

Merge Two Sorted Arrays

a_1 : 5₀ 9₁ 11₂ 15₃ 16₄ 19₅

a_2 : 3₀ 6₁ 8₂ 10₃ j

merge: 3₀ 5₁ 6₂ 8₃ 9₄ 10₅ 11₆ 15₇ 16₈ 19₉ k

$(n+m)$

```

int n = a.length;
int m = b.length;

int[] marr = new int[n+m]; //merged array

int i = 0; //a
int j = 0; //b
int k = 0; //marr

while(i < n && j < m) {
    if(a[i] < b[j]) {
        marr[k] = a[i];
        i++;
        k++;
    }
    else {
        marr[k] = b[j];
        j++;
        k++;
    }
}

//check if elements of first array are Left
while(i < n) {
    marr[k] = a[i];
    k++;
    i++;
}

//check if elements of second array are Left
while(j < m) {
    marr[k] = b[j];
    k++;
    j++;
}

```

a : 5₀ 9₁ 11₂ 15₃ 16₄ 19₅ ;

b : 3₀ 6₁ 8₂ 11₃

j

marr

3	5	6	8	9	11	11	15	16	19
0	1	2	3	4	5	6	7	8	9

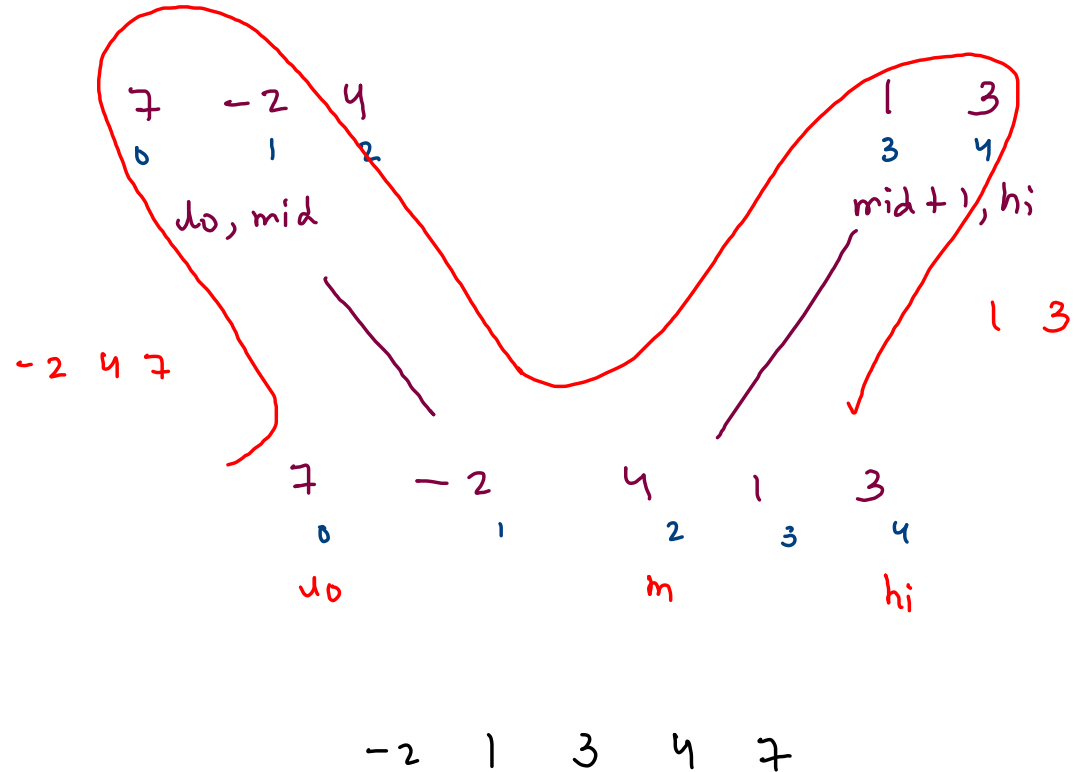
$O(n+m)$

Merge Sort

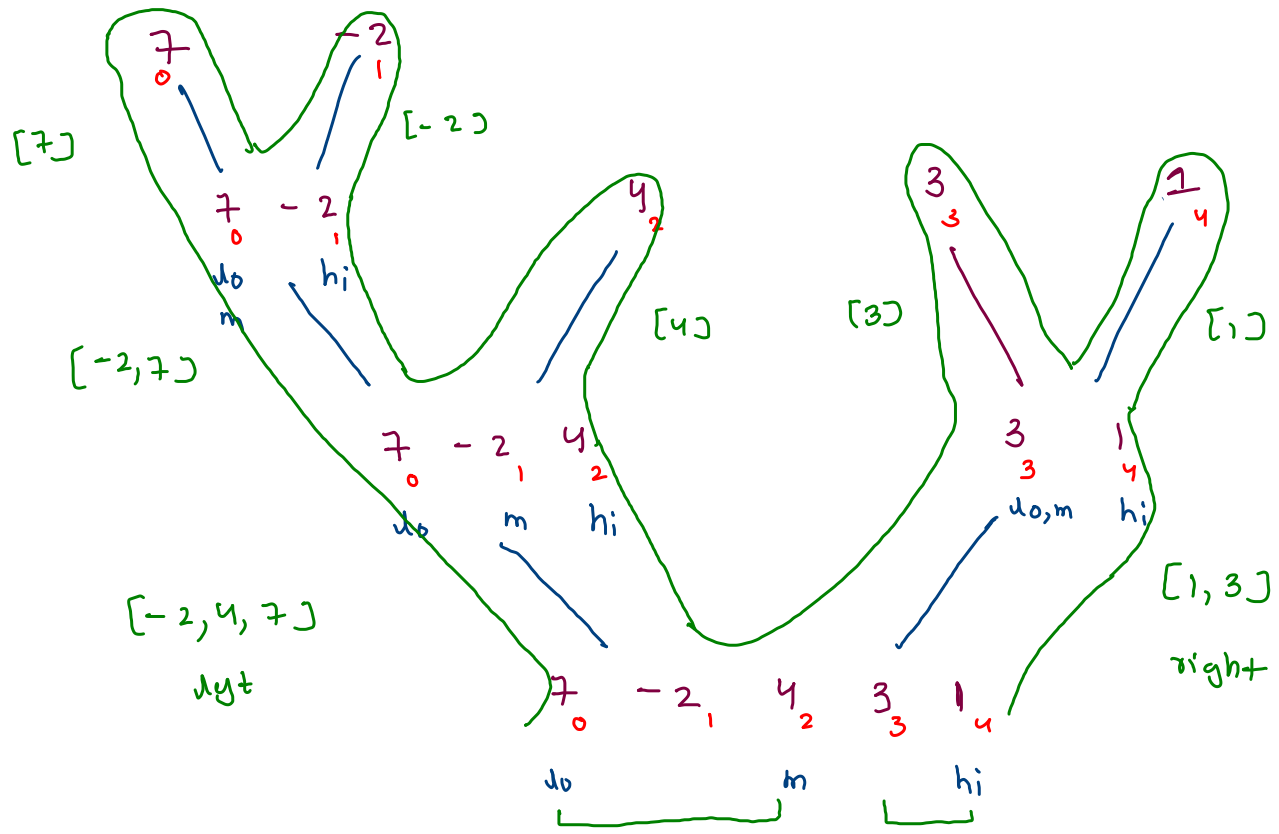
(i) sort

(ii) recursion

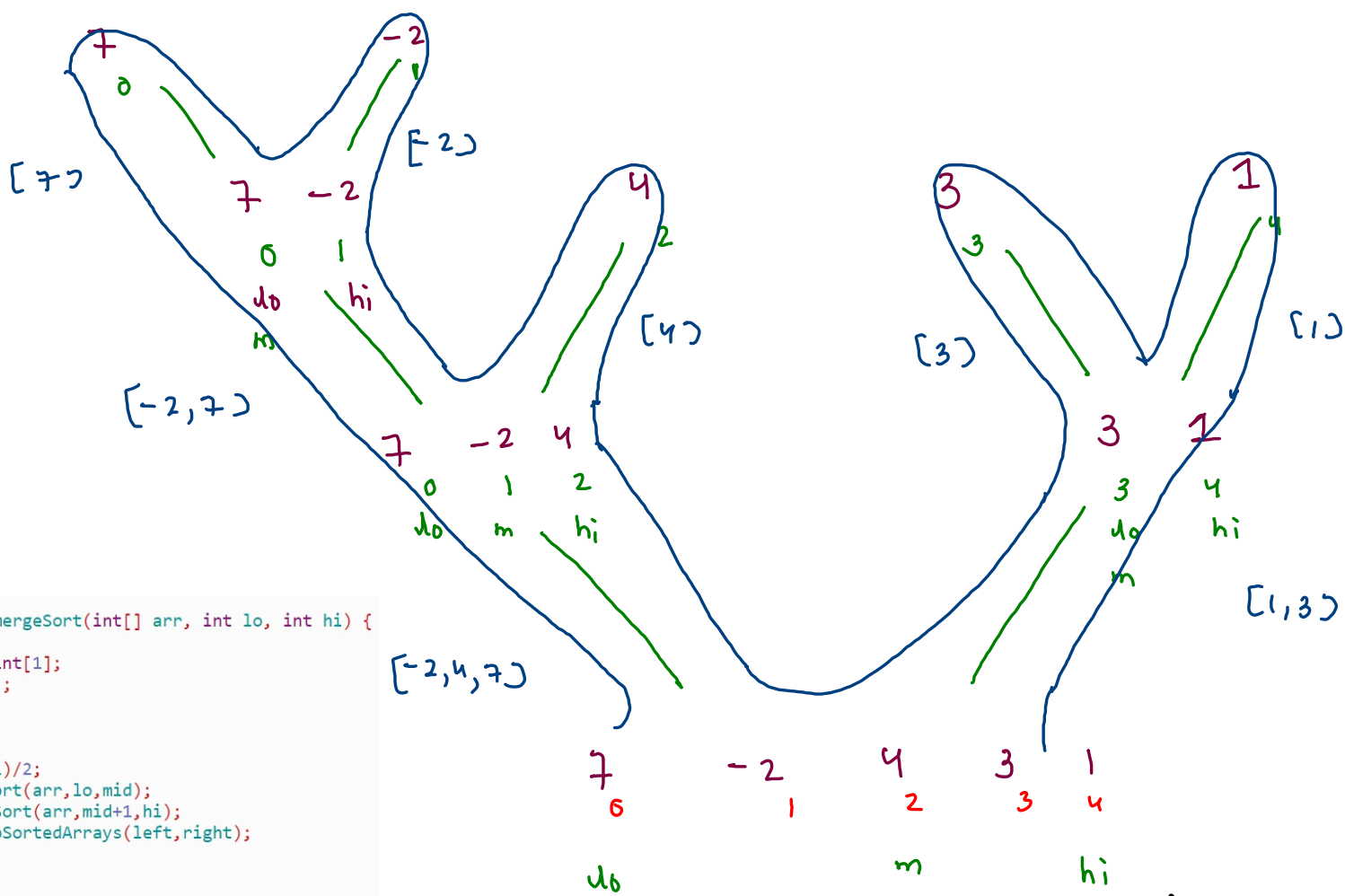
(iii) $\text{int}[] \text{left} = \text{ms}(\text{lo}, \text{mid});$
 $\text{int}[] \text{right} = \text{ms}(\text{mid}+1, \text{hi});$
 $\text{int}[] \text{ans} = \text{m2sA}(\text{left}, \text{right});$



merge sort :



ans: $[-2, 1, 3, 4, 7]$ (merge two sorted array (left, right))



```

public static int[] mergeSort(int[] arr, int lo, int hi) {
    if(lo == hi) {
        int[] ba = new int[1];
        ba[0] = arr[lo];
        return ba;
    }

    int mid = (lo + hi)/2;
    int[] left = mergeSort(arr, lo, mid);
    int[] right = mergeSort(arr, mid+1, hi);
    int[] ans = mergeTwoSortedArrays(left, right);

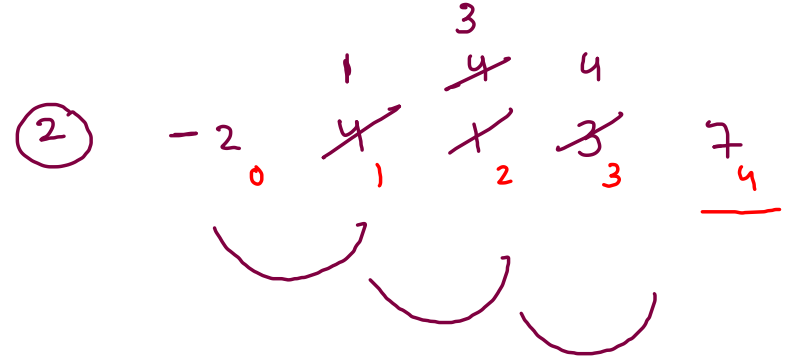
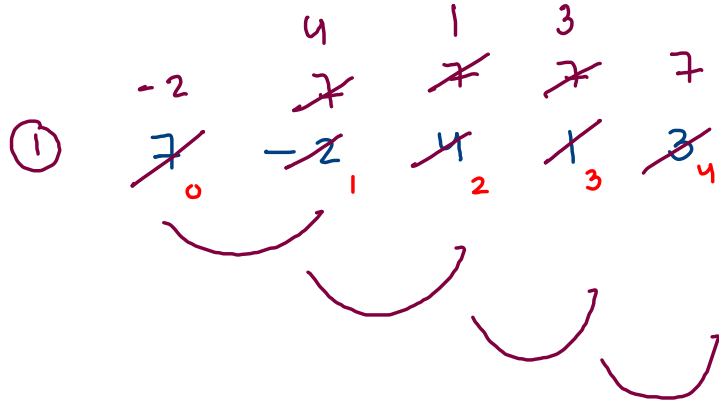
    return ans;
}

```

ans: -2 0 1 3 4 7

Bubble Sort

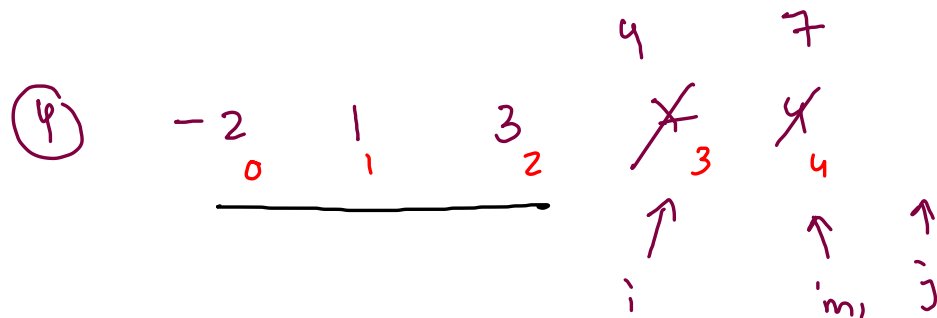
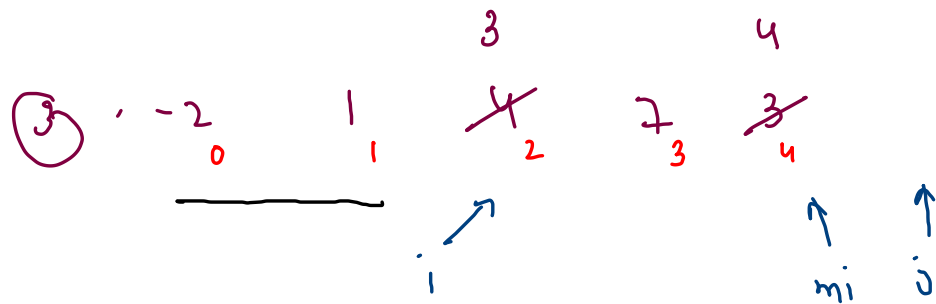
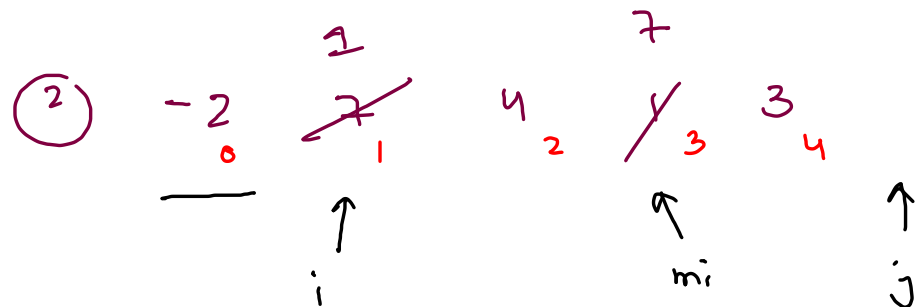
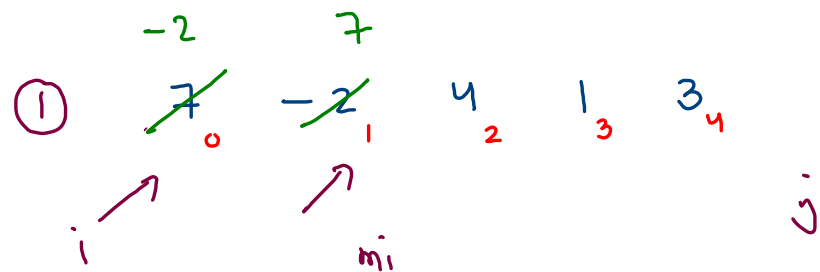
7₀ - 2₁ 4₂ 1₃ 3₄



$i + 1 : n - 1$
 [comparison

Selection Sort

7₀ -2₁ 4₂ 1₃ 3₄



Insertion Sort

7₀ -2₁ 4₂ 1₃ 3₄

①. ⁻² ~~7~~₀ | ⁷ ~~-2~~₁ 4₂ 1₃ 3₄

②. -2₀ ~~7~~₁ | ~~4~~₂ 1₃ 3₄

③. -2₀ ~~4~~₁ ~~7~~₂ | 1₃ 3₄

④. -2₀ 1₁ ~~4~~₂ ~~7~~₃ | ~~3~~₄

itr
[comparisons


```

for(int itr = 1; itr < n; itr++) {
    for(int j = itr; j >= 1; j--) {
        if(isGreater(arr, j-1, j) == true) {
            //swapping is required
            swap(arr, j-1, j);
        }
        else {
            break;
        }
    }
}

```

7₀ -2₁ 4₂ 1₃ 3₄

(i) itr = 1

~~7~~₀ ~~-2~~₁ 4₂ 1₃ 3₄

(ii) itr = 2


-2₀ ~~7~~₁ | ~~4~~₂ 1₃ 3₄


(iii) itr = 3


-2₀ ~~4~~₁ ~~7~~₂ | ~~1~~₃ 3₄


(iv) itr = 4

-2₀ 1₁ ~~4~~₂ ~~7~~₃ | ~~3~~₄

$\overset{2}{\cancel{7}}_0$ $\overset{7}{\cancel{2}}_1$ $\overset{4}{2}_2$ $\overset{2}{3}_3$ $\overset{3}{4}_4$


2 $\overset{4}{\cancel{7}}$ $\overset{7}{\cancel{4}}$ 2 3


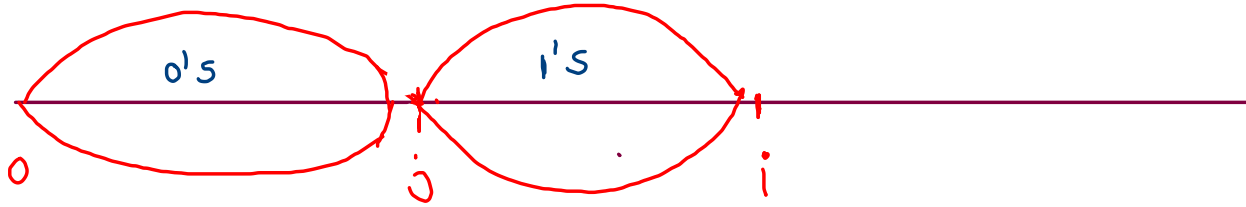
2 $\overset{2}{\cancel{4}}$ $\overset{4}{\cancel{7}}$ $\overset{7}{2}$ 3


2 2 $\overset{3}{\cancel{4}}$ $\overset{4}{\cancel{7}}$ $\overset{7}{3}$


Sort 01

Swap

0 1 0 0 1 0 1 1 1

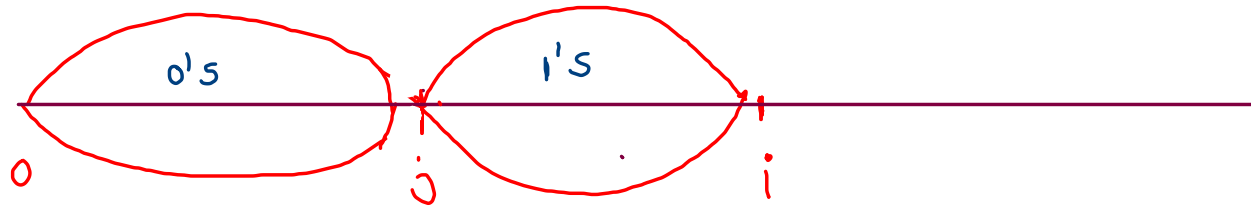


- (i) linear time
- (ii) no extra space
- (iii) single traversal

0's \rightarrow 0 to $j-1$

1's \rightarrow j to $i-1$

unknowns \rightarrow i to end



0's \rightarrow 0 to $j-1$

1's \rightarrow j to $i-1$

unknowns \rightarrow i to end

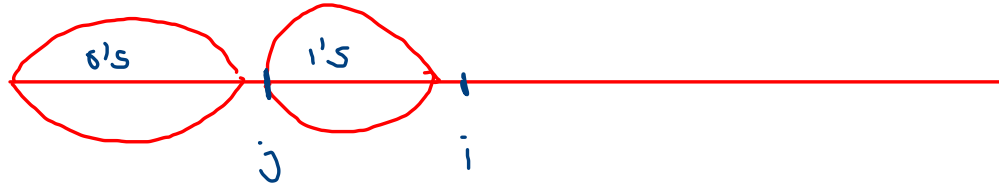
$arr[j] == 1$	$arr[i] == 0$
$i++;$	$swap(i, j);$
	$i++;$
	$j++;$

```

int i = 0;
int j = 0;

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

```



0 to $j-1 \rightarrow$ 0's
 j to $i-1 \rightarrow$ 1's
 i to $n-1 \rightarrow$ unknown

