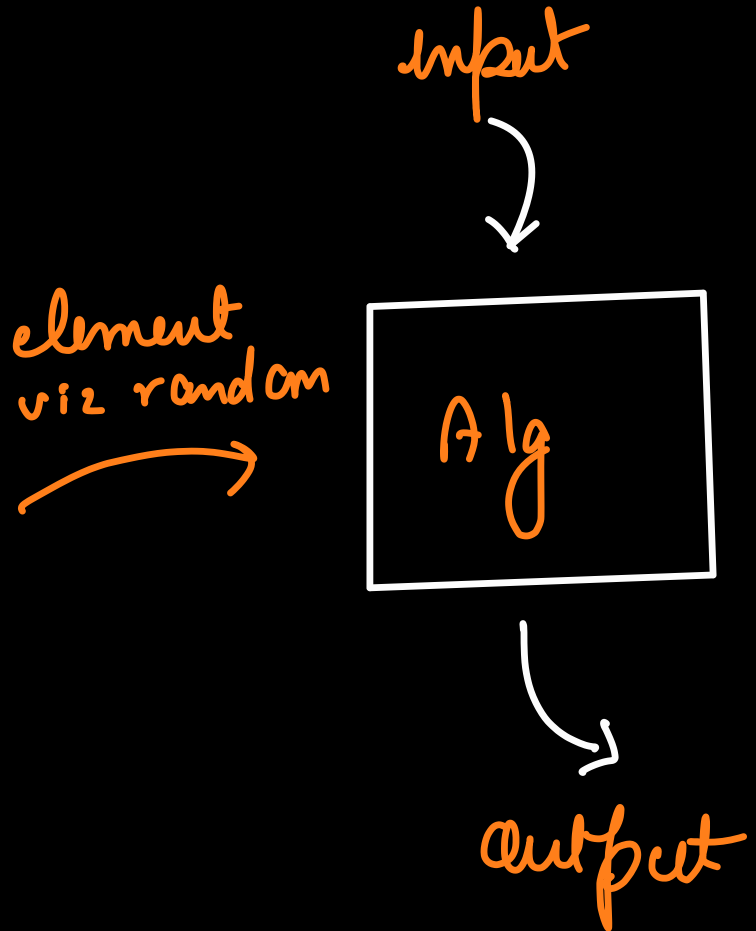
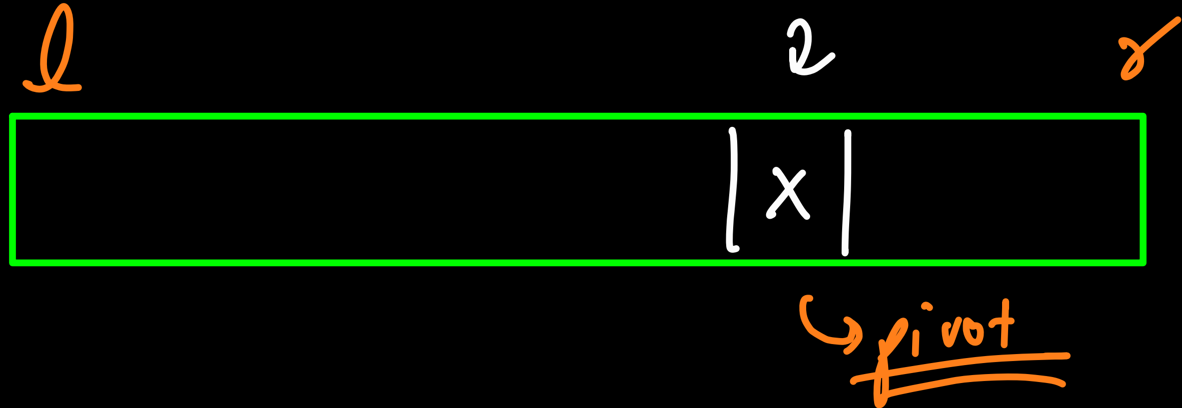
sad

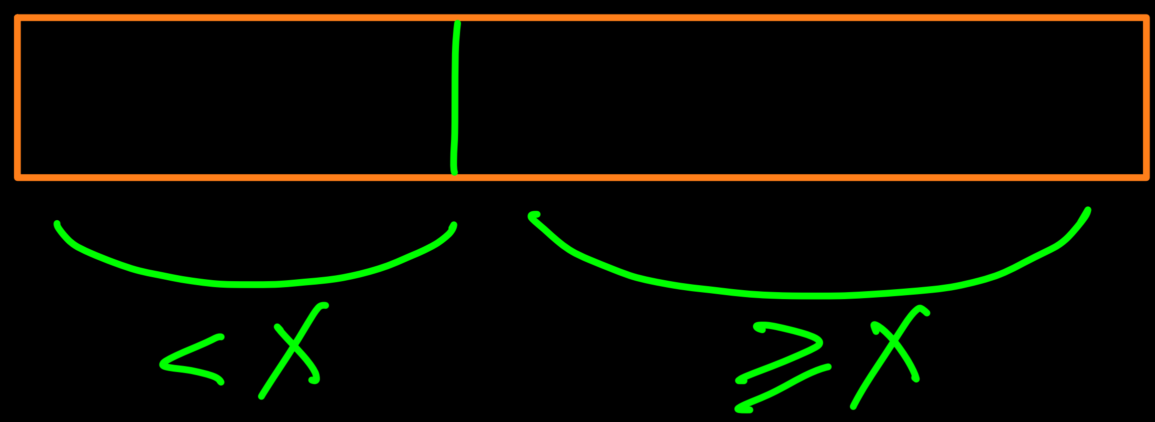
Quick Sort → Randomization



How quick sort works actually ??

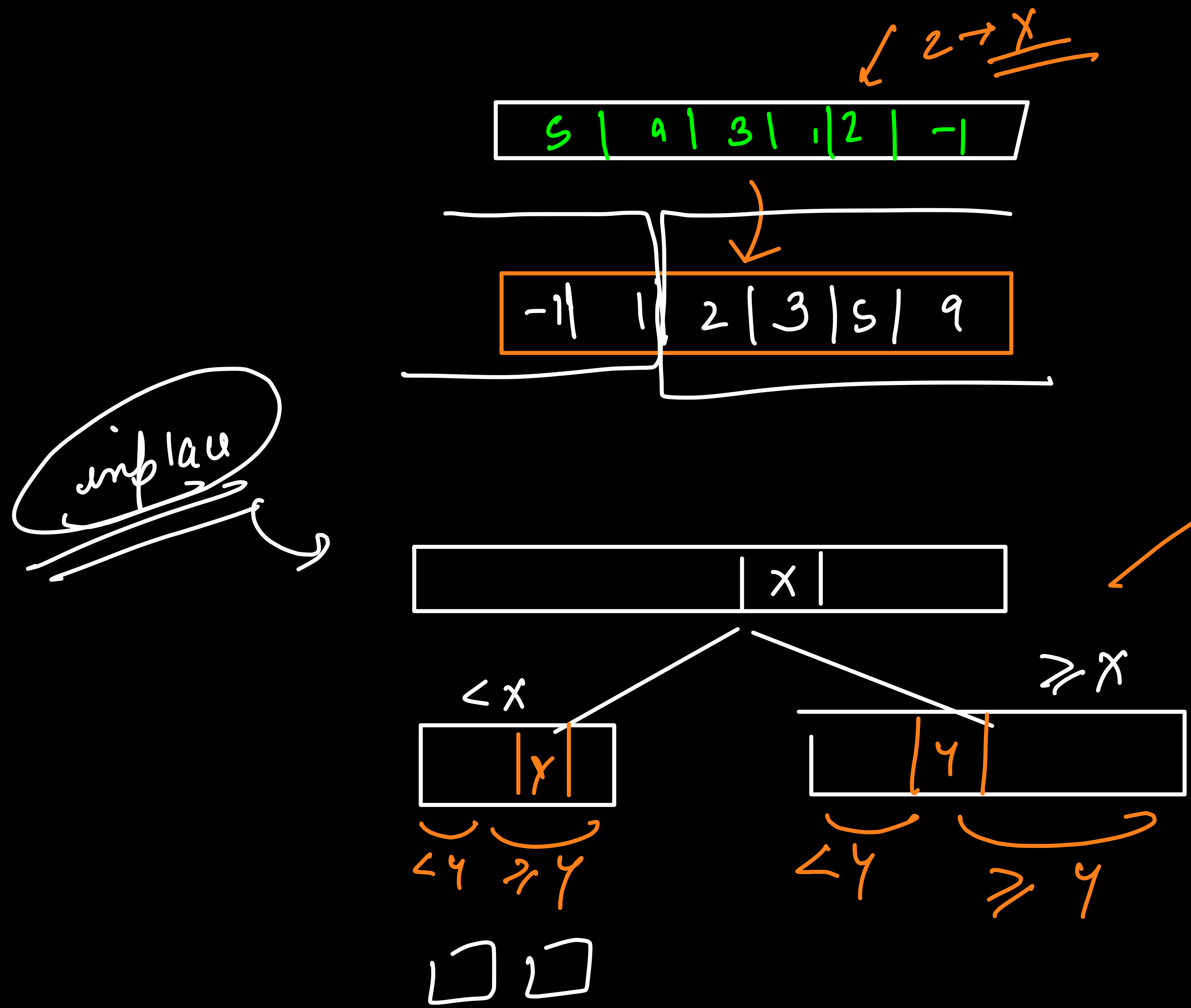


$x = arr[\text{random}(l, r)]$



We segregate elements in the array into 2 parts such that left part has all the elements less than x & right has all element greater or equal to x.

left and right part might not be equal halves.



We won't be making new arrays, & segregate everything in the same array.

Partition algorithm

l			2M		r ← i
23	9	18	32	61	50
0	1	2	3	4	5
l				r-1	

$l, m-1$
$m+1, r$

↑ m → correct index for pivot

pivot → $X = 32$

pivot index → r

randomly

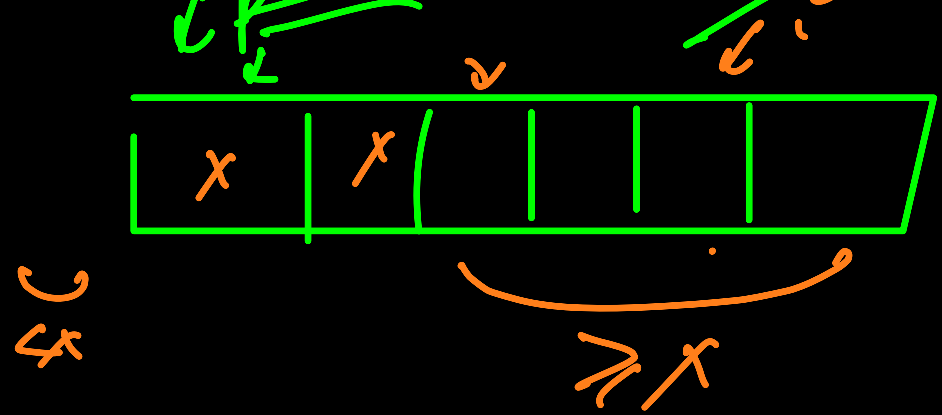
loop → $(l, r-1)$

$[0, 2] \rightarrow < X$

$[3, n-1] \rightarrow \geq X$

→ smallest

< pivot



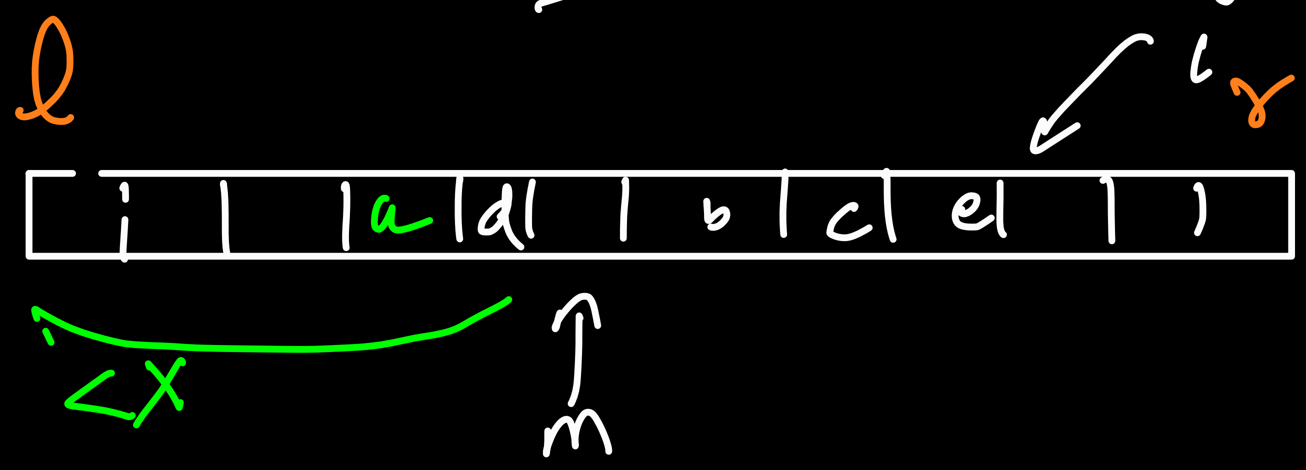
n

Two pointers

$d < x$

$a < x$ → swap(i, m)

$m{+}{+}$



all the elements present on the indices $< m$ are for sure $< x$.

```
partition (arr, l, r, Xind) {
    pivot = arr[Xind]
    swap (arr, Xind, r);
    m = l
    for (i = l ; i <= r-1 ; i++)
        if (arr[i] < pivot) {
            swap (arr, i, m);
            m++;
        }
    }
    swap (arr, m, r);
}
```

$\rightarrow O(n)$

$\rightarrow$ space $\rightarrow O(1)$

$$T(n) = \cancel{T(n-1)} + c_1$$

→ partition

quicksort on

n length

$$\cancel{T(n-1)} = \cancel{T(n-2)} + c(n-1)$$

$$\cancel{T(n-2)} = \cancel{T(n-3)} + c(n-2)$$

$$\cancel{T(n-3)} = \cancel{T(n-4)} + c(n-3)$$

$$\vdots$$

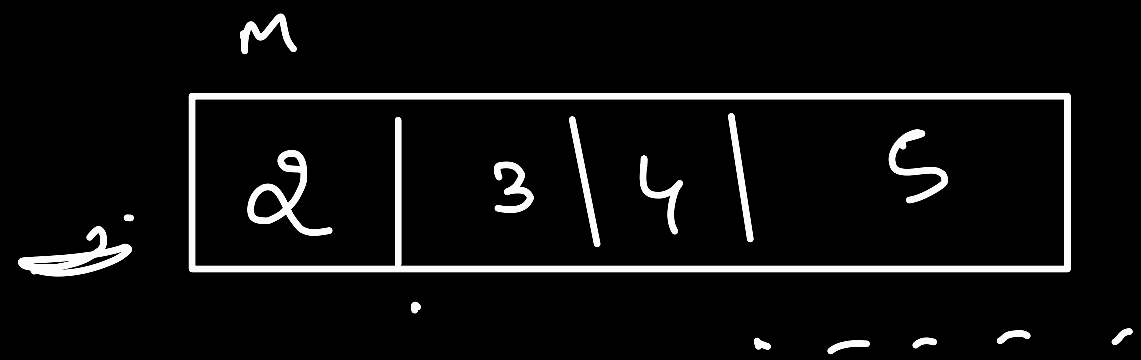
$$\cancel{T(2)} = \cancel{T(1)} + c * 2$$

$$\cancel{T(1)} = T(0) + c$$

$$T(n) = T(0) + c(n-1) + c(n-2) + c(n-3) \cdots\cdots c$$

$$T(n) = c + c\left(\frac{n*(n-1)}{2}\right) \approx O(n^2)$$

$m+1, r$



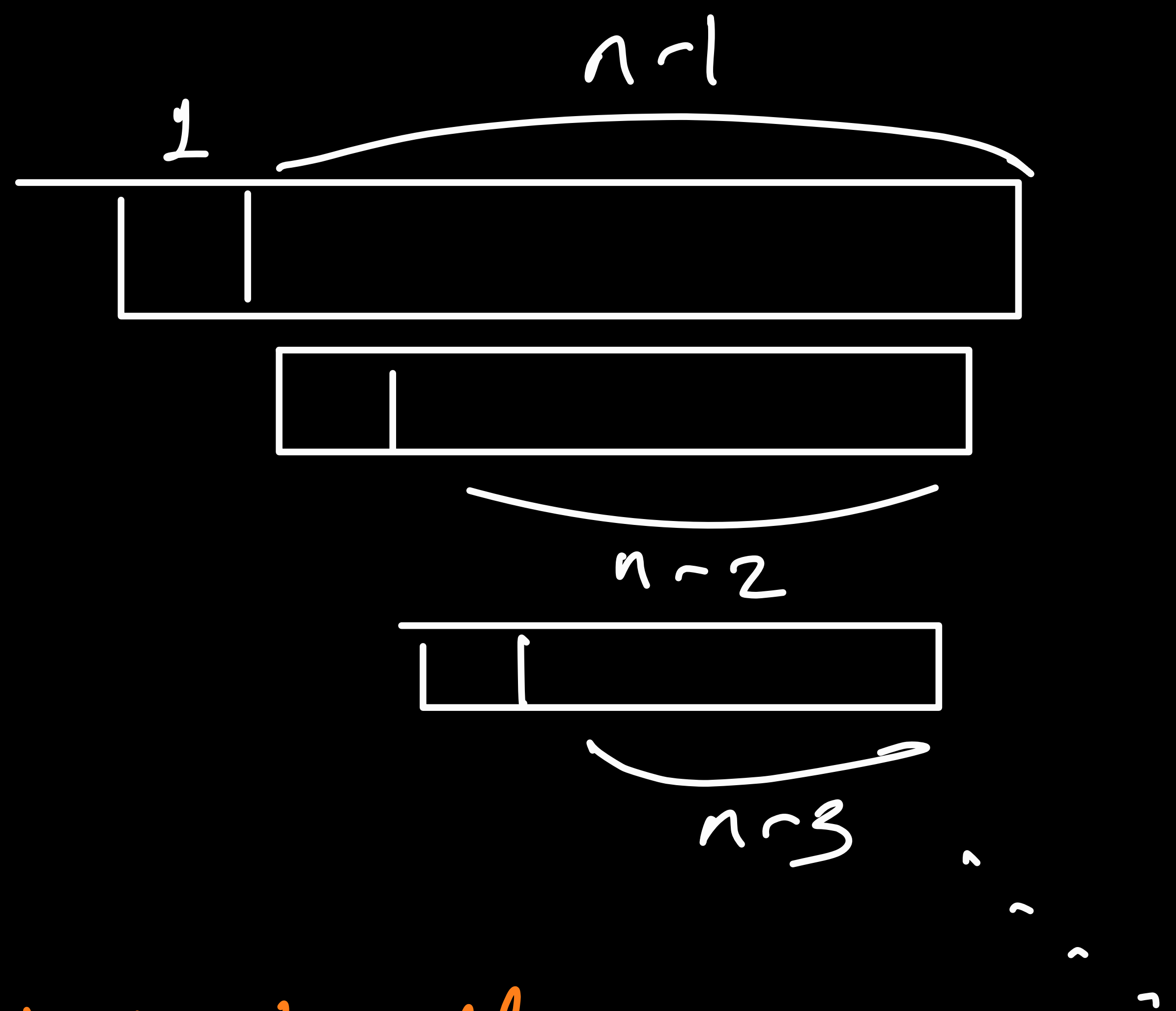
n

$n-1$

$n-2$

extra callu

$O(n^2)$



Worst Case

OR pivot → largest

2 pivot

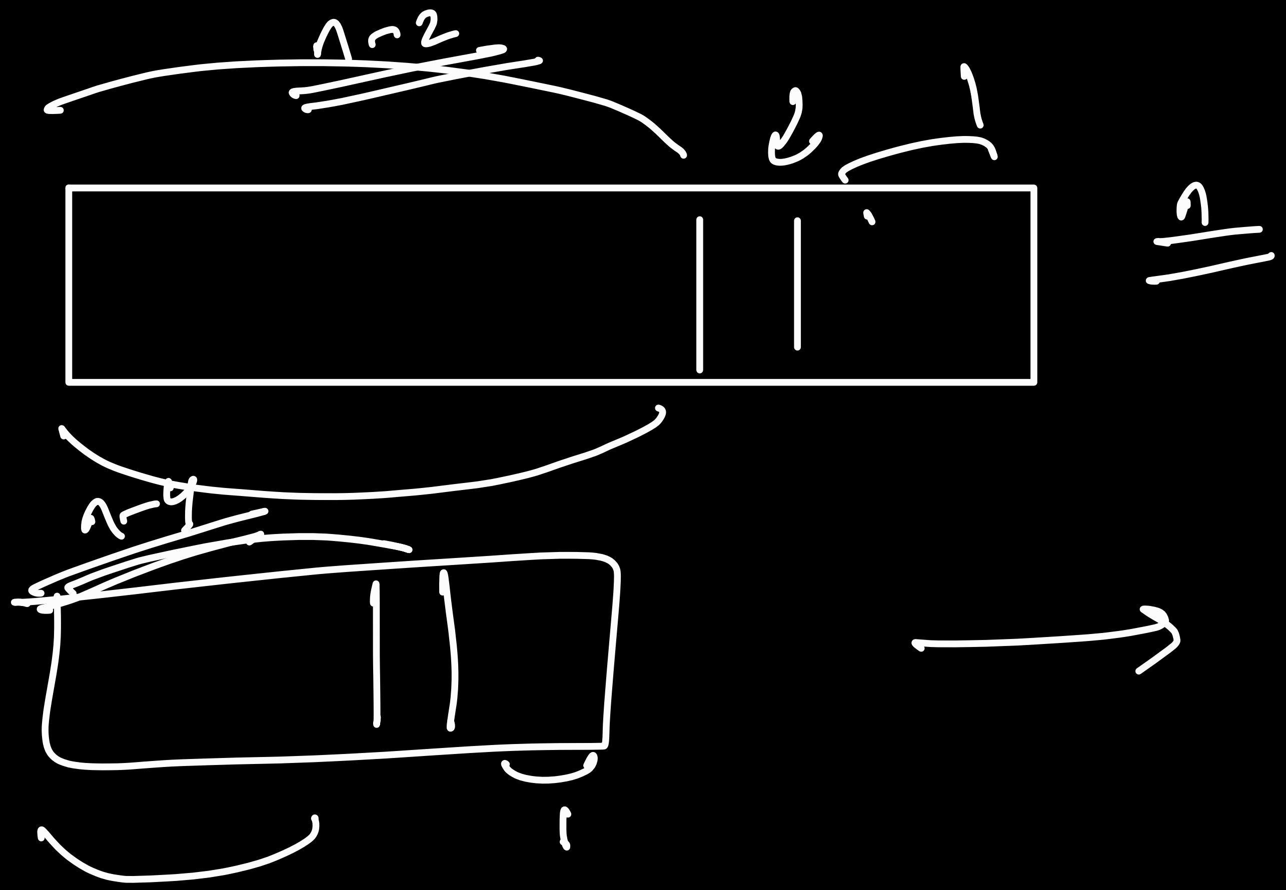
1	2	3	4	5	6	7

→ $n-1$

$n-2$

$n-3$

$O(n^2)$



n

$n-2$

$n-4$

$n-6$

$O(n^2)$

JOIN THE DARKSIDE