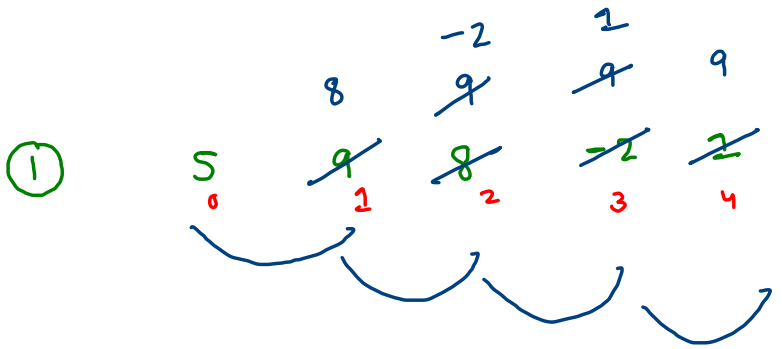
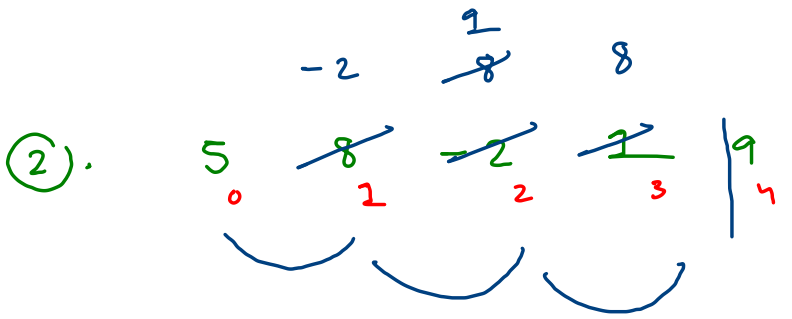
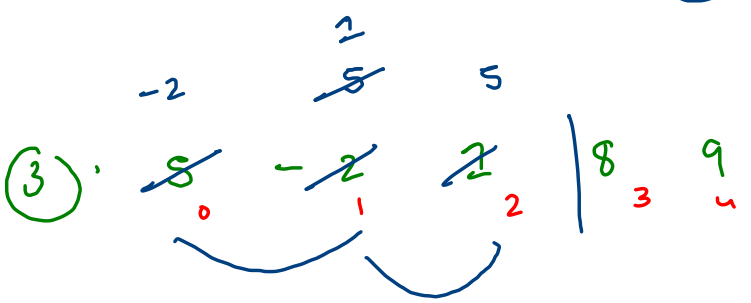
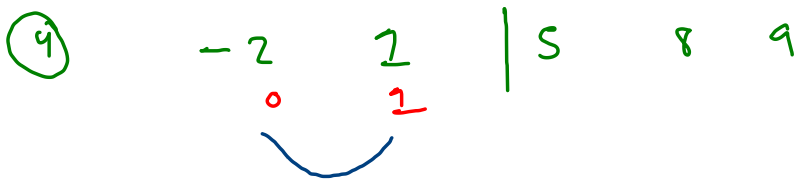
Bubble Sort

5 (0) 9 (1) 8 (2) -2 (3) 1 (4)









Comp = n - i

5 9 8 -2 1

0 1 2 3 4

```
for(int it = 1; it < n;it++) {
    for(int j = 0; j < n-it;j++) {
        if(isSmaller(arr,j+1,j) == true) {
            //arr[j+1] is smaller than arr[j]
            swap(arr,j+1,j);
        }
    }
}
```

it = 2

j = 0, 1, 2

1 -2 8 8

5 8 -2 1 9

0 1 2 3 4

j

selection sort

5 9 8 -2 1

0 1 2 3 4

① -2 9 8 5 1

0 1 2 3 4

i

② -2 | 1 9 8 5 9

0 | 1 2 3 4

i

③ -2 1 | 5 8 9

0 1 | 2 3 4

i

④ -2 1 5 | 8 9

0 1 2 | 3 4

i

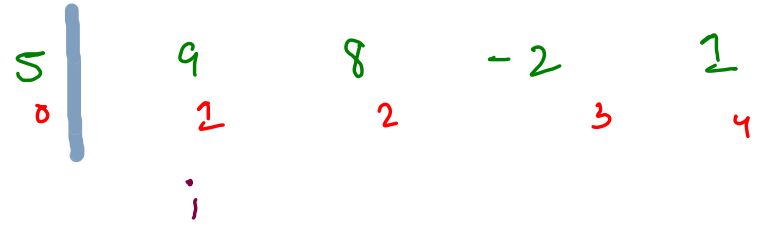

```java
public static void selectionSort(int[] arr) {
  //write your code here

  for(int i = 0 ; i < arr.length-1; i++) {
      int min = i;

      for(int j = i+1; j < arr.length;j++) {
          if(isSmaller(arr,j,min) == true) {
              min = j;
          }
      }

      swap(arr,i,min);
  }
```

Insertion sort

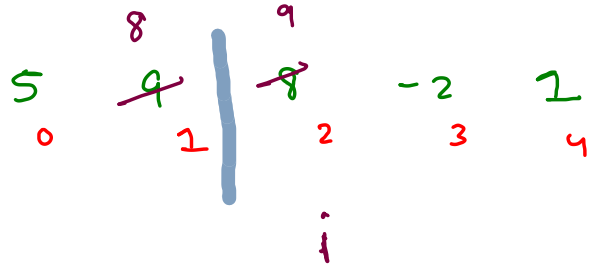
5 9 8 -2 1

0 1 2 3 4

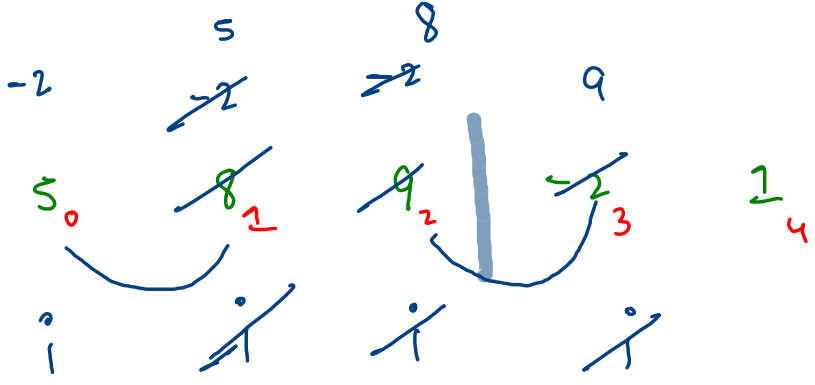
(1).



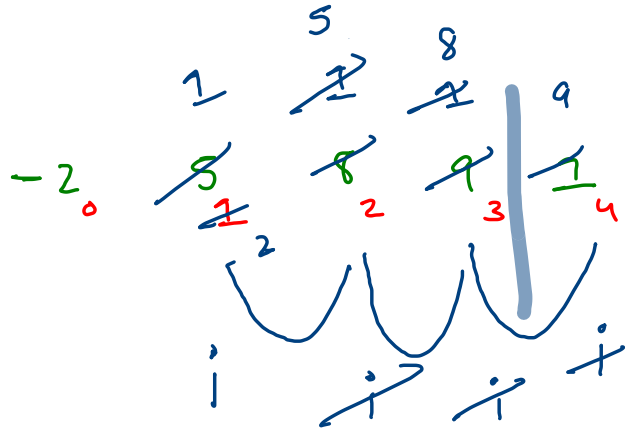
(2).



(3).



(4)



```java
for(int it = 1; it < arr.length;it++) {
    for(int i = it; i >= 1;i--) {
        if(isGreater(arr,i-1,i) == true) {
            swap(arr,i-1,i);
        }
        else {
            break;
        }
    }
}
```

7 −2 4 1 3

0 1 2 3 4

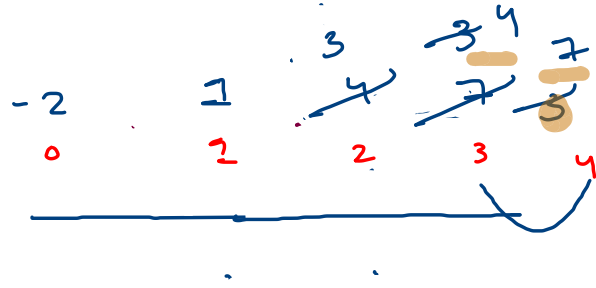
it i

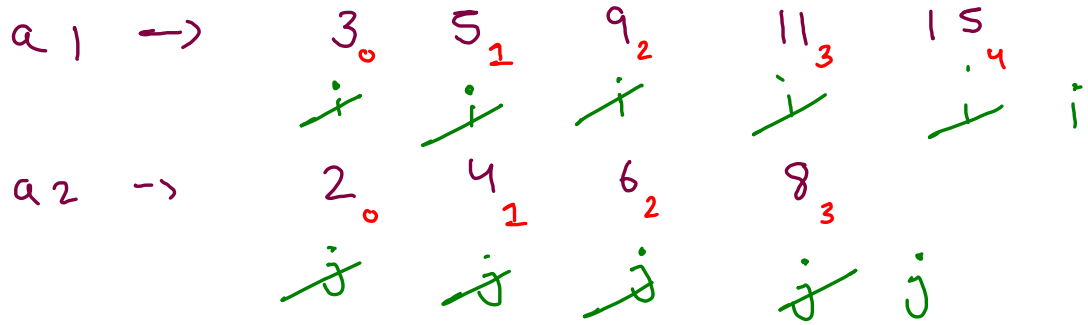
✓ 1 1

✓ 2 2 to 1

✓ 3 3 to 1

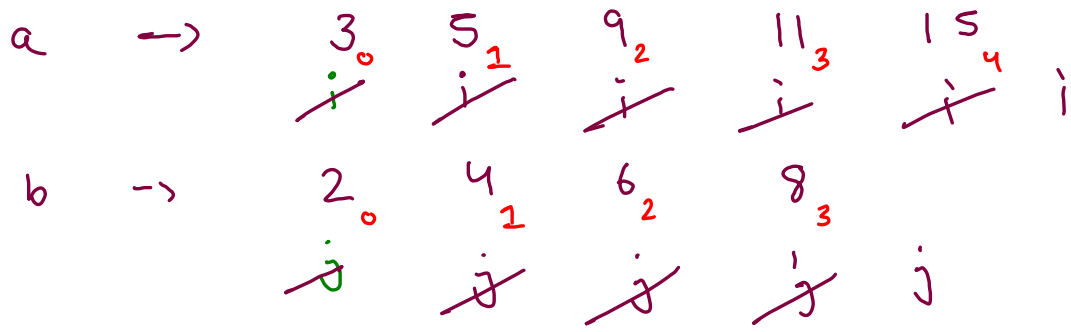
✓ 4 4 to 1



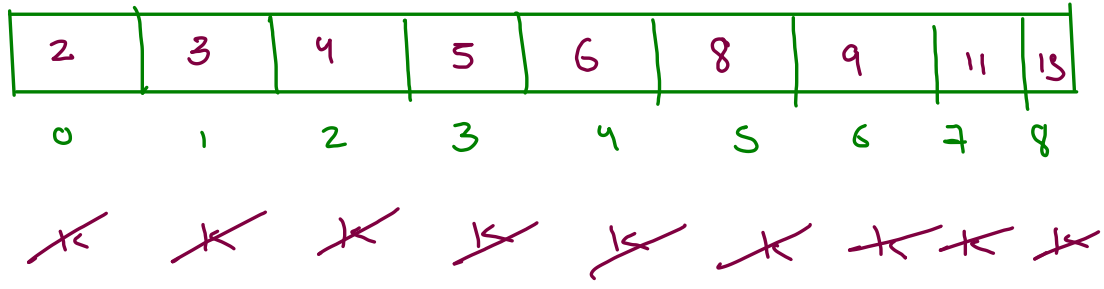


merged : 2 3 4 5 6 8 9 11 15

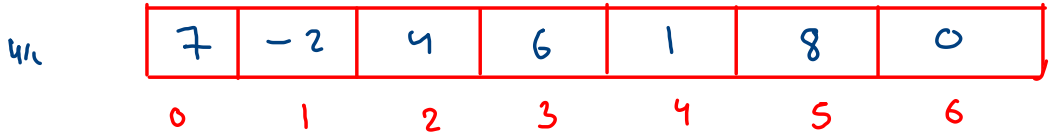
```
while( ) {
    if(a1[i] < a2[j]) {
        m[k] = a1[i];
        k++; i++;
    }
    else {
        m[k] = a2[j];
        k++; j++;
    }
}
```




```
while(i < a.length && j < b.length) {

    if(a[i] < b[j]) {
        merged[k] = a[i];
        i++;
        k++;
    }
    else {
        merged[k] = b[j];
        j++;
        k++;
    }
}
```




```
//if some elements are left in 'a'
while(i < a.length) {
    merged[k] = a[i];
    i++;
    k++;
}

//if some elements are left in 'b'
while(j < b.length) {
    merged[k] = b[j];
    j++;
    k++;
}
```



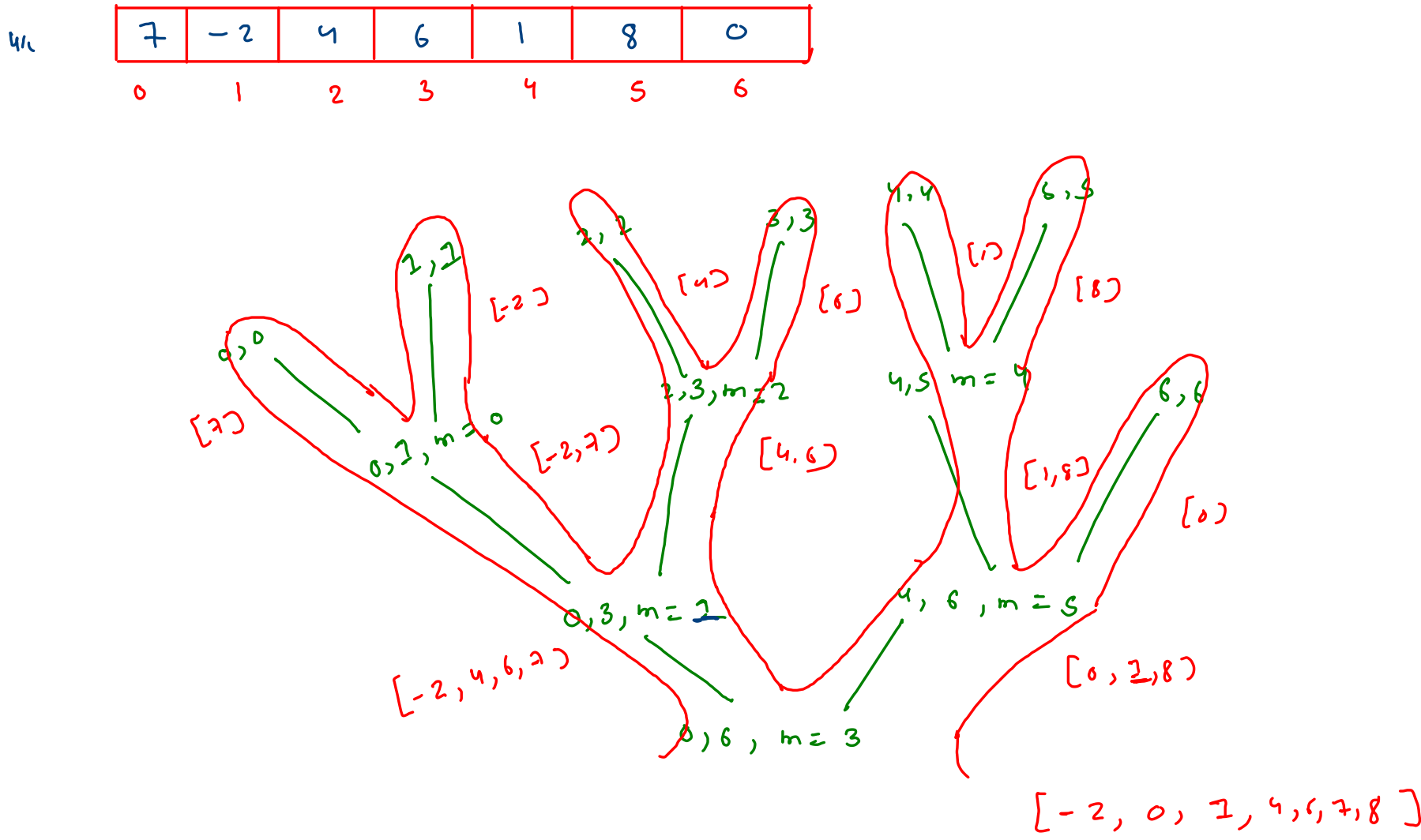

```
int[] ms (int[]arr, int l, int h) {

    int m = (l+h)/2 ;

    left ← ms(arr, l, m);

    right ← ms(arr, m+1, hi)

    merged = mSA(left,right);

    return merged;

}
```

0 -2 1 4 6 7 8

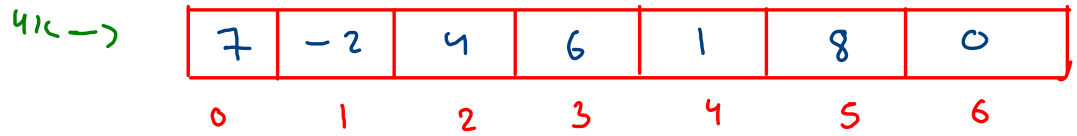
-2 4 6 7 | 0 1 8

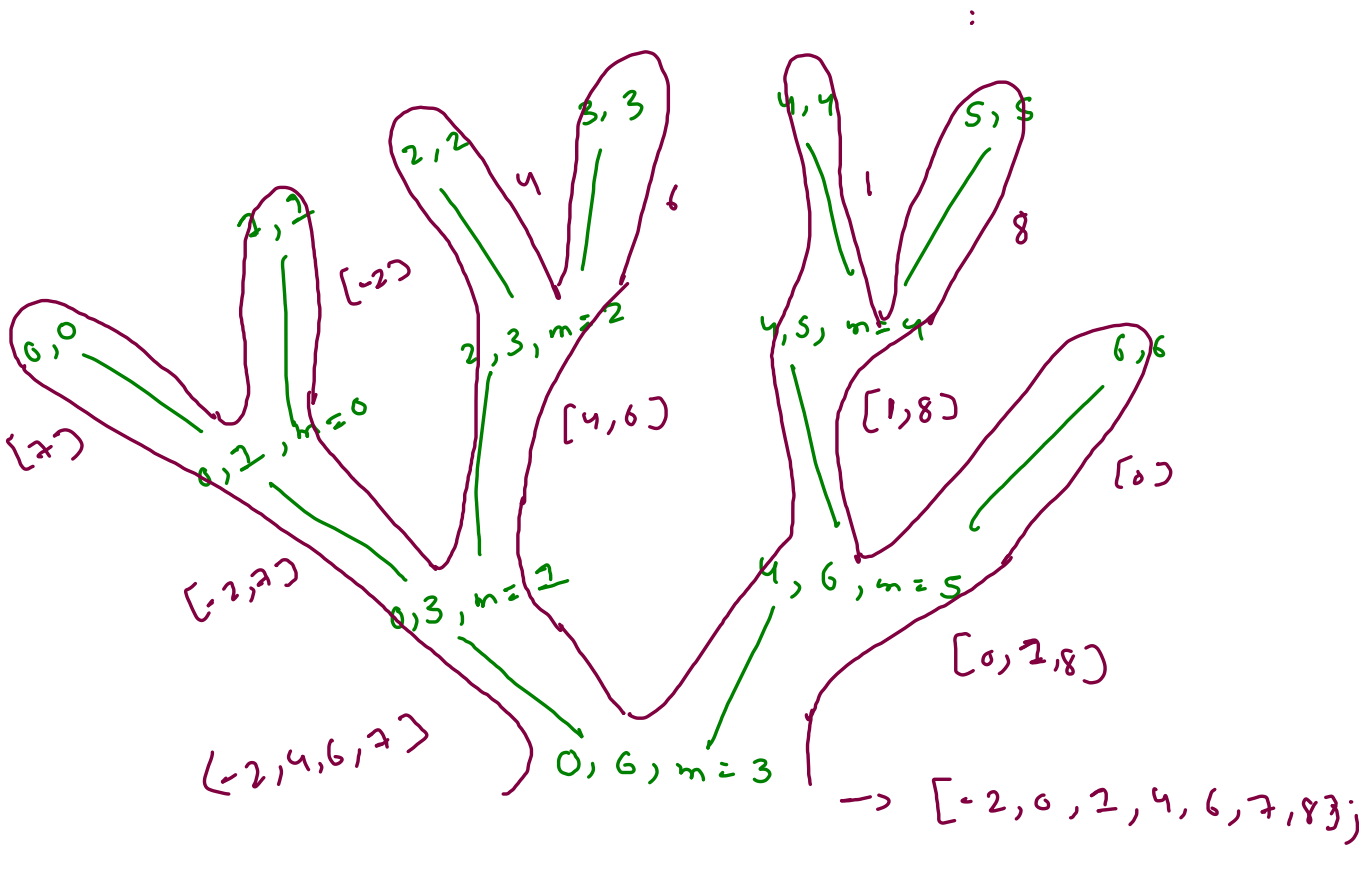
do to mid | mid+1 to hi

lo = 0, hi = 6, m = 3

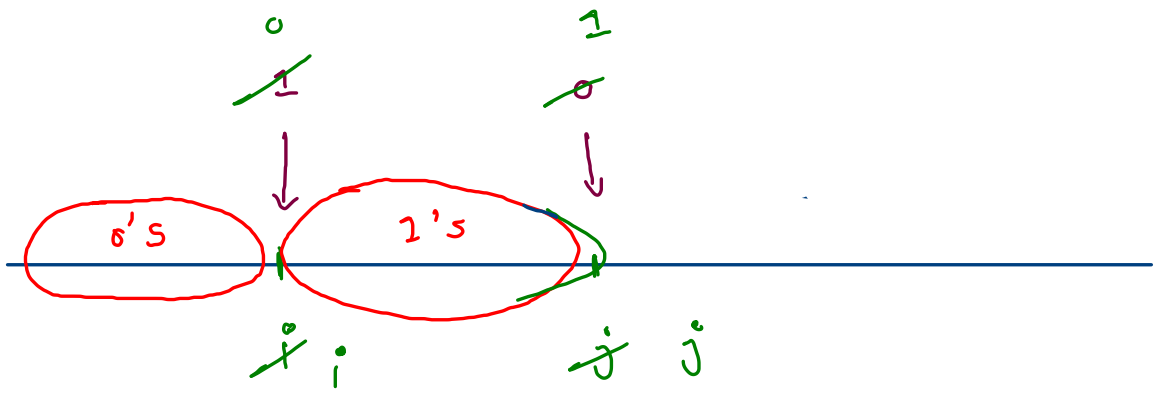

7 -2 4 6 1 8 0
0 1 2 3 4 5 6
0,0
[7]
1,1
[-2]
0,1, m=0
[-2,7]
2,2
[4]
3,3
[6]
2,3, m=2
[4,6]
0,3, m=1
[-2,4,6,7]
0,6, m=3
4,4
[1]
6,6
[8]
4,5 m=4
[1,8]
6,6
[0]
4, 6, m=5
[0,1,8]
[-2, 0, 1, 4, 6, 7, 8]

```java
public static int[] mergeSort(int[] arr, int lo, int hi) {
    //write your code here
    if(lo == hi) {
        int[]ba = {arr[lo]};
        return ba;
    }

    int mid = (lo + hi)/2;

    int[]left = mergeSort(arr,lo,mid);
    int[]right = mergeSort(arr,mid+1,hi);

    int[]merged = mergeTwoSortedArrays(left,right);

    return merged;
}
```





arr : 0 1 0 1 0 1 0 0 1 0



(i) extra space X

(ii) single traversal

(iii) O(n)

```
if (arr[j] == 1) {
    j++;
}
```

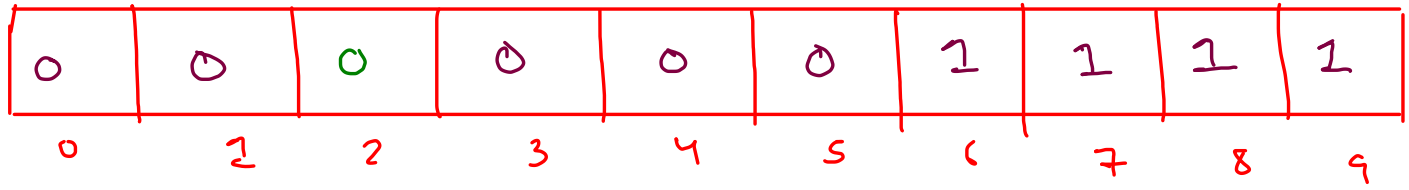
```
else {
    swap(arr, i, j)
    i++; j++;
}
```

0 to i-1 -> 0's

i to j-1 -> 1's

j to end -> uh

```java
public static void sort01(int[] arr){
    //0 to i-1 -> 0's
    //i to j-1 -> 1's
    //j to end -> unknown

    int i = 0;
    int j = 0;

    while(j < arr.length){
        if(arr[j] == 1) {
            j++;
        }
        else {
            swap(arr,j,i);
            i++;
            j++;
        }
    }
}
```

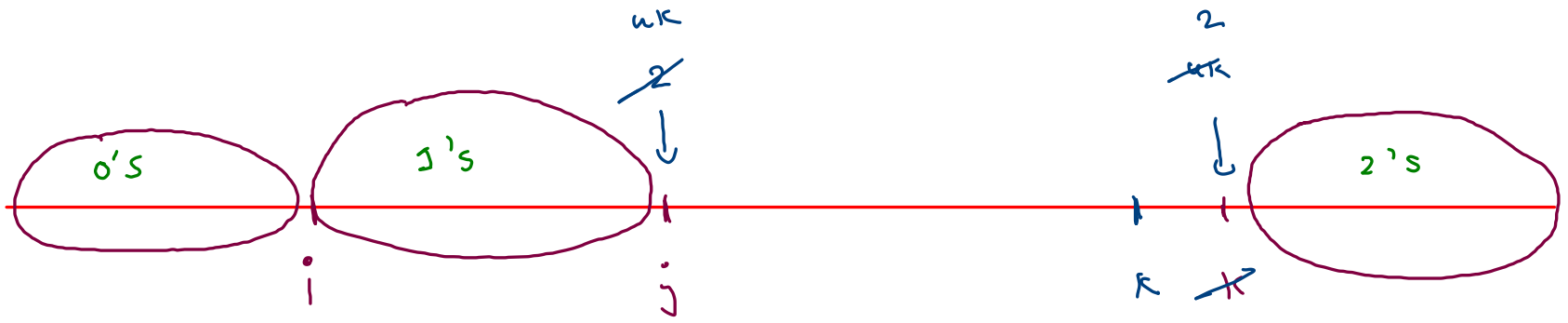


0 to i-1 -> 0's

i to j-1 -> 1's

j to end -> ukn.

arr : 2 1 0 0 2 0 1 1 0 2



arr[j] == 1	arr[j] == 0	arr[j] == 2

j++	swap(i,j);
	i++;
	j++;
		swap(j,k)
		k--;

0 to i-1 = 0's

i to j-1 = 1's

j to k = unk

k+1 to end = 2's

arr : 0 0 0 0 1 1 1 1 2 2

i k j

```
int i = 0;
int j = 0;
int k = arr.length-1;

while(j <= k) {

    if(arr[j] == 1) {
        j++;
    }
    else if(arr[j] == 0) {
        swap(arr,j,i);
        i++;
        j++;
    }
    else {
        swap(arr,j,k);
        k--;
    }
}
```

0 to i-1 = 0's

i to j-1 = 1's

j to k = 4k

k+1 to end = 2's