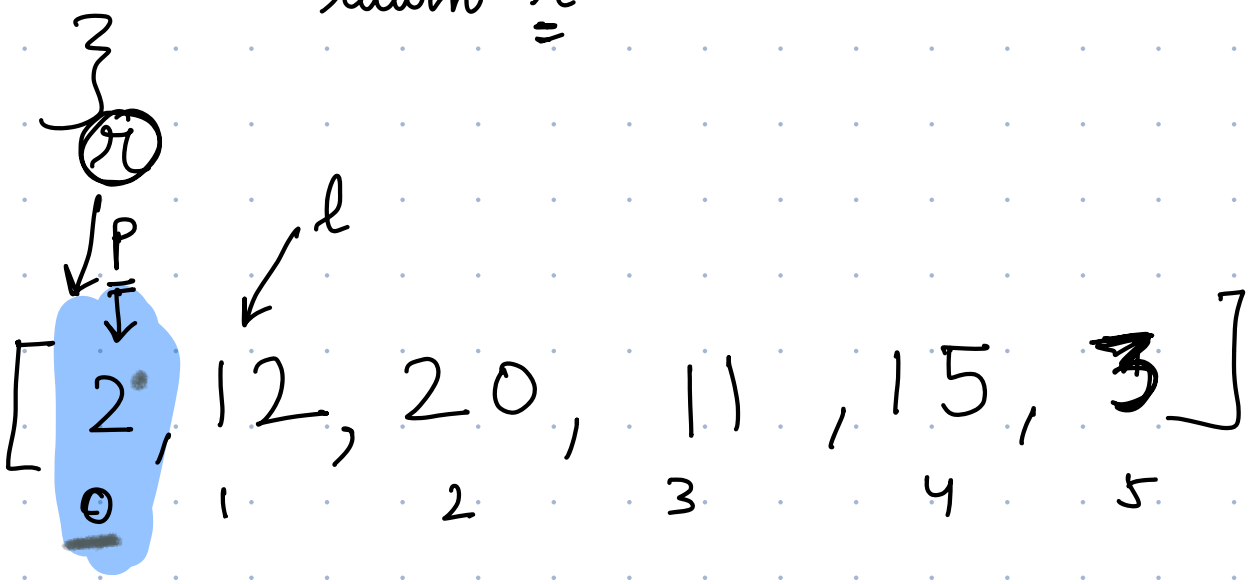
l ⟶ i+1

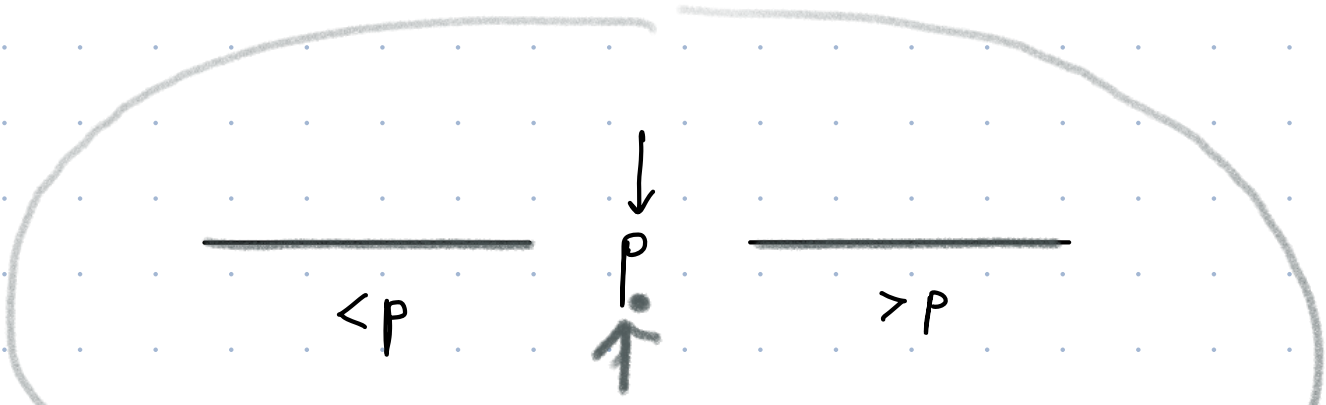
r ⟶ j

```
int pivot (A, i, j) {
    p = A[i]   l = i+1 , r = j
    while (l <= r) {
        if ( p > A[l]) l++
        else {
            swap( l, r)
            r--
        }
    }

    swap( i , r)

    return r
```





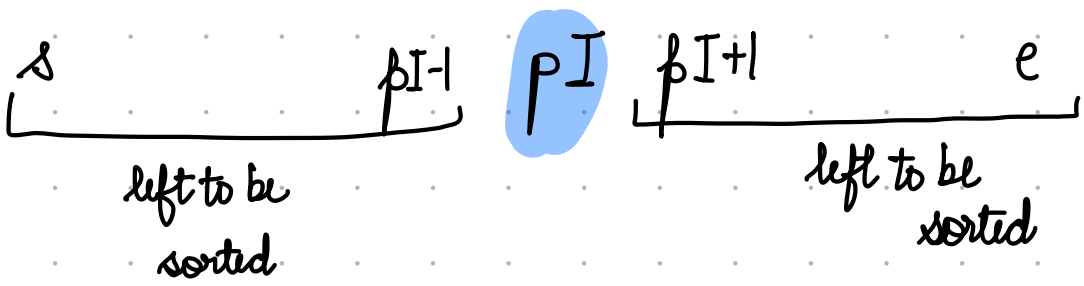
pivot is at its right place

Quicksort —

	s								e
A =	4	1	18	9	3	8	2	19	14

```
void Quicksort ( A , s , e ) {

    pI = pivot (A, s, e)
    Quicksort (A, s, pI-1);
    Quicksort (A, pI+1, e);
}
```



	0	1	2	3	4	5	6	7	8
A =	4 (s)	1	18	9	3	8	2	19	14 (e)

```
void Quicksort ( A , s , e ) {
    if (s > e) return ;
    if (s == e) return ;
    pI = pivot (A, s, e)
    Quicksort ( A, s, pI-1);
    Quicksort ( A, pI+1 , e);
}
```

0	1	2	3	4	5	6	7	8
1	3	2	4	9	8	18	14	19

pi = 3

s to e: 0 to 2

0	1	2
1	3	2

pi = 0

1	3	2

pi = 0

s to pi-1: 0 to -1

s to e: 1 to 2

3	2

1	2
2	3

pi = 2

s to e: 1 to 1

s to e: 3 to 2

s > e

4	—	8						
9	8	18	14	19				
4	5	6	7	8				
8	9	18	14	19				

PI

s to e: 4 to 4

8 8

s == e

s to e: 6 to 8

6	7	8
18	14	19

⇓

6	7	8
14	18	19

pi = 7

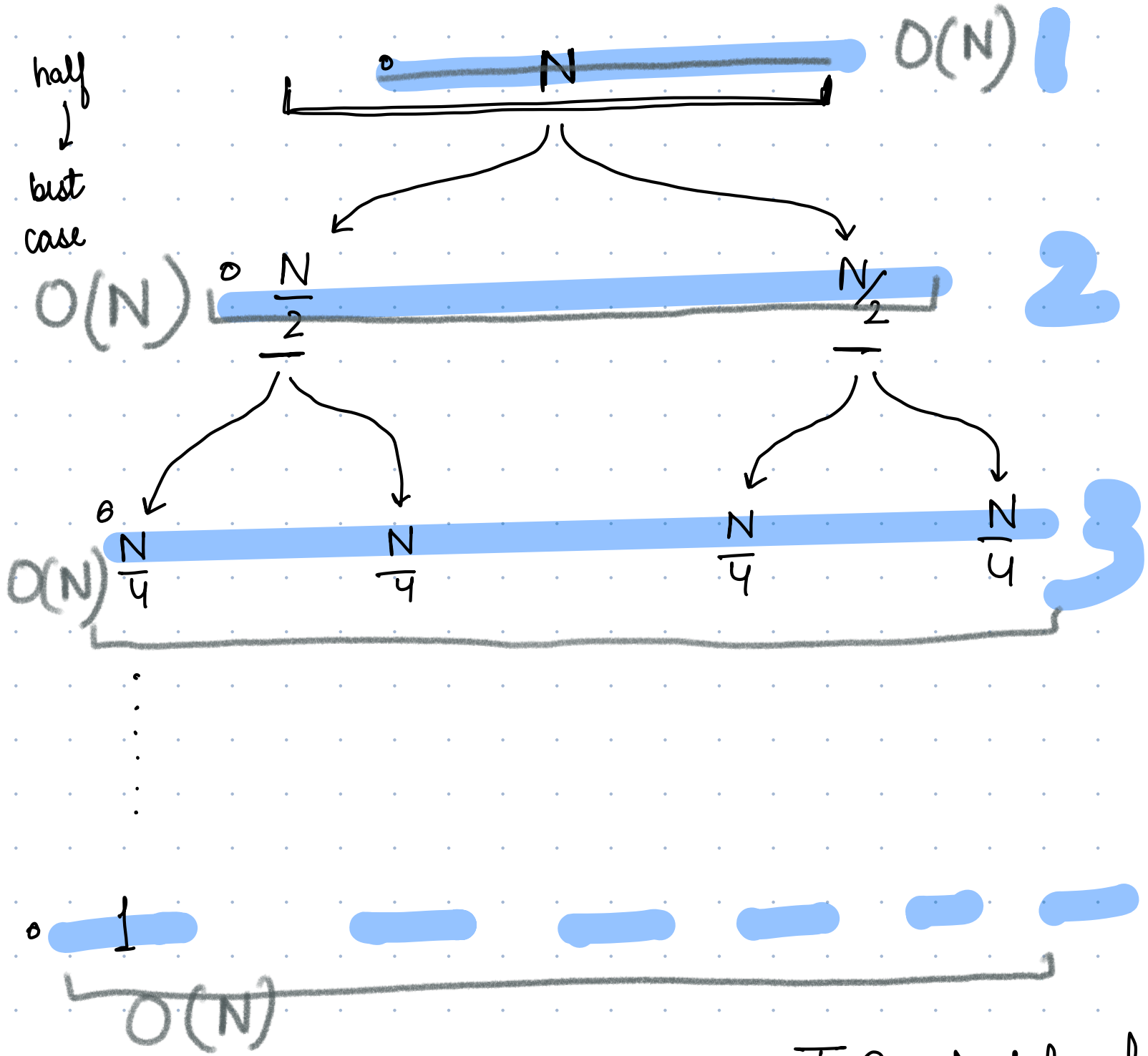
s to e: 6 to 6

s to e: 8 to 8

0	1	2	3	4	5	6	7	8
1	2	3	4	8	9	14	18	19

inplace sort

half → best case



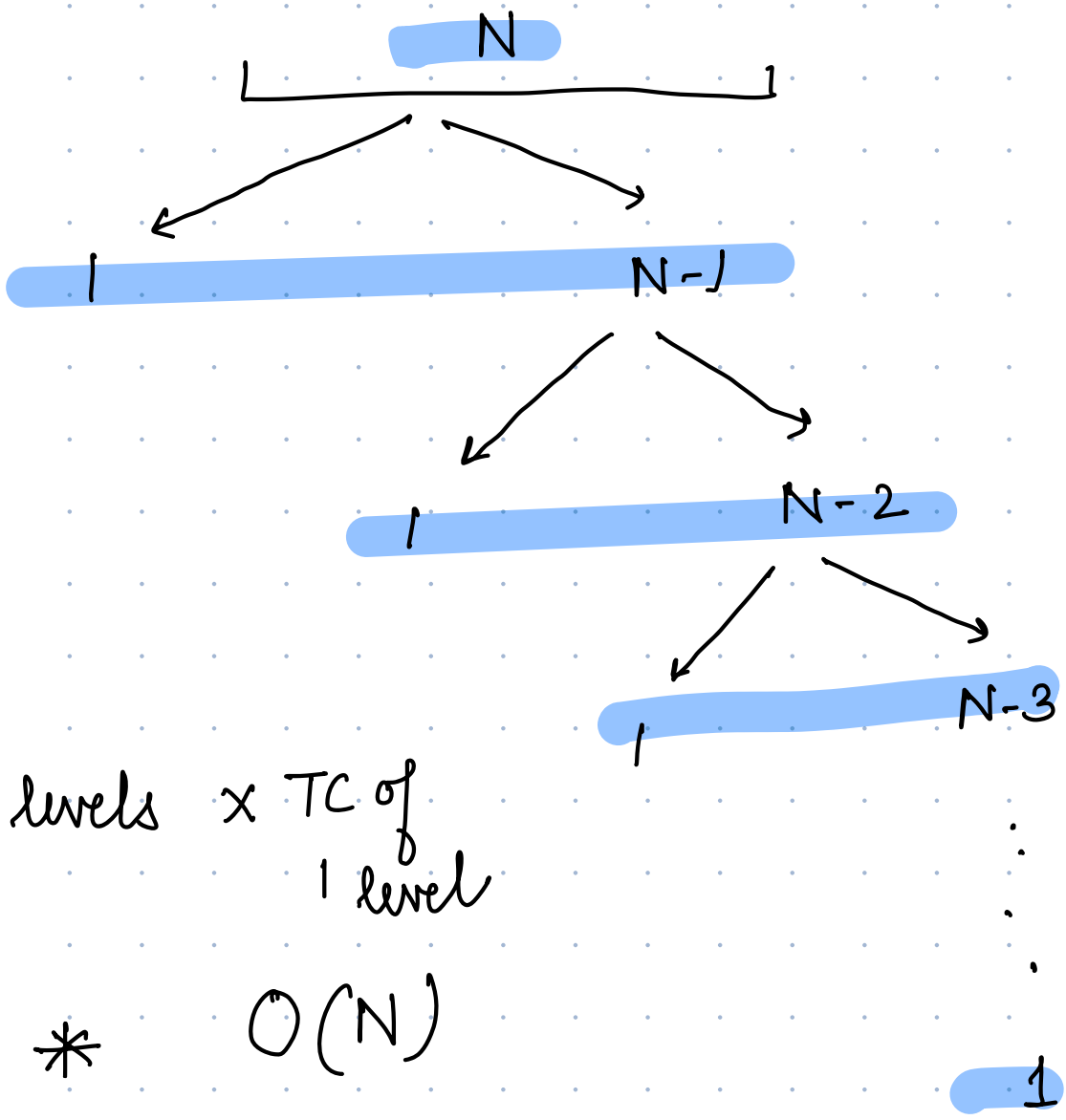
$\log_2 N$ levels * TC of 1 level

$O(N)$

Best Case $O(N * \log N)$

Recursion stack space $= \log N$

Worst Case = Recursion stack space $= N$



Number of levels × TC of 1 level

$N * O(N)$

$O(N^2)$

TC : $O(N \log N)$ to $O(N^2)$

inplace sort same array

Q → do sorting

Arrays.sort()
Collections.sort()

- Bubble
- Merge
- Quicksort

→ to sort an array

Combine 2 sorted feeds

Customize

Comparator

Example 1

```
A[ ] = { 9, 3, 10, 6, 4 }
Ans = { 3, 4, 9, 6, 10 }
```

Explanation:

Total number of factors of 3, 4, 9, 6, 10 are 2, 3, 3, 4, 4.

sort( )

Logic

Array on ranks →

5	6	2	9	10

→ 2 5 6 9 10

smaller element should come first

Array on marks →

50 90 95 86 65 → 95 90 86 65 50

higher element should come first.

Array's sort

sorted array

A	B
96	68
68	96
68	68

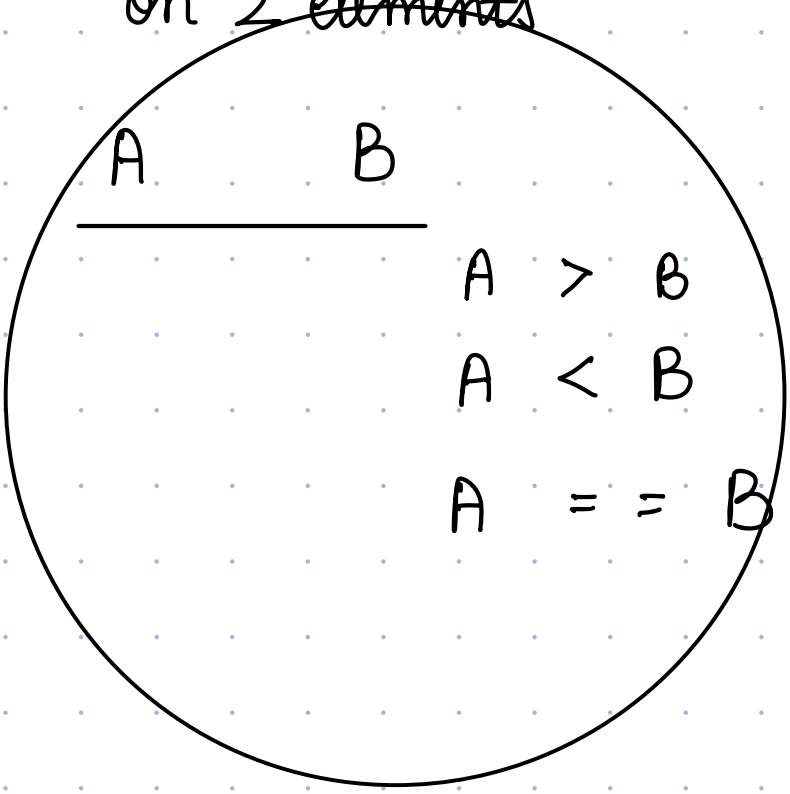
=

A	B
96	68

B	A
96	68

A	B

relative ordering on 2 elements



compare options

↓ input

Arrays.sort()

↓

Output.

Example 1

```
A[ ] = { 9, 3, 10, 6, 4 }
Ans = { 3, 4, 9, 6, 10 }
```

Explanation:

Total number of factors of 3, 4, 9, 6, 10 are 2, 3, 3, 4, 4.

	9	3	10	6	4
count	3	2	4	4	3
	3	4	9	6	10

```
int compare(int A, int B) {
    cf1 = countfactors(A)
    cf2 = countfactors(B)

    if (cf1 < cf2)
        A return -1

    elseif(cf1 > cf2)
        B return 1

    else {
        if (A > B)
            B return 1
        else if (A < B)
            A return -1
        else
            return 0
    }
}
```

For languages - **Java, Python, JS, C#, Ruby**, the following logic is followed.

```
1. In sorted form, if first argument should come before second, -ve value is returned.
2. In sorted form, if second argument should come before first, +ve value is returned.
3. If both are same, 0 is returned.
```

A -ve

A	-ve
B	+ve
A==B	0

[10, 4, 5, 13, 1]

4 3 2 2 1

ans 1 5 13 4 10

Python

```
import functools

//please write the code for finding factors by yourself

def compare(v1, v2):
    if(factors(v1) == factors(v2)):
        if(v1<v2):
            return -1;
        if(v2<v1):
            return 1;
        else
            return 0;
    elif (factors(v1)<factors(v2)):
        return -1;
    else
        return 1;


class Solution:
    def solve(self, A):
        A = sorted(A, key = functools.cmp_to_key(compare))
        return A
```

```java

import java.util.*;

class FactorSort implements Comparator<Integer> {


    public int compare(Integer a, Integer b) {
        int cf1 = factors(a);
        int cf2 = factors(b);


        if(cf1 > cf2){
          return 1;
        }else if(cf1 < cf2){
          return -1;
        }else{
          if(a<b){
            return -1;
          }else if(b<a){
            return 1;
          }else{
            return 0;
          }
        }


    }

    private int factors(int x) {
        int c = 0;

        for(int i=1; i<=x; i++){
          if(x%i==0){
            c++;
          }
        }

        return c;
    }
}

public class Main {
  public static void main(String[] args) {
      ArrayList<Integer> al = new ArrayList<>();
      al.add(10);
      al.add(4);
      al.add(5);
      al.add(13);
      al.add(1);

      Collections.sort(al, new FactorSort());

      for(int i=0; i<al.size(); i++){
        System.out.println(al.get(i));
      }
  }
}
```

C++

```
int factors(int n)
{
    int count = 0;
    int sq = sqrt(n);

    // if the number is a perfect square
    if (sq * sq == n)
        count++;

    // count all other factors
    for (int i=1; i<sqrt(n); i++)
    {
        // if i is a factor then n/i will be
        // another factor. So increment by 2
        if (n % i == 0)
            count += 2;
    }
    return count;
}

bool compare(int val1, int val2)
{
    int cnt_x = count_factors(x);
    int cnt_y = count_factors(y);

    if(factors(val1) == factors(val2))
    {
        if(val1<val2)
        {
            return true;
        }
        return false;
    }
    else if(factors(val1)<factors(val2))
    {
        return true;
    }
    return false;
}

vector<int> solve(vector<int> A) {
    sort(A.begin() , A.end() , compare);
    return A;
}
```

A B

A true

B false

Arrays.sort() | Collections.sort()

✓ [1, 3, 2] → [1, 2, 3]

✓ [1, 10, 100] $\xrightarrow[\text{sort}]{\text{asc}}$ [1, 10, 100]

1 10 100

100 10 1

[3, 30] $\xrightarrow{\text{asc sorting}}$ ✗ [3, 30]

→ [30, 3]

⇒ [3, 30, 34, 5, 9]

30 3 34 5 9

[10, 5, 2, 8, 200]

✓ 10 200 2 5 8

✗ 200 10 2 5 8

[10, 5, 2, 8, 200]

⇓

compare options

A B

90 9 ⇒ A B: 90 9 / B A: 9 90 — AB

89 4 ⇒ AB: 89 4 / BA: 4 89 — BA

concat

AB > BA — smallest: B.

BA > AB — A.

Smallest

largest no.

A, B

concat

AB > BA A

BA > AB B

Doubts :

Given an array A of size n, sort the data in ascending order of count of factors, if count of factors are equal then sort the elements on the basis of their magnitude. `A = [10, 4, 5, 13, 1]`

4 options

Active Duration(Most preferred: 30 seconds)

Appears for	30 Secs

A	[1, 4, 5, 10, 13]
B	[1, 5, 13, 4, 10]
C	[13, 10, 4, 5, 1]
D	[1, 4, 5, 13, 10]

[10, 4, 5, 13, 1]

⇒ 4 3 2 2 1

Java

```
//please write the code for finding factors by yourself

public ArrayList<Integer> solve(ArrayList<Integer> A) {
    Collections.sort(A, new Comparator<Integer>(){
        @Override
        public int comp(Integer v1, Integer v2){
            if(factors(v1) == factors(v2)){
                if(v1<v2) return -1;
                else if(v2<v1) return 1;
                return 0;
            }
            else if(factors(v1)<factors(v2)){
                return -1;
            }
            return 1;
        }
    });
    return A;
}
```