

Q.1 Given an Array of size N , find the k^{th} minimum element. ($k < \log N$)

$A: \{1, 5, -1, 2, 10, 3\}$

$k=2 \Rightarrow 1$

$k=3 \Rightarrow 2$

$k=5 \Rightarrow 5$

Idea 1:- Sort the array & return $A[k-1]$

$A: \{ \overset{0}{1}, \overset{1}{5}, \overset{2}{-1}, \overset{3}{2}, \overset{4}{10}, \overset{5}{3} \}$

↓ sort

$\{ \overset{0}{-1}, \overset{1}{1}, \overset{2}{2}, \overset{3}{3}, \overset{4}{5}, \overset{5}{10} \}$

→ $A[k-1]$

TC: $O(N \log N)$

⇒ $A: \{-1, 4, 2, 3, 1, 2, 5, 2\}$

$A: \{ \overset{0}{1}, \overset{1}{5}, \overset{2}{-1}, \overset{3}{2}, \overset{4}{10}, \overset{5}{3} \}$

$k=4$

$\{ \overset{0}{-1}, \overset{1}{5}, \overset{2}{1}, \overset{3}{2}, \overset{4}{10}, \overset{5}{3} \}$

$\{ \overset{0}{-1}, \overset{1}{1}, \overset{2}{5}, \overset{3}{2}, \overset{4}{10}, \overset{5}{3} \}$

{ -1 1 2 5 10 3 }

{ -1 1 2 3 10 5 }

4th
Min

TC: $O(K \cdot N) < O(N \log N)$

SC: $O(1)$

A: { ⁰1, ¹5, ²-1, ³2, ⁴10, ⁵3 }

{ -1, 5, 1, 2, 10, 3 }

{ -1, 1, 5, 2, 10, 3 }

{ -1 1 2 5 10 3 }

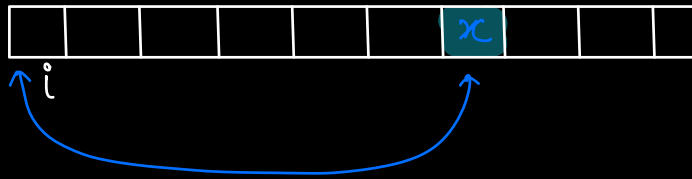
{ -1 1 2 3 10 5 }

{ -1 1 2 3 5 10 }

* If we repeat the above process N times, we'll get SORTED array.

⇒ Selection Sort

Select the minimum element & replace it with its correct position element.



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

A : { ⁰1, ¹5, ²-1, ³2, ⁴10, ⁵3 }

i = 0

m = 1

ind = 2

↓
{ ⁰-1, ¹5, ²1, ³2, ⁴10, ⁵3 }

{ ⁰-1, ¹1, ²5, ³2, ⁴10, ⁵3 }

TC: $O(N^2)$

SC: $O(1) \Rightarrow$ Inplace.

1) Inplace: SC: $O(1)$
 \hookrightarrow NO Extra space.

2) Stable:
 $\{1, 4, 2, 3, 2, 6, 5\}$
 \downarrow sort \Rightarrow Stable.
 $\{1, 2, 2, 3, 4, 5, 6\}$

duplicate

\rightarrow If the relative order of the elements isn't changing after sorting then that Algo is called Stable Sorting algo.

Quiz Maximum no. of swaps in Selection Sort.



A: $\{2, 5, 2, 1, 6\}$
 \downarrow i j
 $\{1, 5, 2, 2, 6\}$
 \downarrow i j
 $\{1, 2, 5, 2, 6\}$
 \downarrow i $\underline{\underline{ind}}$

Unstable

A : { 1 3 2 5 2 6 }

$\begin{matrix} \swarrow & \searrow \\ i & j \end{matrix}$

{ 1 2 3 5 2 6 }

$\begin{matrix} \swarrow & \searrow \\ i & j \end{matrix}$

{ 1 2 2 5 3 6 }

HW Try to implement stable version of Selection Sort.

#

⚭ ⚭ ⚭ ⚭ ⚭ ⚭

Swapping the pos with non consecutive elements isn't allowed.

	0	1	2	3	4	5	6	7	8
{	9	3	8	6	7	2	11	4	5
	3	9	9	9	9	9	4	11	11
		8	6	7	2			5	

{	3	8	6	7	2	9	4	5	11
		6	8	8	8	4	9	9	
			7	2			5		

{	3	6	7	2	8	4	5	9	11
			2	7	4	8	8		
						5			

{	3	6	2	7	4	5	8	9	11
---	---	---	---	---	---	---	---	---	----



⇒ If we repeat the above process N times, we'll get sorted array.

⇒ BUBBLE SORT

for($i=0$; $i < N$; $i++$) {

for($j=0$; $j \leq N-i-2$; $j++$) {

if ($A[j] > A[j+1]$)

swap($A[j]$, $A[j+1]$);

3

3

$i=0$

A :	⁰	¹	²	³	⁴	⁵	⁶	⁷	⁸	
	4	3	8	6	7	2	11	4	5	3
	3	4	7	7	7	9	4	11	5	
		8	6	7	2					

$j \rightarrow [0, N-2]$

$i=1$

							^{N-3}	^{N-2}	^{N-1}	
							4	5	11	
							4	9		
							5	$j+1$		
							j			

$j \in [0, N-3]$

$i=2$

									4	11
									8	

$j \in [0, N-4]$

$j \in [0, N-i-2]$

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



i	j
0	$[0, N-2] \Rightarrow \underline{\underline{N-1}}$
1	$[0, N-3] \Rightarrow N-2$
	⋮

$$N-1 \Rightarrow 0$$

$$(N-1) + (N-2) + \dots + 1$$

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

$$SC: O(1) \Rightarrow \underline{\underline{Inplace.}}$$

$A :$	{	2	5	2	1	6	}
			↓				
$A :$	{	1	2	2	5	6	}

}

Stable.

	0	1	2	3	4	5	6	7	8	
{	9	3	8	6	7	2	11	4	5	}
	3	9	9	9	9	9	4	11	11	
		8	6	7	2			5		
					↓					
{	3	8	6	7	2	9	4	5	11	}
		6	8	8	8	4	9	9		
			7	2			5			
					↓					
{	3	6	7	2	8	4	5	9	11	}
			2	7	4	8	8			
					↓					
{	3	6	2	7	4	5	8	9	11	}
		2	6	4	7	7				
					↓					
{	3	2	6	4	5	7	8	9	11	}
	2	3	4	6	6					
					↓					
{	2	3	4	5	6	7	8	9	11	}

⇒ In any iteration, if there's NO swap happening then Array has already become sorted.

Q: Merge 2 Sorted Arrays.

⇒ Given 2 sorted arrays of size M & N , Merge them into a single sorted array.

A: { 2 5 7 12 20 24 29 } → M

B: { 6 9 10 14 18 19 } \rightarrow 21

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

$$\begin{array}{l} \rightarrow \\ M+N \\ \hline \hline \end{array}$$

```
int C[N+M]
```

$$O(M) + O(N) + O((M+N) \log(M+N))$$

A: { 2 5 7 12 20 24 29 } M=7
i

B: { 6 9 10 14 18 19 } $N = 6$

$$C[k] = \min(A[i], B[j])$$

C:

0	1	2	3	4	5	6	7	8	9	10	11	12
2	5	6	7	9	10	12	14	18	19			

```
int merge ( A[], M, B[], N) {
```

```
    C[N+M];
```

```
    i=0, j=0, k=0
```

```
    while ( i < M && j < N) {
```

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

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

```
            i++, k++
```

$C[k++] = A[i++]$

```
        }
```

```
    } else {
```

```
        C[k] = B[j];
```

```
        j++, k++
```

```
    }
```

```
}
```

```
while ( i < M) {
```

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

```
    i++, k++
```

```
}
```

```
while ( j < N) {
```

```
    C[k] = B[j];
```

```
    j++, k++
```

```
}
```

```
return C;
```

```
}
```

A = [1,2,3] B = [4,5,6] so
TC : $O(A+B)$

TC : $O(N+M)$

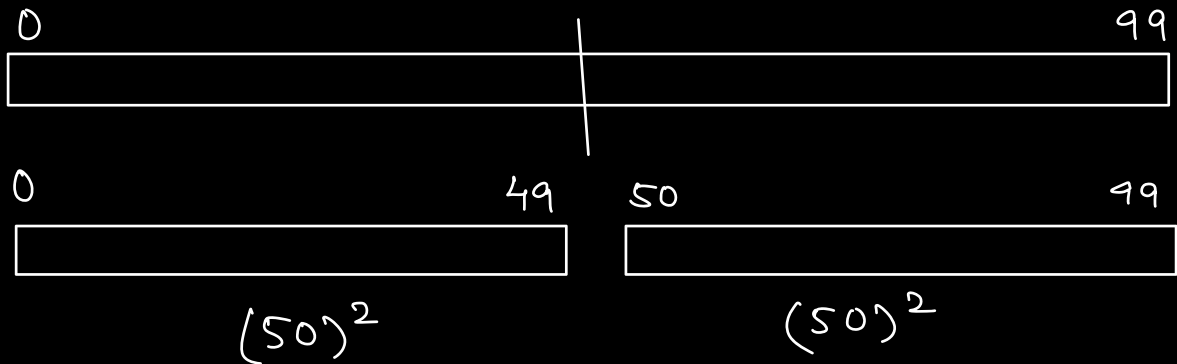
SC : $O(N+M)$

{ Storing the
ans. array }

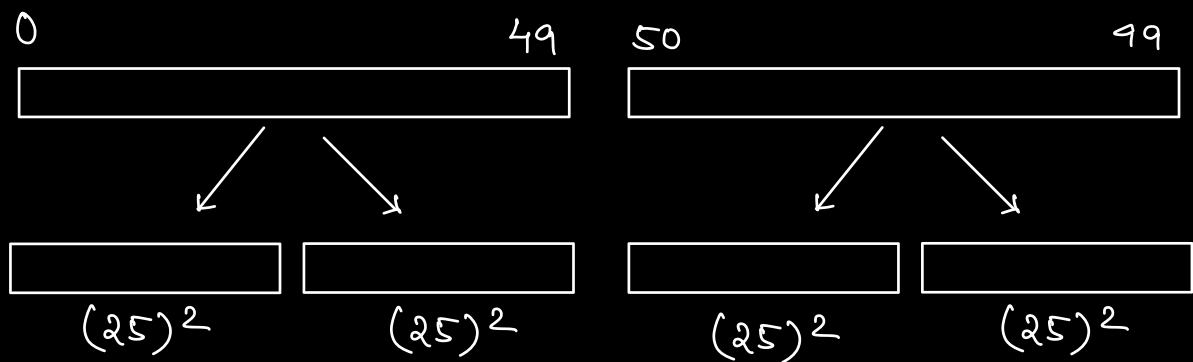
→ Selection } $\Rightarrow \underline{\underline{O(N^2)}}$
 → Bubble

int a[100]

$$\Rightarrow (100)^2 \rightarrow \underline{\underline{10000}}$$

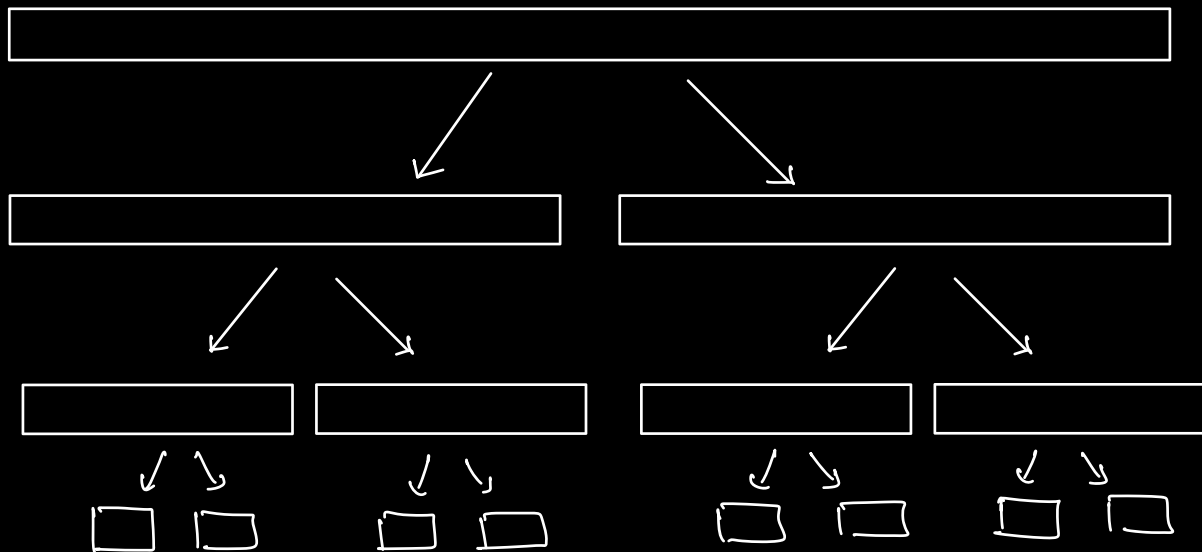


$$\begin{aligned} \text{iterations} &= (50)^2 + (50)^2 + (50+50) \\ &= \underline{\underline{5100}} \end{aligned}$$



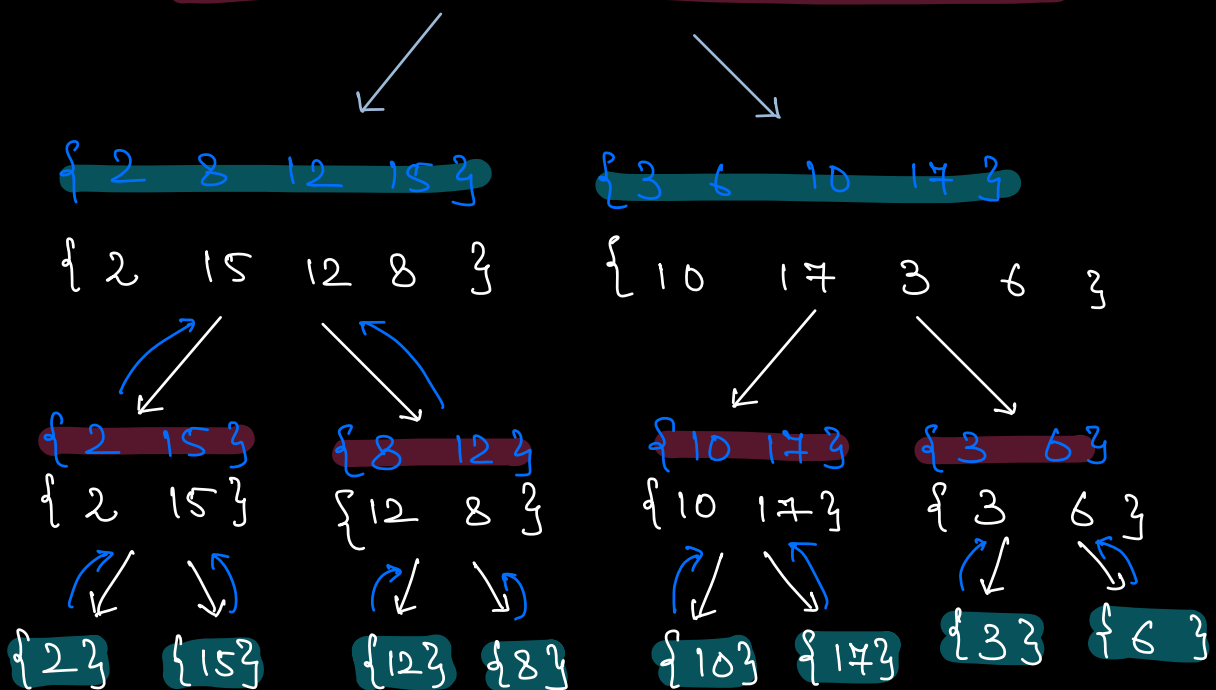
$$\begin{aligned} \text{iterations} &= 4 * (25)^2 + (25+25) + (25+25) \\ &\quad + (50+50) \\ &= \underline{\underline{2700}} \end{aligned}$$

$$10000 \rightarrow 5100 \rightarrow 2700$$



MERGE SORT (Divide & Conquer)

$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \{ 2 & 15 & 12 & 8 & 10 & 14 & 3 & 6 \} \end{matrix}$
 $\{ 2 \ 3 \ 6 \ 8 \ 10 \ 12 \ 15 \ 14 \}$



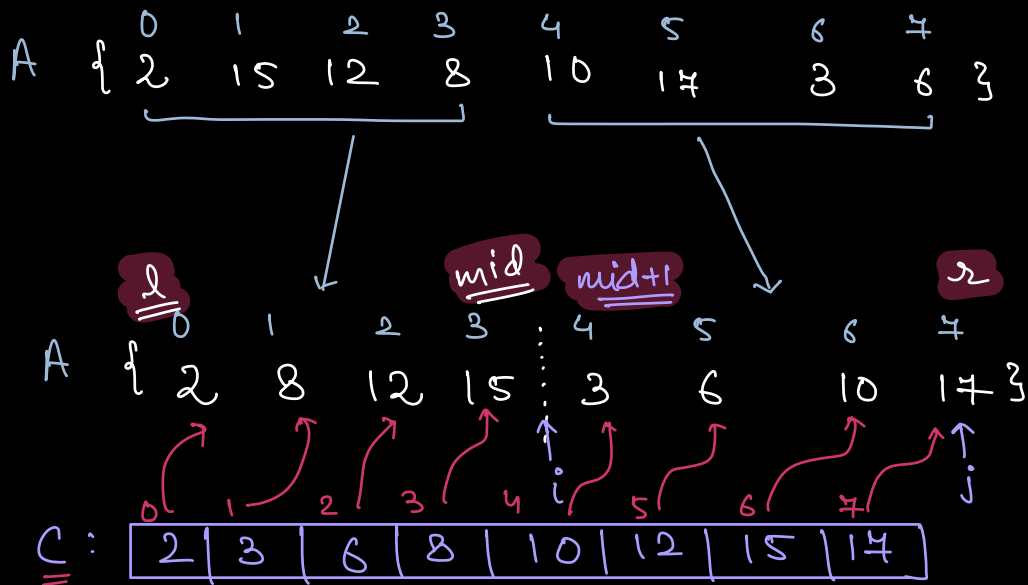
→ Recursion

$\begin{matrix} 0 & N-1 \\ \uparrow & \uparrow \end{matrix}$

```

Void mergeSort (A[], l, r) {
    // Assumption: mergeSort(A, n, y) fun^
    // sorts the Array from n to y
    if (l == r) return;
    mid = (l+r)/2;
    mergeSort(A, l, mid);
    mergeSort(A, mid+1, r);
    merge(A, l, mid, r);
}
  
```

3



$ms(A, 0, 7)$

$\rightarrow m = 3$

$\rightarrow ms(A, 0, 3)$

$\rightarrow ms(A, 4, 7)$

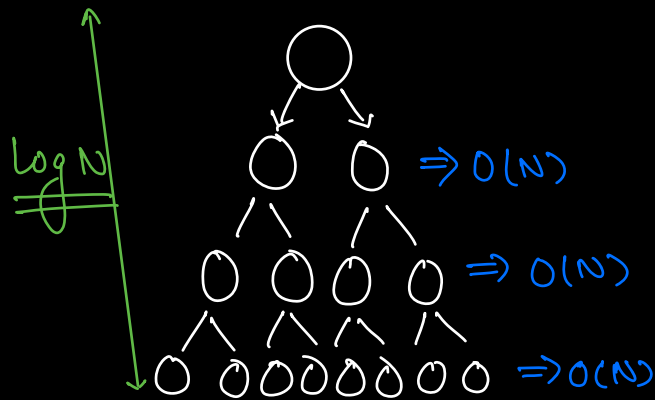
$\rightarrow merge(A, 0, 3, 7)$

$\rightarrow 1^{st} \text{ part: } 0 \text{ to } 3$

$\rightarrow 2^{nd} \text{ part: } 4 \text{ to } 7$

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

$$TC: O(N \log N)$$



SC:

$$\underbrace{O(\log N)}_{\text{Recursive Stack}} + \underbrace{O(N)}_{\text{Merge fun}} \rightarrow \underline{\underline{O(N)}}$$

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

Inplace X

HW Stability

————— * —————

Merge two sorted subarrays from l to y & $y+1$ to r .

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

```
    i = l
```

```
    j = y+1
```

```
    k = 0
```

```
    int C[r-l+1]
```

```
    while ( i <= y && j <= r ) {
```

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

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

```
            i++, k++;
```

```
        }
```

```
        else {
```

```
            C[k] = A[j];
```

```
            j++, k++;
```

```
        }
```

```
    }
```

```
    while ( i <= y ) {
```

```
        C[k] = A[i]
```

```
        i++, k++;
```

```
    }
```

```
    while ( j <= r ) {
```

```
        C[k] = A[j]
```

```
        j++, k++;
```

```
    }
```

// copy the array C into Array A

```

for (x = 1; x <= n; x++) {
    A[x] = C[x-1];
}

```

3

Doubt

A:	1	2	3	i
B:	3	4	5	j

C[6]: { 1 2 3 3 4 5 }