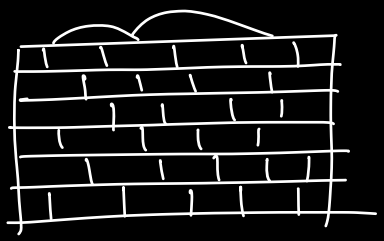


What is sorting ?

Arranging data on the basis of some specific parameters following some specific rules.

1	2	3	4	5

→ Asc order by value

5	4	3	2	1

→ Desc order by value

7	2	4	9	6

→ Asc order by no of factors.

Why to sort ? → Makes searching easy & fast.

Name	12th %
Ankur	83
Ashish	71
Nikhil	95
Rohit	70
Navneet	59
Kamal	73
Abhay	93
Poornima	93
Kshitij	86

Sort ASC by 12th %

Name	12th %
Navneet	59
Rohit	70
Ashish	71
Kamal	73
Ankur	83
Kshitij	86
Abhay	93
Poornima	93
Nikhil	95

Stable : If two data points have same value then their relative order in the initial data should be maintained.

7 (2) 3 4 (2) 8 1

Stable: 1 2 2 3 4 7 8.

Non Stable: 1 2 2 3 4 7 8.

Inplace Algo : Sorting without using any extra space.
SC : O(1)

Q Given an array of N elements. Find the K^{th} minimum.

$K < \log N$

1	5	-1	2	10	3

K = 3 ⟶ 2

K = 5 ⟶ 5

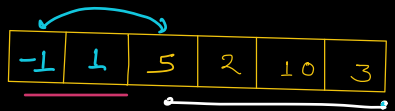
Approach 1

- Sort the array
- Ret A[K-1]

TC: O(N log N)

SC: Depends on sorting algo

Approach 2



i = 0 . Select the 1st min ⟶ Iterate from 0 to N-1 & find min swap(0, indMin) ⇒ O(N)

i = 1 . Select the 2nd min ⟶ Iterate from 1 to N-1 & find min swap(1, indMin) ⇒ O(N) (N-1)

i = 2 . Select the 3rd min ⟶ Iterate from 2 to N-1 & find min swap(2, indMin) O(N) (N-2)

⋮

Select the K^{th} min ⟶ . . . .

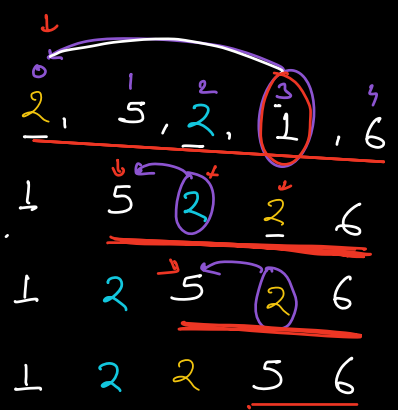
TC: O(KN) SC: O(1)

Repeat this N times ⟶ Sorted array.

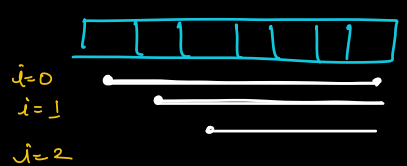
Selection Sort

TC : $O(N^2)$ $(N + (N-1) + (N-2) \text{ ---- } + 1) \Rightarrow \frac{N \times (N+1)}{2}$

SC : $O(1)$




```
for (i=0; i<N; i++){
    min = A[i], ind = i;
    for(j=i; j<N; j++){
        if (A[j] < min){
            min = A[j];
            ind = j;
        }
    }
    swap(A[i], A[ind]);
}
```



In-place ⟶ Yes

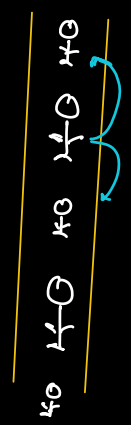
Stable ⟶ No

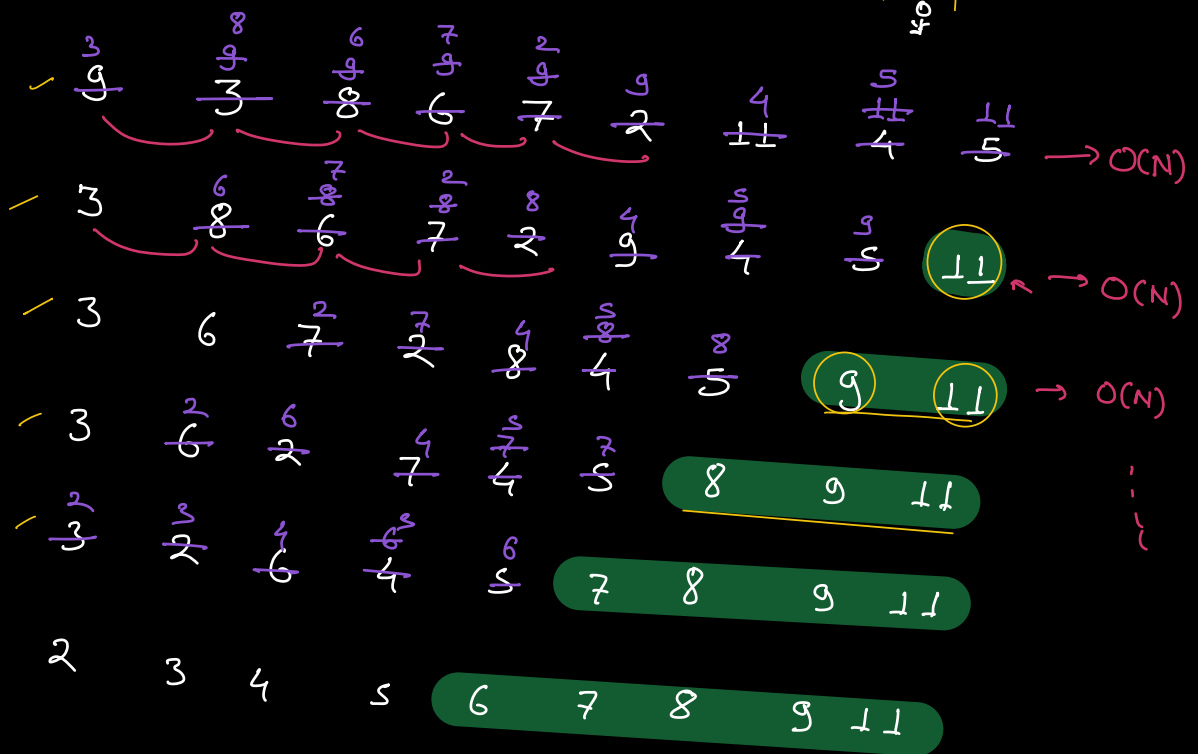
#Swaps ⟶ (N-1)

HW : Write a stable implementation of selection sort.

SC : O(1)

Q Given an array of N elements.
Swapping of non-consecutive indices is not allowed.
Sort the array in ASC order.

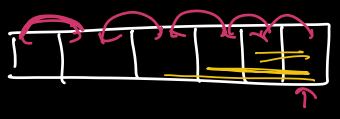




TC : $O(N^2)$
SC : $O(1)$

i	j
0	N-2
1	N-3
2	N-4
4	N-5
⋮	⋮

```
for (i=0; i< N; i++) {
    for (j=0; j<(N-1)-i ; j++) {
        if (A[j] > A[j+1]) {
            swap(A[j], A[j+1]);
        }
    }
}
```



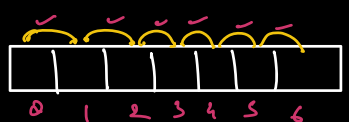
TC : $O(N^2)$
SC : $O(1)$

Optimizations :
- Count no of swaps in inner loop
- if swaps == 0
  - → array is already sorted
  - → Break out.

Stable ⟶ Yes

Inplace ⟶ Yes

Swaps ⟶ $\frac{N(N-2)}{2} \Rightarrow O(N^2)$



i=0 ⟶ (N-1)
i=1 ⟶ N-2
i=2 ⟶ N-3
⋮
i=N-1 ⟶ 0

$(N-1)+(N-2)+(N-3)+\cdots+1+0$

$\Rightarrow \frac{N(N-1)}{2}$

Q Amazon Whatsfin

Given 2 sorted arrays of size N & M. Merge them & return a new sorted array.

A : 2, 5, 7, 12, 20, 24, 29

B : 6, 9, 10, 14, 18, 19

C : 2, 5, 6, 7, 9, 10, 12, 14, 18, 19, 20, 24, 29

A : 2, 5, 7, 12, 20, 24, 29

B : 6, 9, 10, 14, 18, 19

{ C[N+M] $\Longrightarrow O(N+M)$

SC

Input + Space created in program

2, 5, 6, 7, 9, 10, ... 19 _ _ _

```
int [] merge ( A[], N, B[], M) {
    ⇒ C[N+M] ⟶ New Array
    a = 0 ; b = 0; c = 0;
    while ( a < N && b < M) {
        if ( A[a] < B[b] ){
            C[c] = A[a];
            a++;
        }
        else {
            C[c] = B[b];
            b++ ;
        }
        c++;
    }

    while (a < N){
        C[c] = A[a];
        a++; c++;
    }

    while (b < M){
        C[c] = B[b];
        b++; c++;
    }

    ret c;
}
```

TC : O(N+M)

SC : O(N+M)

Linear

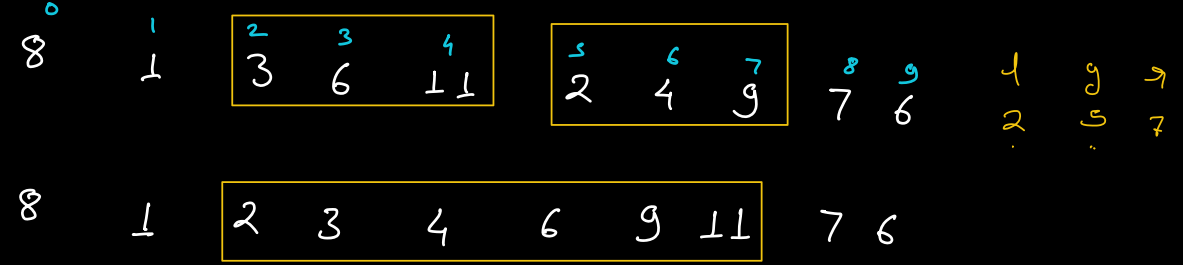
Q Given an array of size N & 3 indices l, y, r.

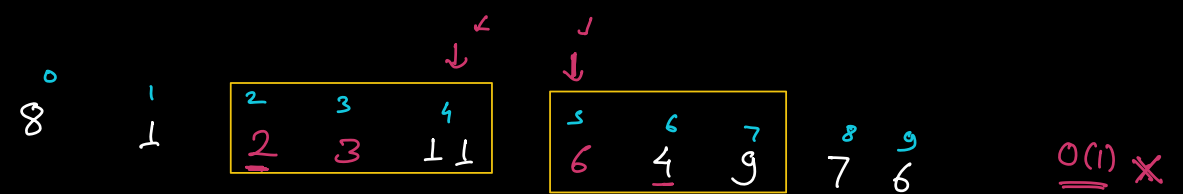
Given that subarray from l, (y-1)

& subarray from y, r

are sorted in ASC

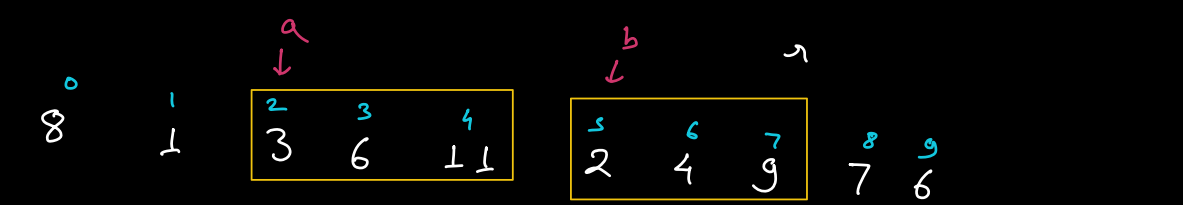
Sort the subarray from l to r. in ASC order



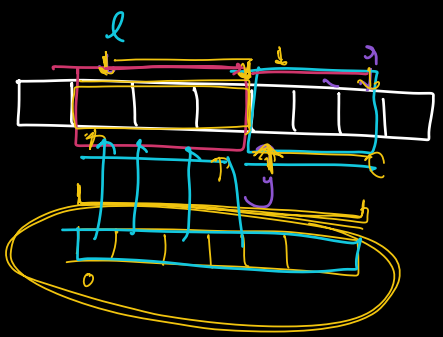


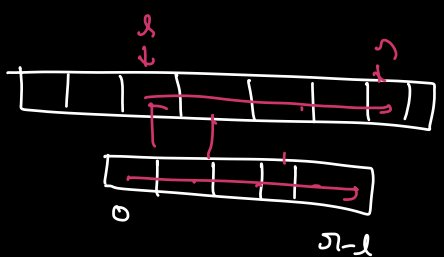
O(1) ✗

Can not be solved in O(1)SC & O(N) TC. ←



```
int [] merge ( A[], l, y, r ) {

    C[r-l+1] ⟶ New Array
⇒ a = l ; b = y; c=0;
    while ( a<y && b≤r) {

        if ( A[a] < A[b] ){
            C[c] = A[a];
            a++;
        }
        else {
            C[c] = A[b];
            b++ ;
        }
        c++;
}

    while (a<y){
        C[c]= A[a];
        a++; c++;
    }

    while (b≤r){
        C[c]= A[b];
        b++; c++;
    }
    // Copy values from C to original array.
    for ( i=0; i<(r-l+1); i++){
        A[i+l] = C[i];
    }
}
```





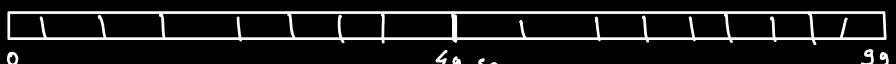
TC : (r-l+1) ⟶ O(N)

SC : (r-l+1) ⟶ O(N)

{ Bubble Sort → $O(N^2)$ →
  Selection Sort → $O(N^2)$ →

$(100)^2$

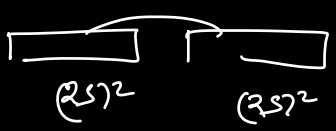
10,000

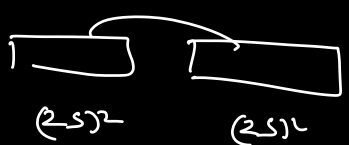


$(50)^2$ $(50)^2$

2500 + 2500 + 100

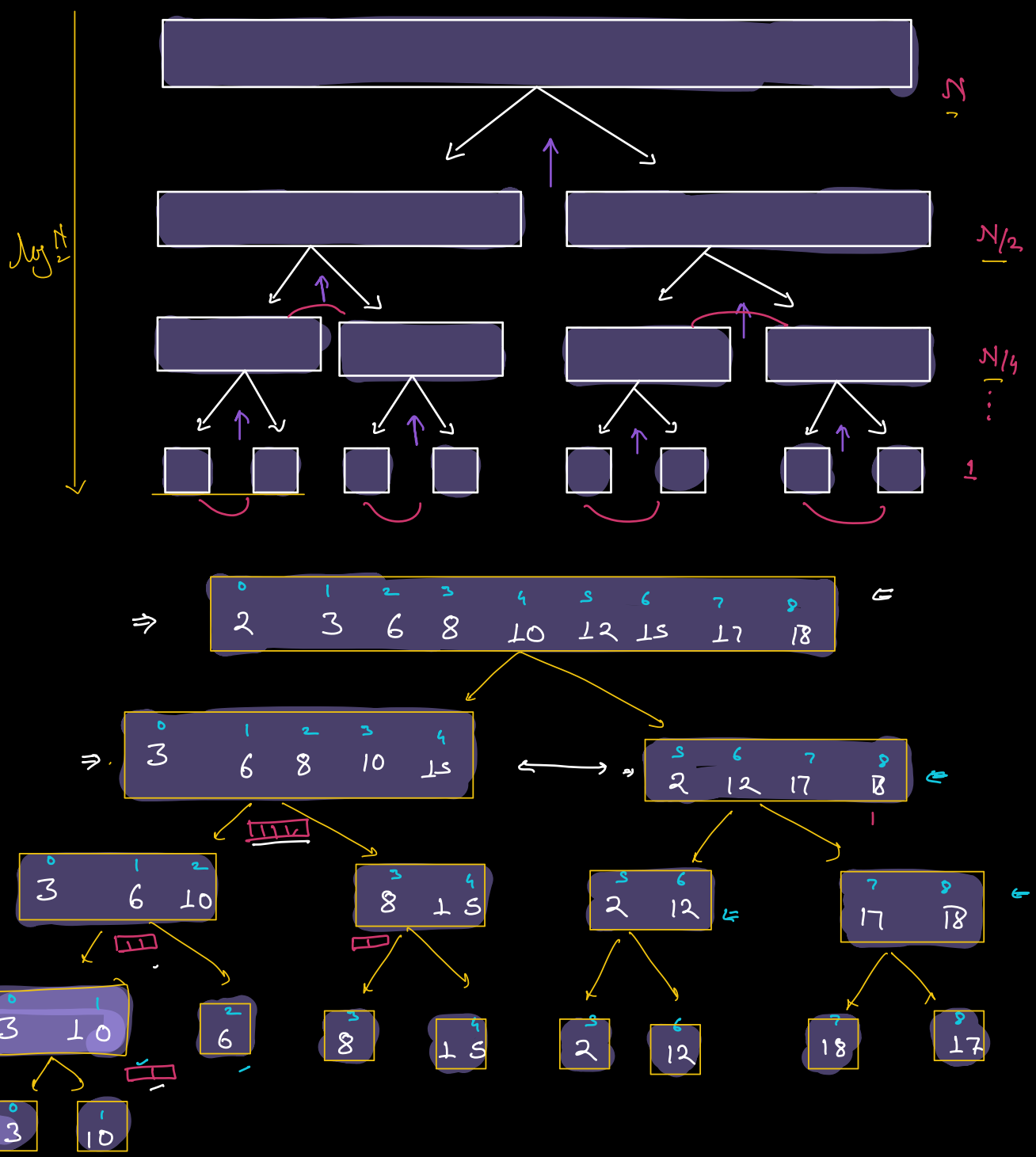
5100



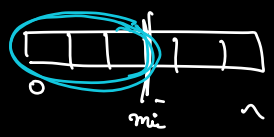


$4 \times (25)^2 + 200$

≈ 2700


log N
N
N/2
N/4
1
0 1 2 3 4 5 6 7 8
2 3 6 8 10 12 15 17 18
0 1 2 3 4
3 6 8 10 15
5 6 7 8
2 12 17 18
0 1 2
3 6 10
3 4
8 15
5 6
2 12
7 8
17 18
0 1
3 10
2
6
3
8
4
15
5
2
6
12
7
18
8
17
0
3
1
10

```
void mergeSort ( A[], l, r ) {
    if ( l == r) ret;
// Assumption : merge(A, l, r) sorts the subarray in A
                from l to r
    mid = (l+r)/2;
    mergeSort ( A, l, mid); ⟶ N/2
    mergeSort ( A, mid+1, r); ⟶ N/2
⇒   merge ( A, l, mid+1, r); ⟶ O(N) / O(N)
}
```



$$T(N) = 2\,T(N/2) + O(N)$$ → Solve it using substitution

TC : $O(N \log N)$

SC : $O(N + \log N)$

$\Rightarrow O(N)$

In-Place ⟶ No

Stable ⟶ ?

Comparisons ⟶ ?

HW ⟶ (Make it stable)