SORTING 2

Good Evening

Today's content

01. Insertion sort
02. Nuts & Bolts example
03. Pivot partition
04. Quick sort
05. Quick select sort

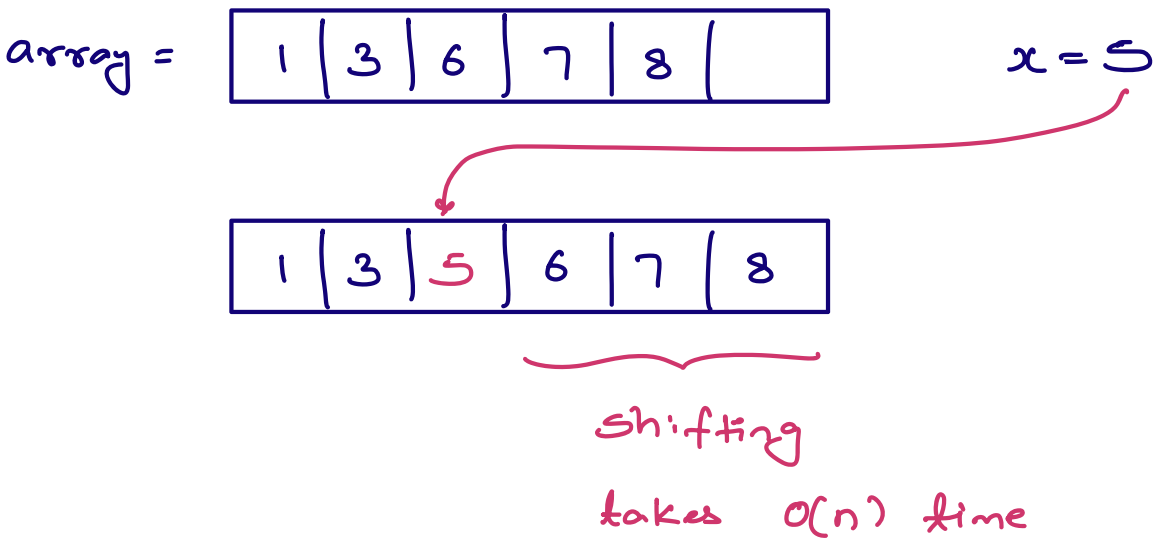
Bubble sort, Selection sort or Mergesort

↳ works when you have the complete data

Q Sort the running stream of all integers

Input = 2 10 8 7 15 1 ....

Sorted = 1 2 7 8 10 15



Minimum no. of shifts = 0 shifts

Maximum no. of shifts = n shifts

Time complexity = $O(n^2)$

Stream = 9 8 7 6

output = 6 7 8 9

Total no. of shifts = $\underbrace{0+1+2+3}$

Sum of (n-1) natural no.

$$= \frac{(n-1)*(n-1+1)}{2} = \frac{(n-1)*n}{2}$$

* int i = 0

n = 0

```
for (all input of x)
{
    for( i= n ; i>0 ; i--){
        if (A[i-1] > x){
            A[i]= A[i-1]:
        }
        else{
            break:
        }
    }
    A[i] =x :
    n= n+1
}
```

TC: $O(n^2)$

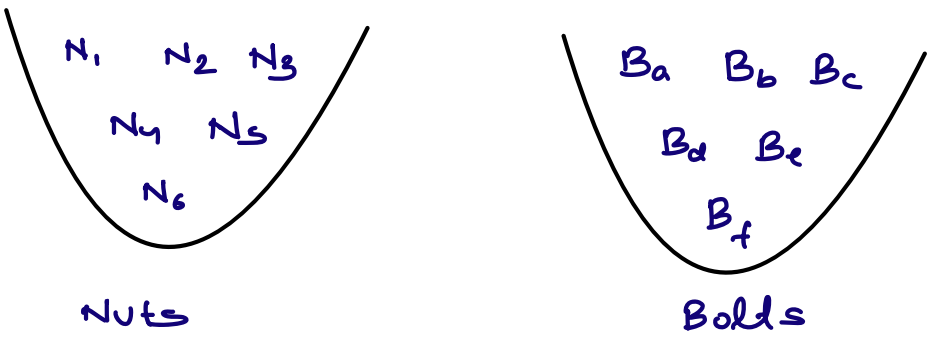
SC: $O(1)$

Nuts & Bolts

Given N nuts of different sizes & B bolts of different sizes.

There is a 1:1 mapping b/w nuts & bolts

Match the nut & bolts with a given constraint that comparing a nut with a nut & a bolt with another bolt is not allowed.



Compare a nut & bolt
- exactly fit
- nut is small
- nut is bigg

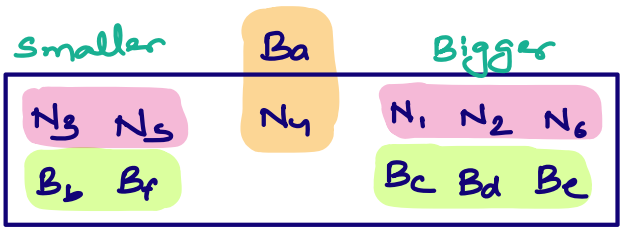
Arrays.sort(nuts) → comparing a nut with a nut ✗

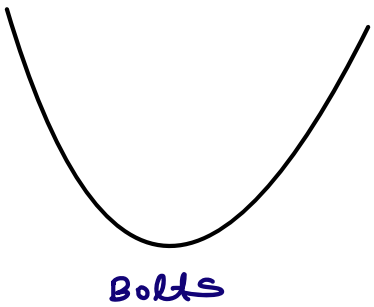
Brute force approach → Compare a nut with all the bolts

$$TC = 6+5+4+3+2+1$$

$$TC = O(n^2)$$

Divide & Conquer technique





Partition → Dividing the nuts & bolts in two parts using a pivot ele

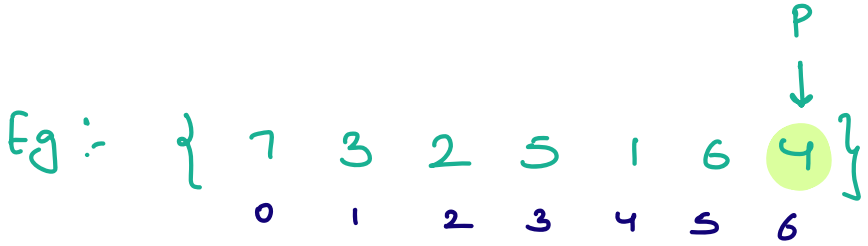
Q ↴

Given an integer array, consider last ele 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

Given all distinct elements



3 2 1 4 7 5 6

Eg: { 7 3 2 5 1 6 }
0 1 2 3 4 5

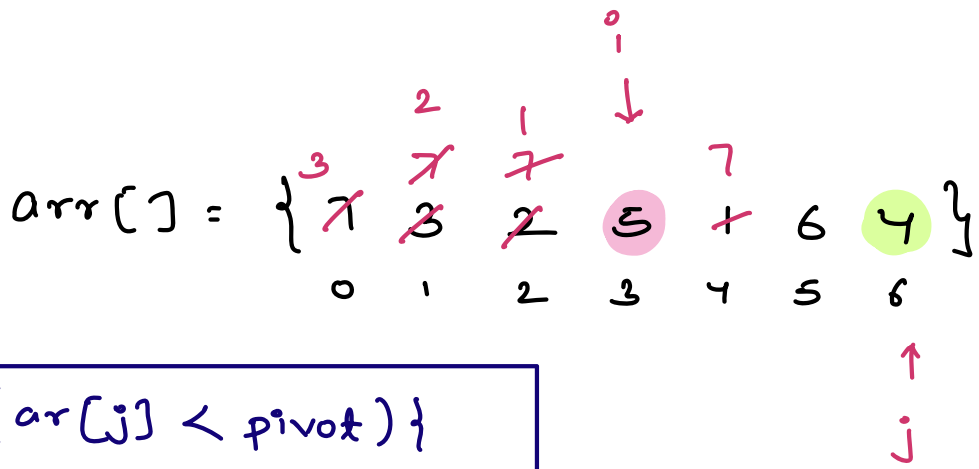
3 2 1 5 6 7

Idea 1 → Sort the entire array = TC: O(nlogn)

Idea 2 → Use two pointer approach

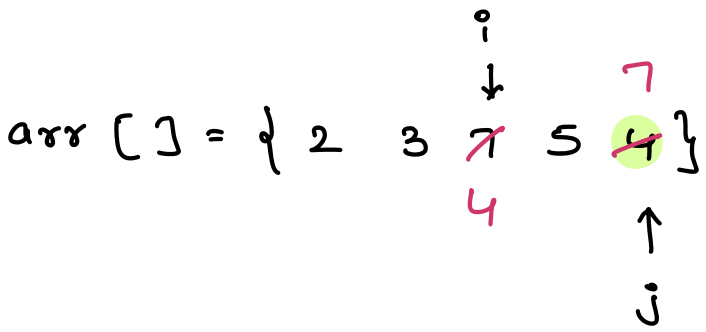
i → pointing at the posn where smaller ele need to be inserted

j → to iterate on array




```
if (ar[j] < pivot){
    swap(A[i], A[j]);
    i++;
}
j++;
```

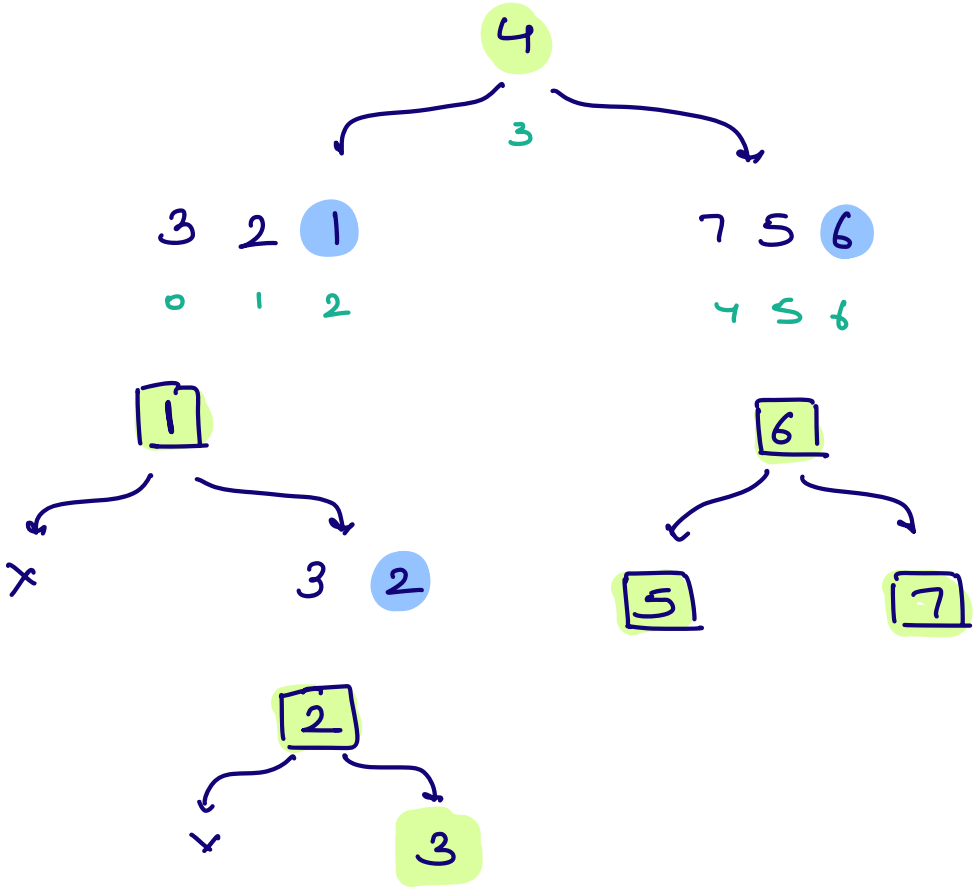
A[n-1] swap A[i] ⟶ to move pivot at correct posn




```
int partition (int [] A, int lo, int hi)
{
    int pivot = A[hi];
    int i = lo;

    for (j = lo; j <= hi-1; j++){
        if (A[j] < pivot){
            swap(A[j], A[i]);
            i++;
        }
    }
    swap(A[i], A[hi]);
    return i;
}
```

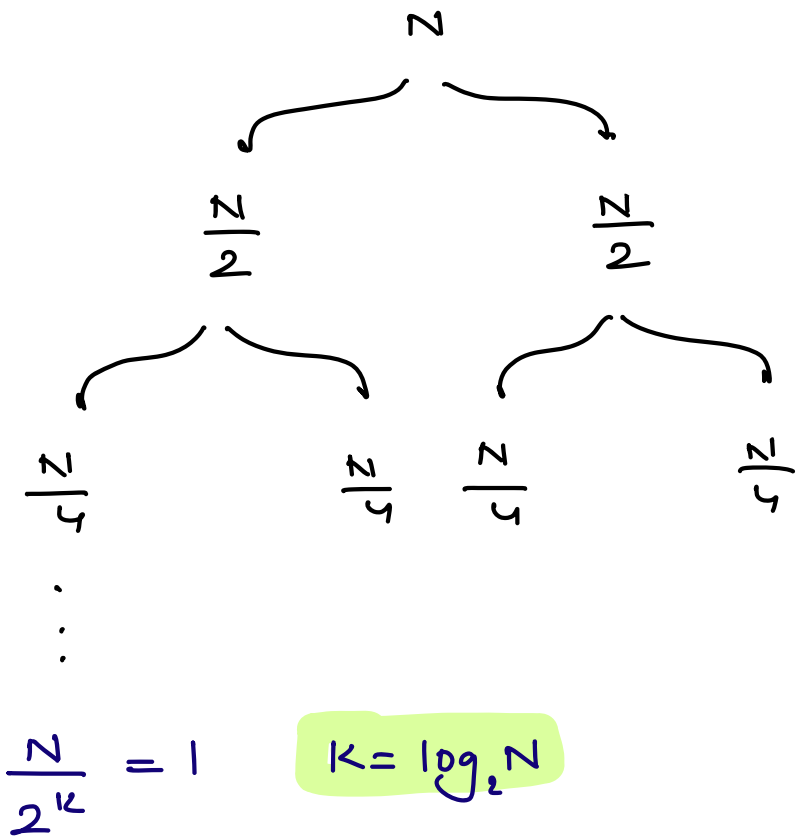
arr[] = { 7, 3, 2, 5, 1, 6, 4 }



arr[] = { 1, 2, 3, 4, 5, 6, 7 }

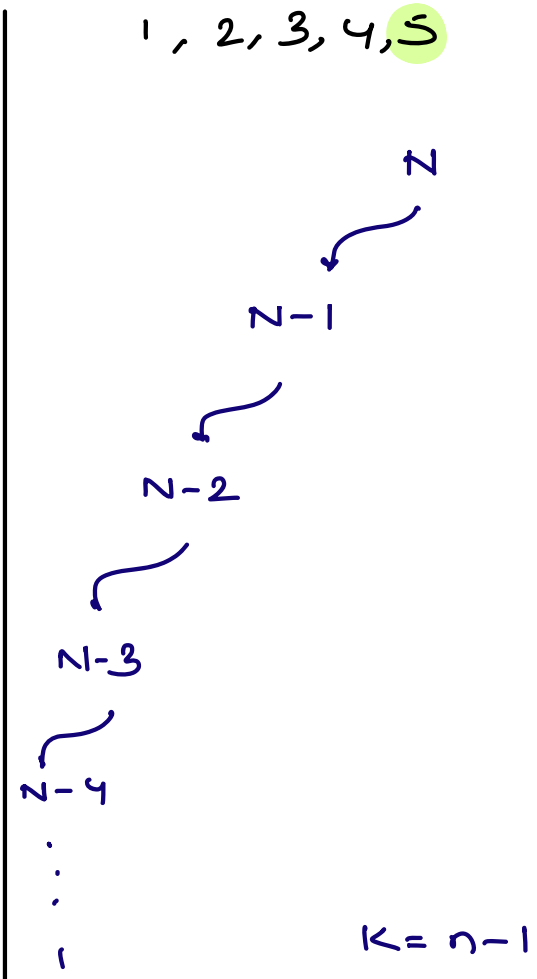
* void quicksort (Arr; s , e)

```
if (s ≥ e) { return } :
int pi = partition (arr, s, e) :
quicksort ( arr, s , pi-1)
quicksort (arr, pi+1, e )
}
```



$\frac{N}{2^K} = 1$ $K = \log_2 N$

TC: $O(n \log N)$



TC: $O(n^2)$

$O(n \log n) \leq$ TC for quick sort $\leq O(n^2)$

Best

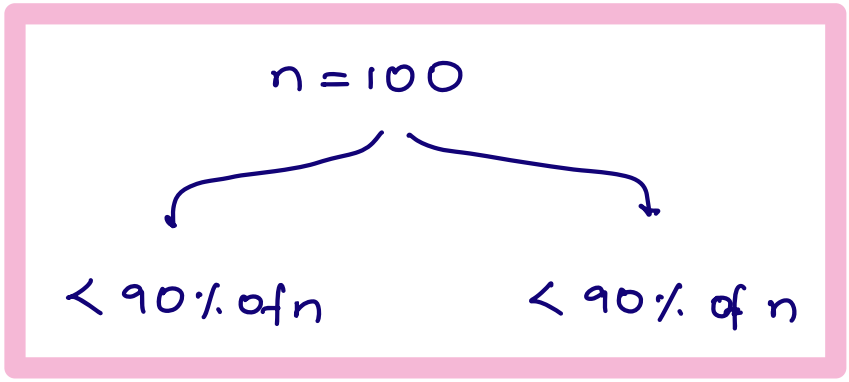
Worst

Probabilistic Analysis of TC

{ 1, 2, 3, 4, 5 . .... 100 } ← Random order

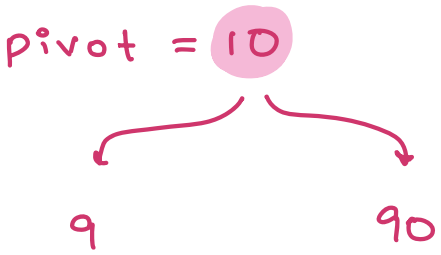
Worst possible pivot {1, 100}

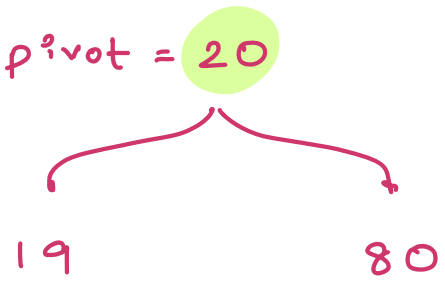
Best possible pivot {50, 51}

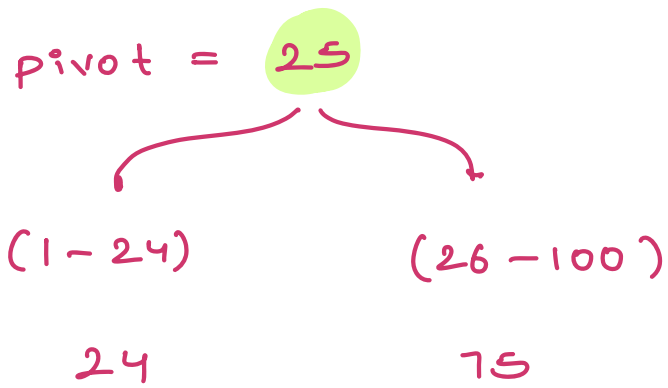


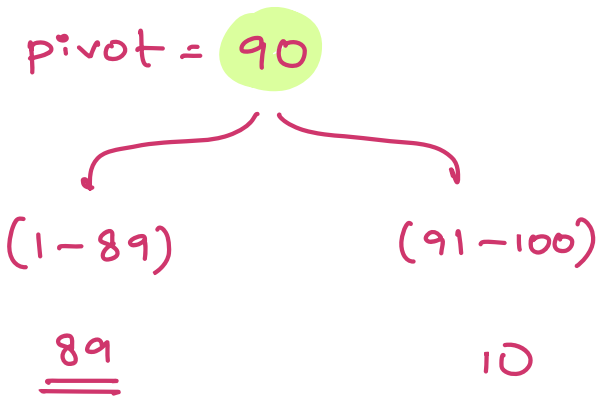
Range of pivot: [11-90]

Total ele for pivot = 80



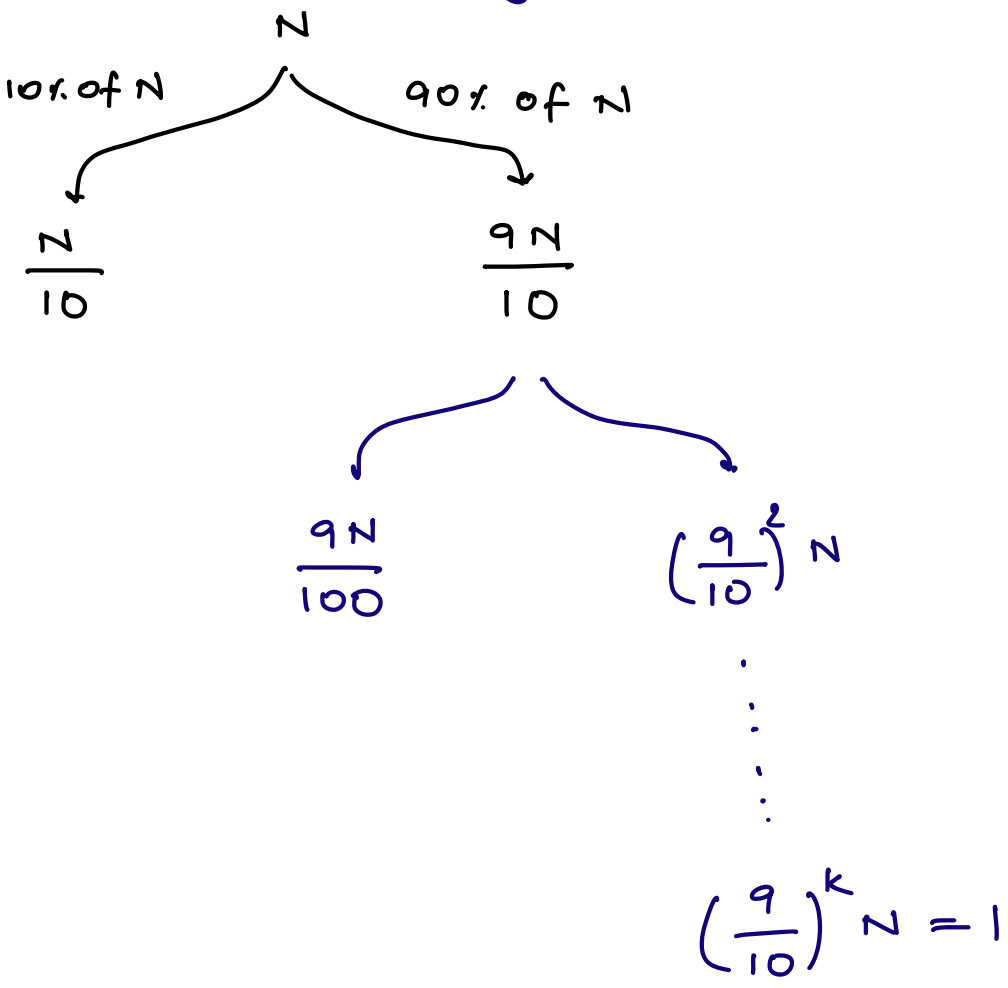






Probability of picking an ele out of 100 such that the subproblem < 90% $= \frac{80}{100} =$ 80%.

Worst case scenario → my pivot lies in 20%



$$K = \log_{(\frac{10}{9})} n$$

$n = 10^5$

$K = \log_{(\frac{10}{9})}(10^5) \approx 10^2$

$TC = O(N \log_{(\frac{10}{9})} N)$ $\quad N = 10^5$

$\Rightarrow 10^5 * 10^2$

$= 10^7$ iterations/sec

$TC = O(N \log_{(\frac{10}{9})} N)$ ⟶ 80% of the time we are going to take less iteration than this

	Merge Sort		Quick sort
Time	$O(n \log n)$	>	In worst case → $O(n^2)$
Space	$O(n)$	<	$O(1)$

* You are given an array. Find the k^{th} smallest ele in given array

arr [ ] = { 3, 7, 2, 4, 5 } K = 3

0 1 2 3 4

sorted = { 2, 3, 4, 5, 7 }

0 1 (2) 3 4

arr = { 3, 7, 2, 4, 5 }
0 1 2 3 4

arr = { 3, 2, 4, 5, 7 }
0 1 2 3 4

arr = { 3, 2, 4 }
0 1 2

```
int quickselect ( arr, lo, hi, k )        → k = k-1
|   pivot = ar[hi];
|   pidx = partition (arr, lo, hi);
|   if (pidx == k) return ar[pidx]
|   else if (pidx > k) return quickselect(arr, lo, pi-1, k)
|   else return quickselect ( arr, pi+1, hi, k)
}
```

TC = Google