Selection Procedure

Pre-requisite → Quicksort
↳ Partition algorithm

0	1	2	3	4	5
~~20~~ 10	~~10~~ ~~80~~ 20	50 ~~10~~	70	60	30
x	J	J	J	J	J
~~i~~	i				
1	2	3	4		

Lesser than 20

→ Larger than 20

$n = 6$

$k = 3$ ⟹ 3rd smallest element

Result = 30

① Selection sort ⟶ Array

↳ after every iteration, we are getting minimum element in extreme left.

$O(n \cdot k)$

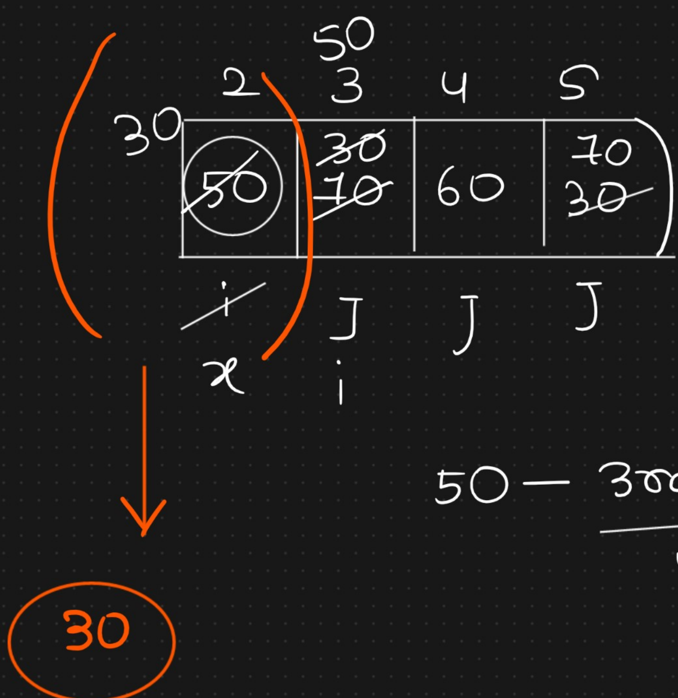
2) Partition Algorithm

Pivot element → at its correct position

index

20 — 1

↳ 2nd smallest element in an array



50 — 3rd index

↳ 4th smallest element

SelectionProcedure(arr, l, h, k){

Logic

Method Name

int m = partition(arr, l, h);

$\hookrightarrow$ Divide

if(m == k-1){

return arr(k);

c ________

}

else if(m < k-1){

Right side

conquer

return selectionProcedure(arr, m+1, h, k);

$h-(m+1)+1$

$T(h-m)$

}

else

Left side

return selectionProcedure(arr, l, m-1, k);

}

$m-1-l+1$

$T(m-l)$

Time & Space complexity

Recurrence Relation

$$T(n) = \begin{cases} 1 & n=1 \\ T(n-m)+n & n>1 \\ \text{OR} & \\ T(m-l)+n & \end{cases}$$

↑ Right side → Partition

↳ Partition

(30, 10, 20) 50 (70, 60, 90) Left side

(10) 50 (70, 60, 90, 80)

Best case	Worst case
$T(n) = T\left(\frac{n}{2}\right) + n$	$T(n) = T(n-1) + n$
Partition	Partition
$a=1$	$= O(n^2)$
$b=1$	

$n^{\log_b a} = n^0 = 1 \mid n$

$O(n)$

Space complexity

Recursive code

↪ Stack

↪ to store the function calls

$O(n)$ ⟸