Q1 Given N array elem, rearrange such that

→ all elems ≤ ar[0] are to the left of ar[0]

→ all elems > ar[0] are to the right of ar[0]

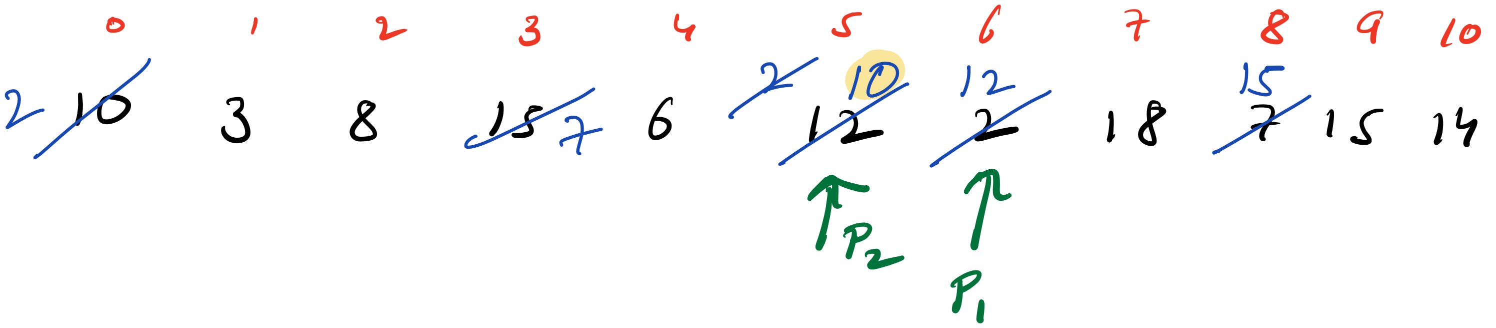
	0	1	2	3	4	5	6	7	8	9	10
Eg-	10	3	8	15	6	12	2	18	7	15	14

⇒ ≤ 10 10 > 10

0	1	2	3	4	5	6	7	8	9	10
10	3	8	15	6	12	2	18	7	15	14

temp	3	8	6	2	7	10	14	15	18	12	15

SC: O(n)
TC: O(n) } O(1)



```
void rearrange (int ar[], int N) {
    P1 = 1              P2 = n-1
    while (P1 ≤ P2) {
        if (ar[p1] ≤ ar[0])
                p1 ++
        else if ( ar[p2] > ar[0])
                p2 --
        else {
            swap( a[p1] , a[p2] )
            p1++        p2 --
        }
    swap (a[0], a[p2]
}
```

TC: O(n)

SC: O(1)

Q2 Given N array elem, rearrange subarray [s:e] st ar[s] is correct position of subarray. Return correct pos

what to change in above code ?

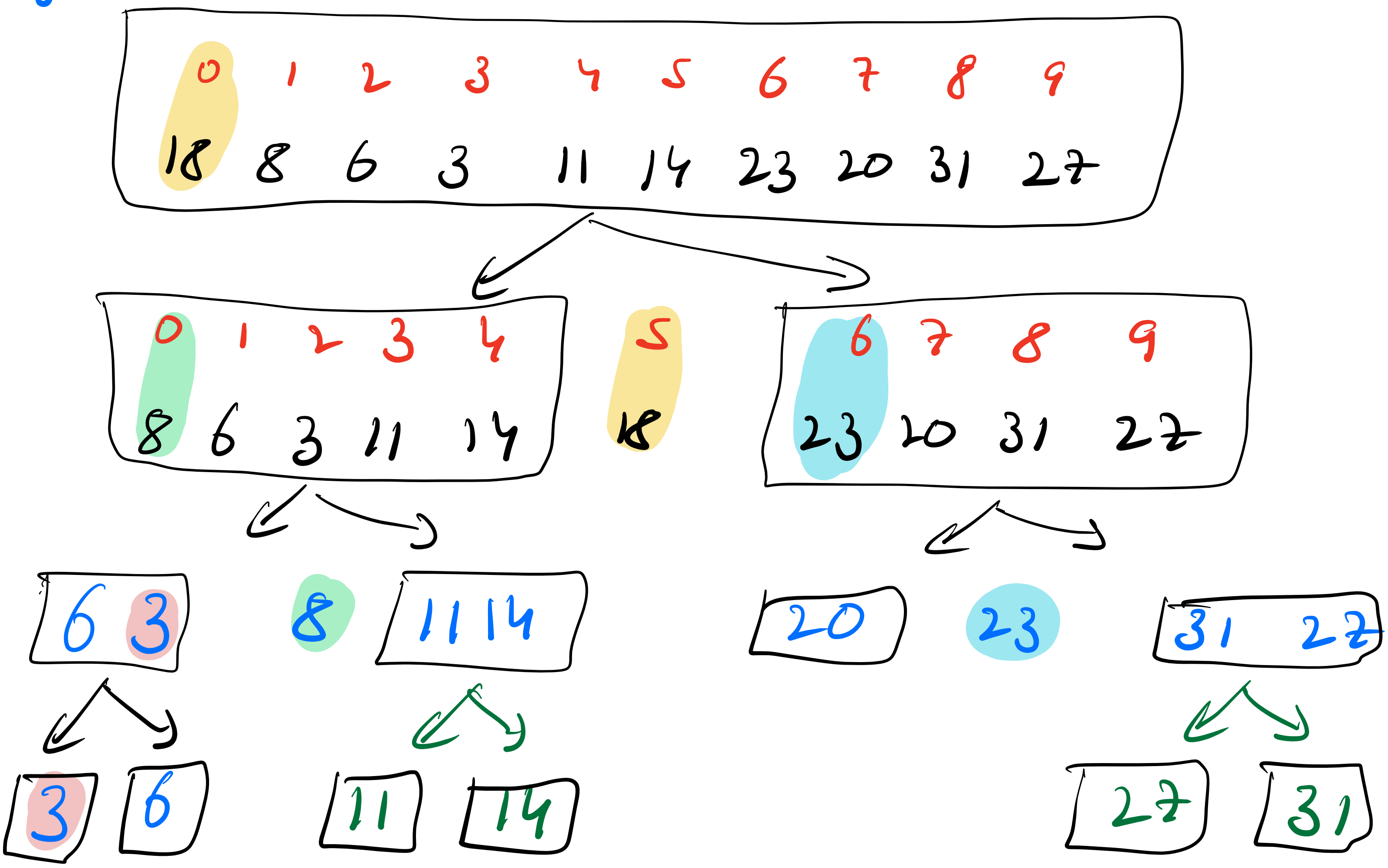
```
int rearrange (int ar[], int N) {
    P1 = S+1        P2 = e                 s ---- e
    while (P1 ≤ P2) {
        if (ar[p1] ≤ ar[s])
                p1++
        else if ( ar[p2] > ar[s])
                p2--
        else {
            swap( a[p1], a[p2])
            p1++        p2--
    }
    swap ( a[s], a[p2])
    return p2
}
```

TC: O(n)

SC: O(1)

How to sort subarray [s:e]

```
void Qsort(int ar[], int s, int e){
    if(s>=e) return
    p = rearrange(ar, s, e)   }  O(n) TC
    //now recurse                      s....p.....e
    Qsort(arr, s , p-1)
    Qsort(arr, p+1, e )
}
```



Time Complexity.

Best Case

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

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

We know this is $O(n\log n)$

Worst Case

$T(N) = N + T(N-1) + T(1)$

$T(N) = N + T(N-1)$

$T(N) = 2N + T(N-2)$

$\vdots$

$T(N) = kN + T(N-k)$

$k = ?$

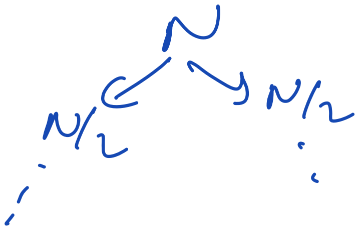
$T(N) = O(N^2)$ worst case

Eg of worst case Any sorted assay.

2 3 4 5 6 7 8

$$\underbrace{N + N-1 + N-2 + \dots + 1} = \frac{n(n+1)}{2}$$

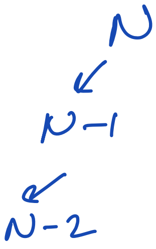
$$\Rightarrow O(n^2)$$



SC best : log n

worst : n

{done}

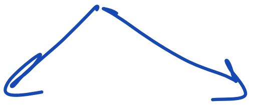


- Main Concept:

Instead of picking the start of subarray as reference, pick random index

0	1	2	3	4	5	6
~~9~~ 10	6	8	2	~~10~~ 9	11	14

6 8 2 9 10 14 11



14 11

This random picking makes average TC: O(n logn)

```
int rearrange (int ar[], int N) {
    int r = rand(s, e)
    swap( ar[s], ar[r]) → So that ref
                          is now at s.
    P1 = S+1        P2 = e
    while (P1 ≤ P2) {
      if (ar[p1] ≤ ar[s])
              p1 ++
      else if ( ar[p2] > ar[s])
                p2 --
    else {
          swap( a[p1], a[p2])
          p1++         p2 --
   }
   swap ( a[s], a[p2])            S
   return p2
}
```

Comparator

Q Given N array elem, sort in increasing order of no of factors. If 2 elem has same no of factors, element with less value should come first

Note: Cannot use extra space

Eg	9	3	4	8	16	37	6	13	15
factors	3	2	3	4	5	2	4	2	4

3 13 37 4 9 6 8 15 16

```
sort(arr, comp)
```

comparator function.

Comparator allows to sort using our rules.

```
bool comp(int a, int b) {
    // if you want a to appear before b
    in the sorted order, return true
        else return false.

    int f1 = factors(a)
    int f2 = factors(b)

    if (f1 < f2)
        return true
    else if (f1 > f2)
        return false
    else    f1==f2    {
        if (a<b)
            return true
        else
            return false
    }
}
sort (arr, comp)
```

Q4 K closest points to origin (0,0)

Given list of points ⇒ return K closest points to origin

Eg - 1,3 -2,2 K = 1

$\sqrt{10}$ $\sqrt{8}$ ans=[{-2,2}

$\sqrt{x^2+y^2}$

Eg 1,3 1,-1 2,-1 K=2

10 2 5

ans=[{1,-1} , {2,-1}]

Distance from origin of point x,y

sqrt (x^2+y^2)

We need to sort on basis of distance.

Ques If I compare x^2+y^2 instead of sqrt (x^2+y^2) ⇒ this works

Idea : Sort using comparator

Code

```
bool cmp( pair<int, int> a, pair<int, int> b){

    long d1 = (long) a.first * a.first +
              (long) a.second * a.second

    long d2 = (long) b.first * b.first +
              (long) b.second * b.second

    if ( d1 < d2 )
            return true
    else
            return false
}


list<pair<int,int>> closest (list<pair<int,int>> A, int K ){

    list<pair<int,int>>  ans .
    sort ( A , cmp )
    for( i=0 ; i<K ; i++ ){
            ans. insert (A[i])
    }
    return  ans
}
```

TC: O(nlogn)
SC: O(n)

0	1	2	3	4	5	6
10	20	30	40	50	60	70

1 to 10^6 all spf spf(i)

1 → 10^6

10^6+1 ⇒ cnt [spf(i)]++