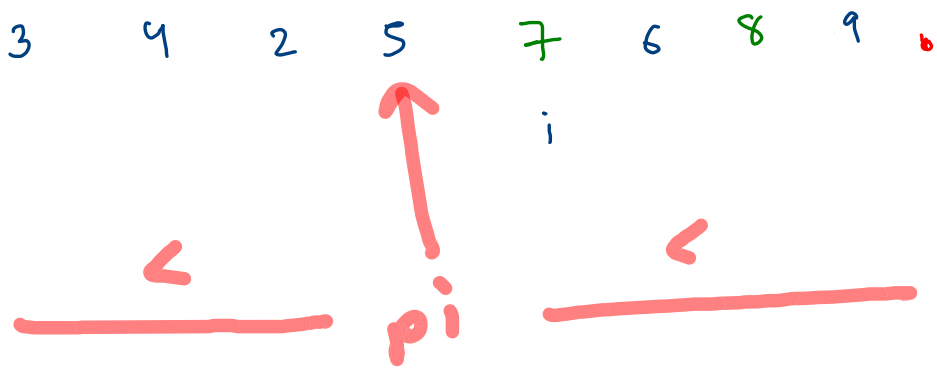


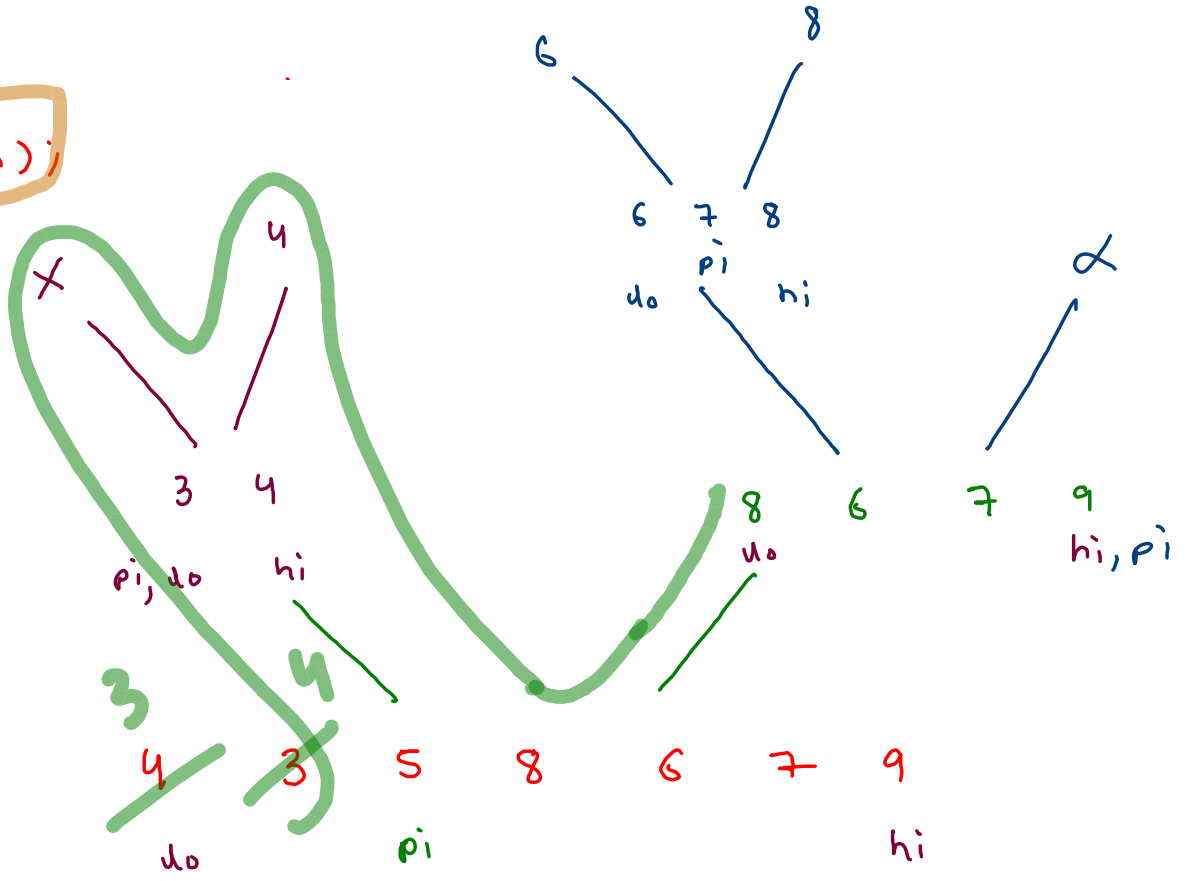

```
if (arr[j] > pi) {
    j++;   -> 1's
}
else {
    swap(i,j);   -> 0's
    i++; j++;
}
```

0 to i-1 -> ≤pi

i to j-1 -> >pi

j to n -> ul

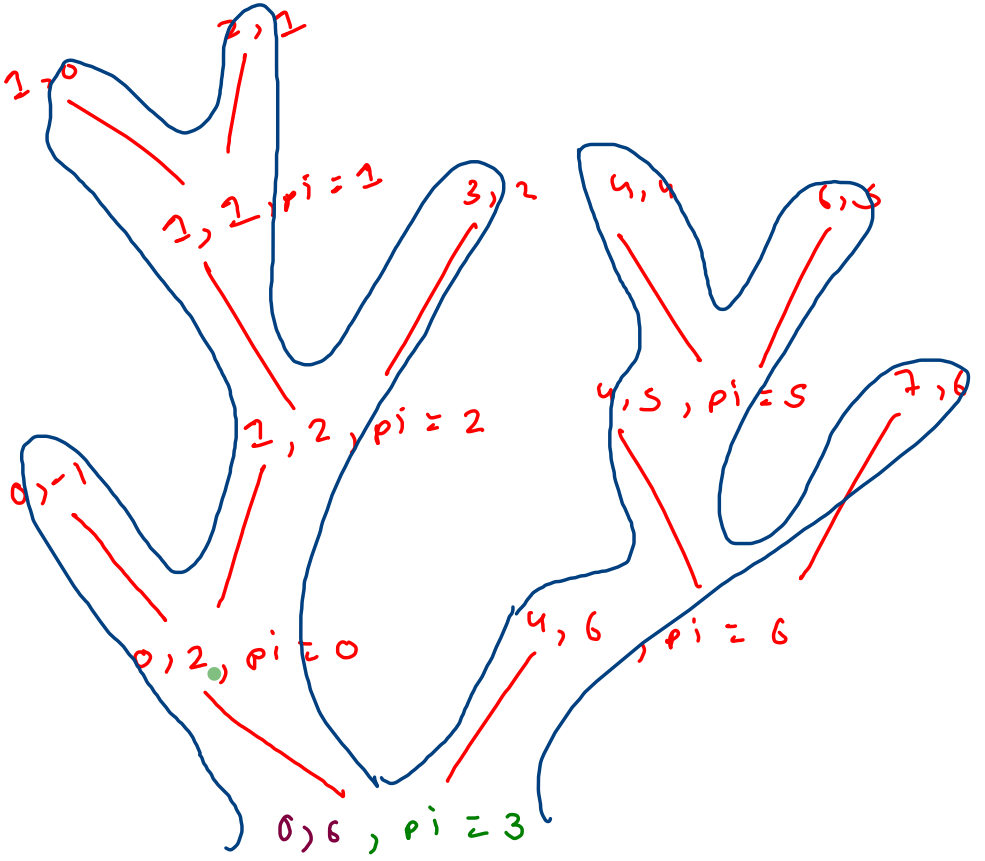
```
void qs (arr, lo, hi) {

  int pi = partition ( arr, arr[hi], lo, hi );
  qs(arr, lo, pi-1);
  qs(arr, pi+1, hi);

}
```



arr = 2 3 4 5 6 8 9
0 1 2 3 4 5 6

```java
public static void quickSort(int[] arr, int lo, int hi) {
    if(lo > hi) {
        return;
    }

    int pi = partition(arr,arr[hi],lo,hi);
    quickSort(arr,lo,pi-1);
    quickSort(arr,pi+1,hi);

}
```

```java
int i = lo, j = lo;
while (i <= hi) {
  if (arr[i] <= pivot) {
    swap(arr, i, j);
    i++;
    j++;
  } else {
    i++;
  }
}
System.out.println("pivot index -> " + (j - 1));
return (j - 1);
```

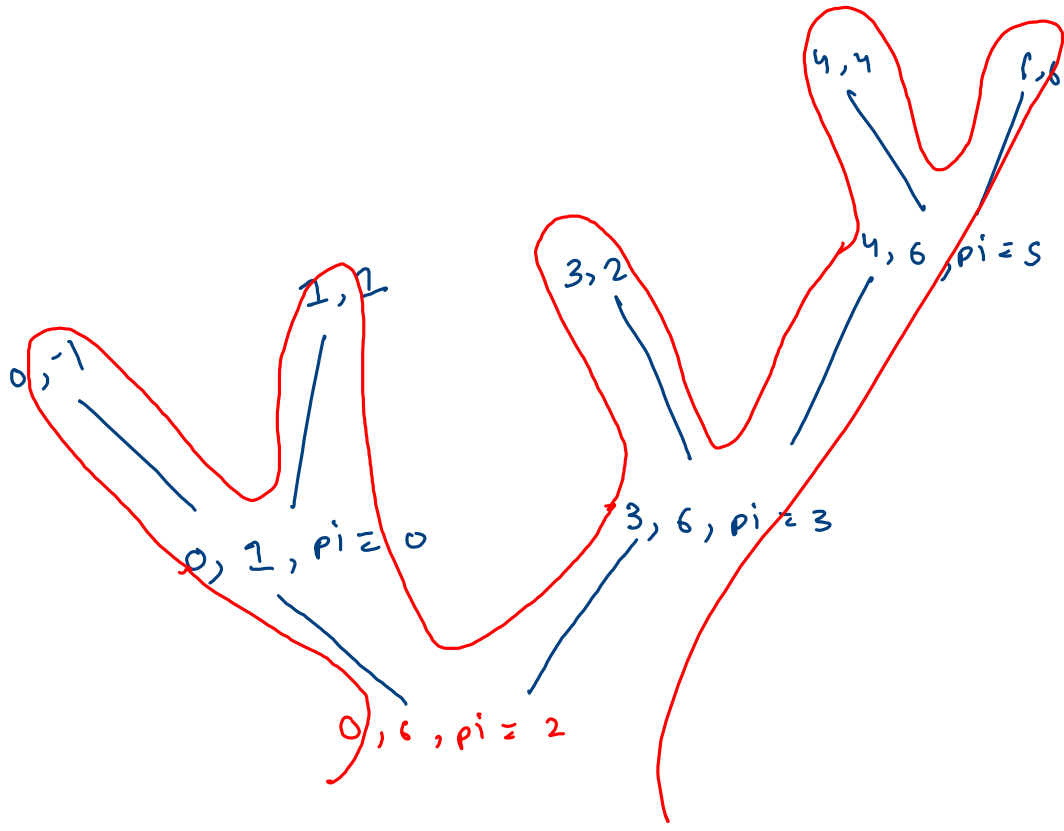


arr :

3	4	5	6	7	8	9
0	1	2	3	4	5	6

```java
public static void quickSort(int[] arr, int lo, int hi) {
    if(lo >= hi) {
        return;
    }

    int pi = partition(arr,arr[hi],lo,hi);
    quickSort(arr,lo,pi-1);
    quickSort(arr,pi+1,hi);

}
```

```java
int i = lo, j = lo;
while (i <= hi) {
  if (arr[i] <= pivot) {
    swap(arr, i, j);
    i++;
    j++;
  } else {
    i++;
  }
}
System.out.println("pivot index -> " + (j - 1));
return (j - 1);
```

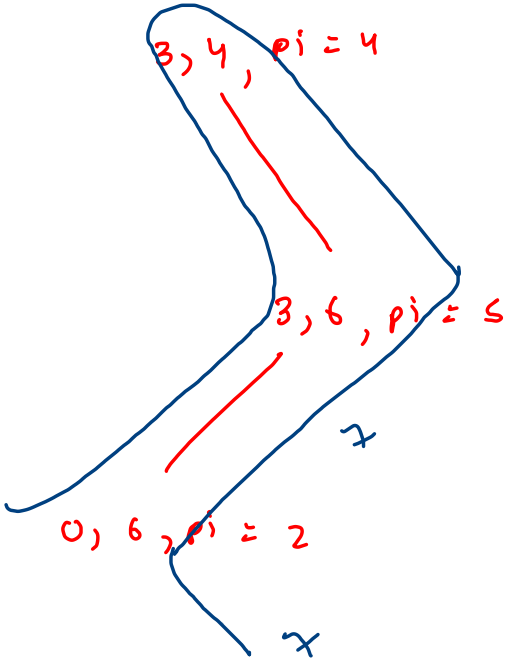


9 4 6 3 7 8 5

4	3	5	6	7	8	9
0	1	2	3	4	5	6

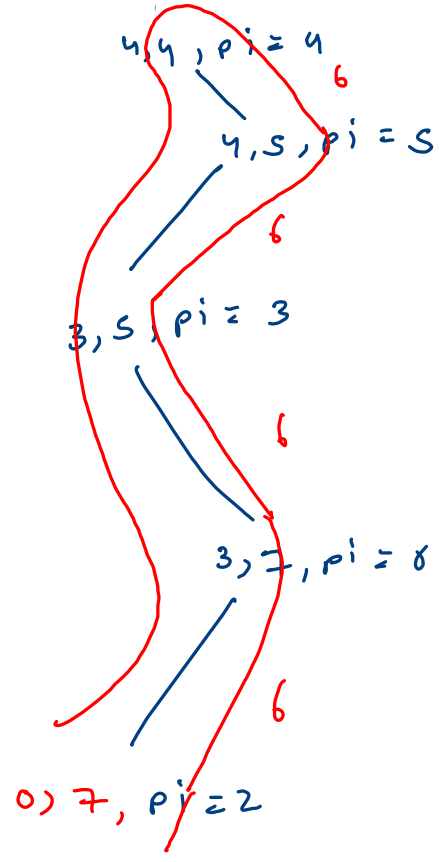
k = 5

idx = 4



3	2	4	5	6	7	8	9
0	1	2	3	4	5	6	7

```java
public static int quickSelect(int[] arr, int lo, int hi, int k) {
  //write your code here

    int pi = partition(arr,arr[hi],lo,hi);

    if(pi == k) {
        return arr[pi];
    }
    else if(pi < k) {
        return quickSelect(arr,pi+1,hi,k);
    }
    else {
        return quickSelect(arr,lo,pi-1,k);
    }
}
```



k = 4

(5th smallest)

Count Sort

9	6	3	5	3	4	3	9	6	4	6	5	8	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3	3	3	4	4	5	5	6	6	6	8	9	9	9	9

min = 3

max = 9

freq -> 9 - 3 + 1 = 7

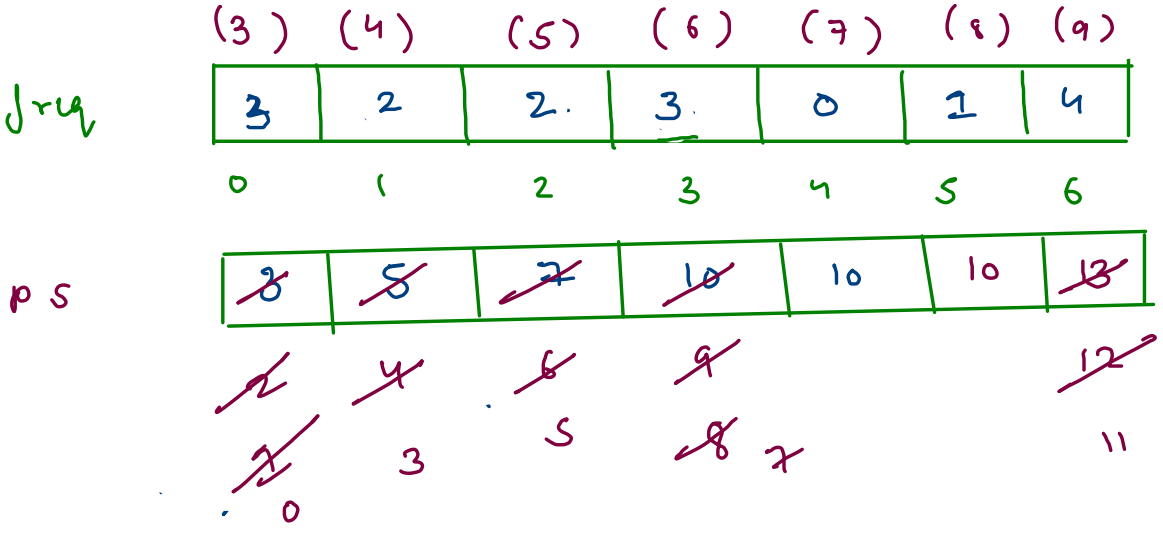
3	2	2	3	0	1	4
0	1	2	3	4	5	6

(3) (4) (5) (6) (7) (8) (9)

9	6	3	5	3	4	3	9	6	4	6	5	8	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	3	3	4	4	5	5	6	6	6	8	9	9	9	9

min = 3 max = 9

size = 7

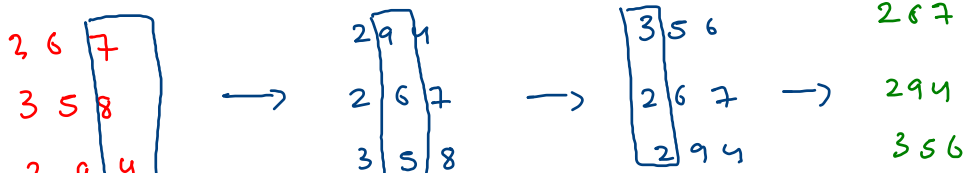


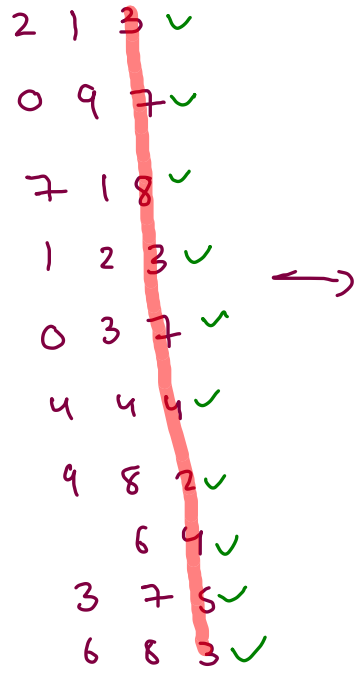

```
//write your code here
int size = max - min + 1;
int[]freq = new int[size];

//fill freq array
for(int i=0; i < arr.length;i++) {
    freq[arr[i]-min]++;
}

//prefix sum array
int[]ps = new int[freq.length];
ps[0] = freq[0];

for(int i=1; i < freq.length;i++) {
    ps[i] = ps[i-1] + freq[i]; //0 to i ka sum
}

//create ans array
int[]ans = new int[arr.length];

for(int i=arr.length-1; i >= 0;i--) {
    int idx = ps[arr[i] - min] - 1;
    ans[idx] = arr[i];
    ps[arr[i]-min]--;
}
```

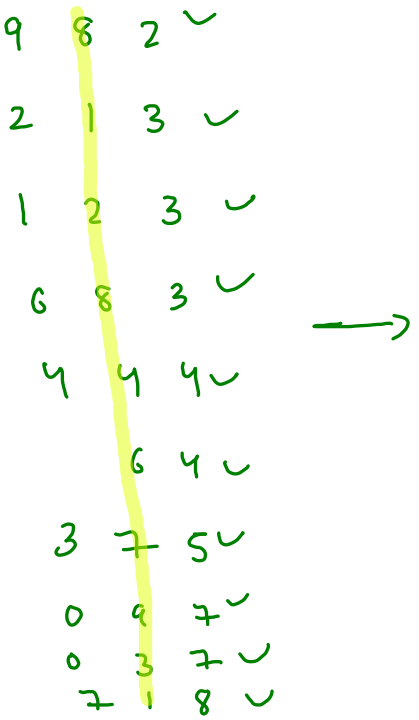
(i) stable sort

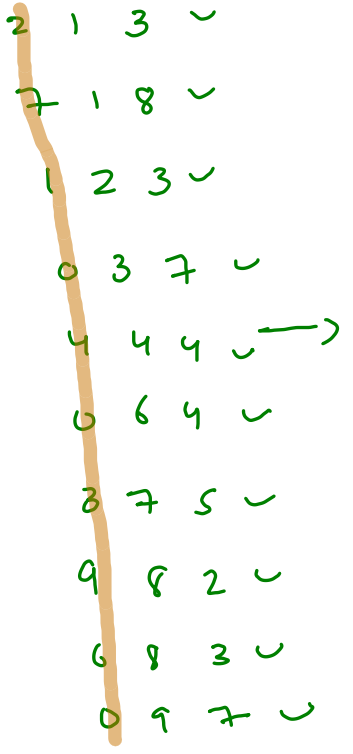
(ii) count sort → stable sort

(iii) why stability → radix sort









0 3 7
0 6 4
0 9 7
1 2 3
2 1 3
3 7 5
4 4 4
6 8 3
7 1 8
9 8 2