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

Brute idea: Use temp array.

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

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

↑↑ $p_1 p_2$

TC: O(N)
SC: O(N) } But we want O(1) SC

0	1	2	3	4	5	6	7	8	9	10
2	3	8	7	6	10	12	18	15	15	14
					↑ P_2	↑ P_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
    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 (ar[s], ar[p2])
    return p2
}
```

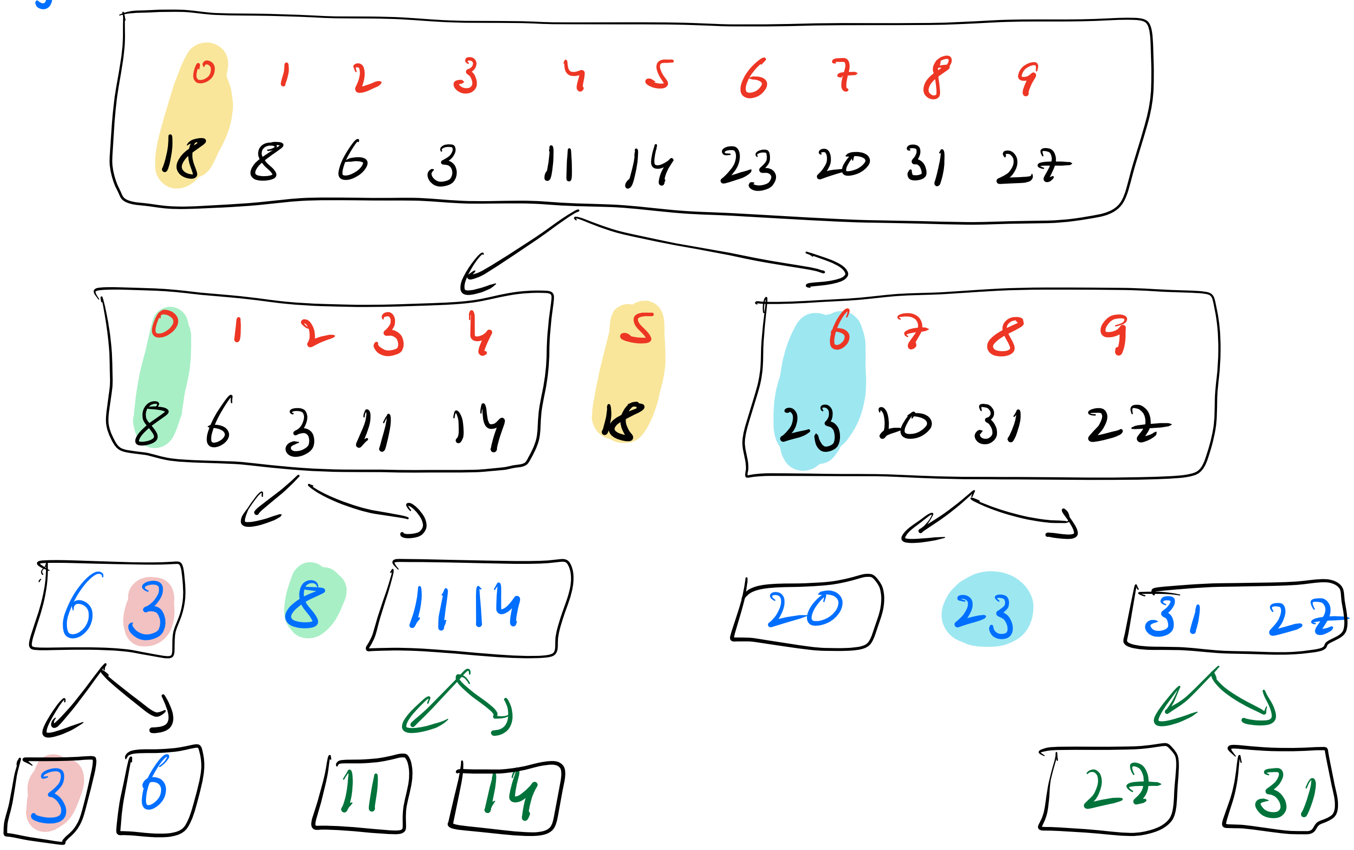
s - - - - p . . . . . e

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)
    //now recurse
    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

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

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

$$T(N-1) = N-1 + T(N-2)$$

$$T(N) = N + (N-1) + T(N-2)$$

$$= N + (N-1) + (N-2) + T(N-3)$$

$$= N + N-1 + N-2 + - - - - - 1$$

$$= \frac{n(n+1)}{2} \Longrightarrow O(n^2)$$

Eg of worst case Any sorted array in desc order

2 3 4 5 6 7 8

start of sub ⟹ reference

pivot

0 1 2 3 4

2 ~~7~~ 5 ~~2~~ ~~7~~ 4 3

0 1 2 3 4

3 7 11 12 14

- Main Concept:

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

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

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

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

Q Unique elements

Make all elem unique. How?
you can do +1 to any elem
any no of times. Min moves

Eg 1   1, 1, 3   ans = 1

Eg 2   4, 5, 2   ans = 0

Eg 3   1, 1, 1
↑
ans = 3

Obs ⇒ Lets first sort the array

Now, start from $i = 1$

If $a[i] \leq a[i-1]$ $a_i \Rightarrow a_{i-1} + 1$

$ans += a[i-1] - a[i] + 1$

If a[i] > a[i-1]

ignore continue

Code

```
ans = 0
sort (arr)
for (i=1 ; i<n ; i++) {
    if ( arr[i] <= arr[i-1]) {
        ans + = arr[i-1]+1 - arr[i]
        arr[i] = arr[i-1] + 1
    }
}
return ans.
```

TC : O(nlogn)

SC : O(1)

1, 1, 1, 1

1 2 1 1

1 2 3 4

1+2+3

=6

- Insertion sort

Sort elem by elem by placing at correct position

9 2 7 5 6 4 1

9

2 9

2 7 9

2 5 7 9

2 5 6 7 9

2 4 5 6 7 9

1 2 4 5 6 7 9

0	1	2	3	4	5	6	7
1	2	4	5	5	6	7	9

```
for(i=1;i<n;i++){
    // 0 to i-1 is sorted
    for(j=i-1;j>=0;j--){
        if( ar[j] > ar[j+1])
            swap(ar[j],ar[j+1])
        else
            break
    }
}
```

TC: $O(n^2)$

SC: $O(1)$

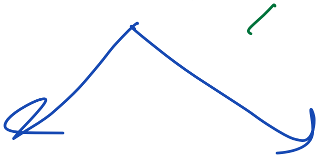
{done}

a ⇒ array

$i < j$

$a[i] > a[j]$

```
list <list> ans
  list<          > ,      i
```



```
list.add(i)                          recurse(list, i+1)
 recurse(list, i+1)
```

```
base case => i == n+1
             if (list.size == B)
                  ans.append(list)
             else
                  invalid
```

[redacted]