Merge Sort

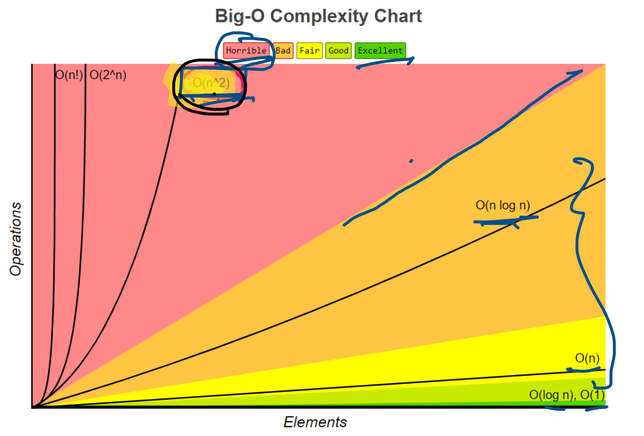
27 June 2022 20:09

Basic algo:-
1. Bubble Sort
2. Selection Sort
3. Insertion Sort.

1. InsertionSort - worst - $O(n^2)$
Best - $O(n)$

2. Bubble Sort - worst - $O(n^2)$
Best - $O(n)$ / n^3

3. Selection Sort - worst - $O(n^2)$
Best - n^3



Merge Sort:-

* merge two sorted arrays.

Q Two arrays a1 & a2

I/P:- a1 :- [2,4,5,6] // sorted 4 2
a2 :- [1,3,4,8] // sorted 4 3
5

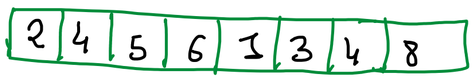
Merge two sorted arrays :-

O/P:- [1,2,3,4,4,5,6,8] // 8

Two approaches:

Approach 1:- length of a1 = m
length of a2 = n

1. Create an empty array of size m+n — O(1)

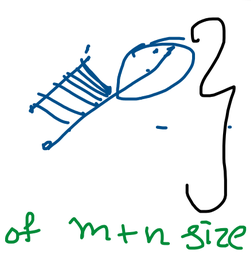


2. Copy elem of a1 & a2 into new array. — O(m+n)
3. Apply any of sorting algo known to the res. array. — B.I.S — $O((m+n)^2)$

TC = O(1) + O(m+n) + $O((m+n)^2)$
TC = $O((m+n)^2)$

Approach-2

a1 = [2, 4, 5, 6] P1
a2 = [1, 3, 4, 8]

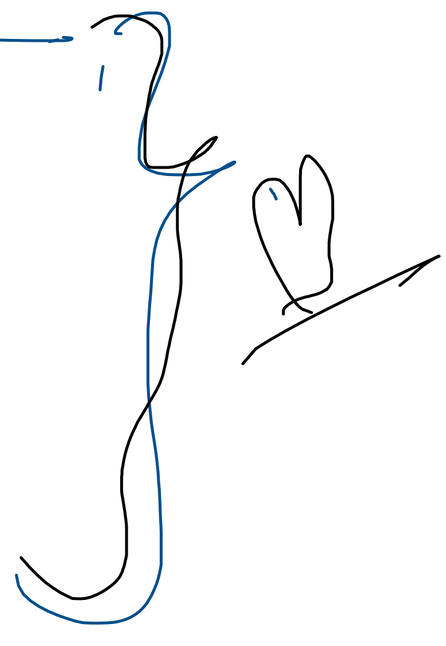


Create an empty array of m+n size

res = [1 2 3 4 4 5 6 8] k

a1[P1] < a2[P2]

```
if (a1[P1] < a2[P2]){
    res[k] = a1[P1]
    P1++;
    k++;
}
else {
    res[k] = a2[P2]
    P2++;
    k++;
}
```



```
function MergeSortedArrays(a1,a2){
    //Create an array
    let res = [];
```

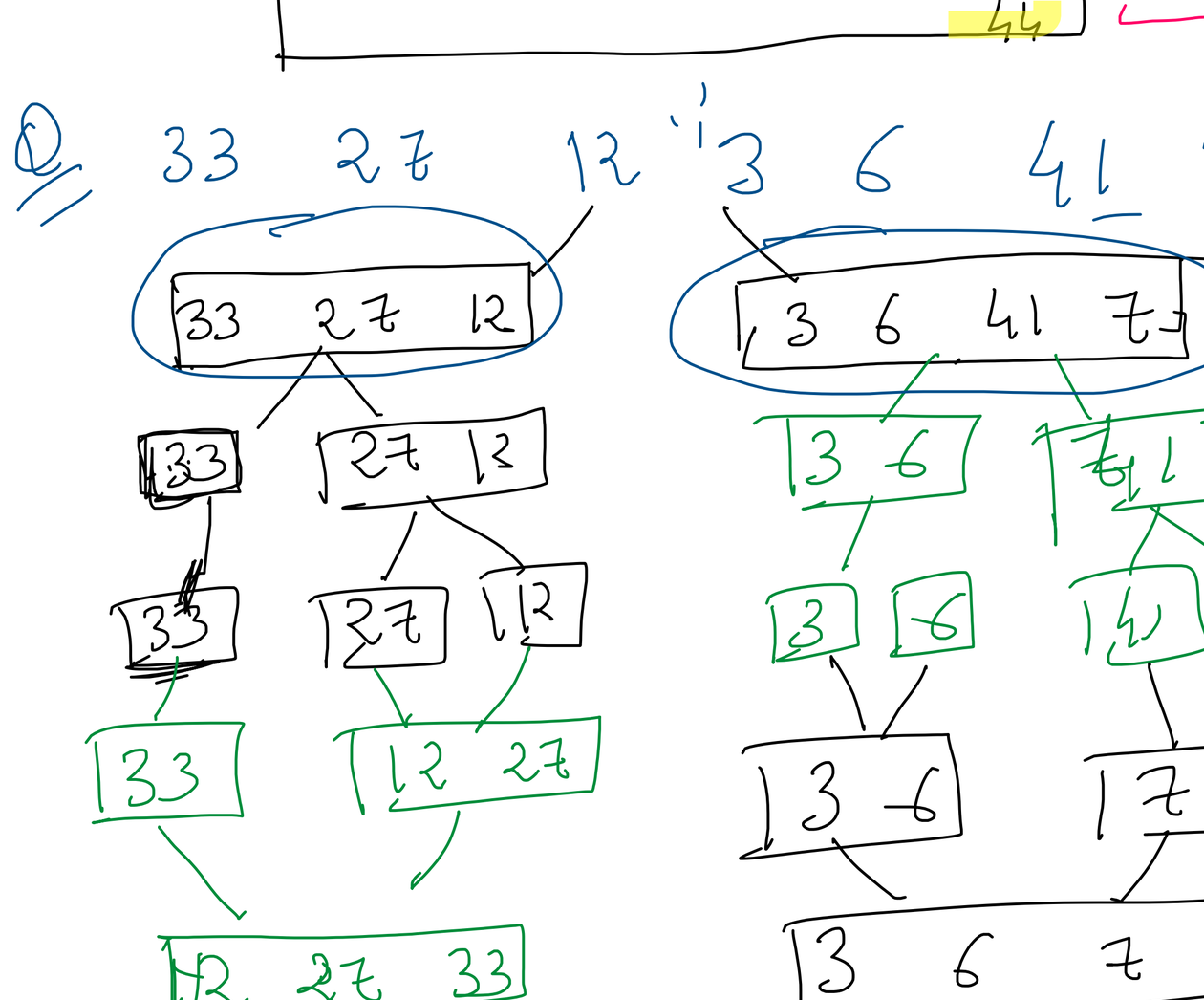
```
    let p1 = 0; //a1
    let p2 = 0; //a2
    let k = 0; //res
    while(p1 < a1.length && p2 < a2.length){
        if(a1[p1] < a2[p2]){
            res[k] = a1[p1];
            p1++;
            k++;
        }
        else{
            res[k] = a2[p2];
            p2++;
            k++;
        }
    }
    if(p1 == a1.length){
        while(p2 != a2.length){
            res[k] = a2[p2];
            k++;
            p2++;
        }
    }
    if(p2 == a2.length){
        while(p1 != a1.length){
            res[k] = a1[p1];
            k++;
            p1++;
        }
    }
    return res;
}
console.log(MergeSortedArrays([2,4,5,6],[1,3,4,8]));
```

Merge Sort.

* Merge Sort:

1. Divide & conquer:

1. Divide & conquer:



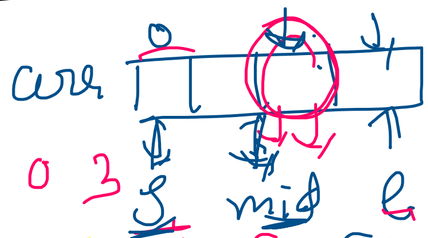

```
function MergeSort(arr,start,end){
    if(start>=end) return;

    let mid = parseInt((start+end)/2);
    MergeSort(arr,start,mid);
    MergeSort(arr,mid+1,end);
    merge(arr,start,mid,end);
}
```

Divide

Conquer

Q. 33 27 12 3 6 41 7

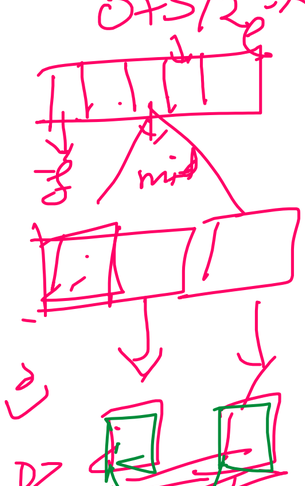



```
function mergeSort(arr, s, e) {
    if (start >= end) {
        return;          // Base case
    }
    var mid = parseInt((start+end)/2)
    mergeSort(arr, start, mid)
    mergeSort(arr, mid+1, end)
    mergeTwoSortedArray(arr, start, mid, end)
}
```

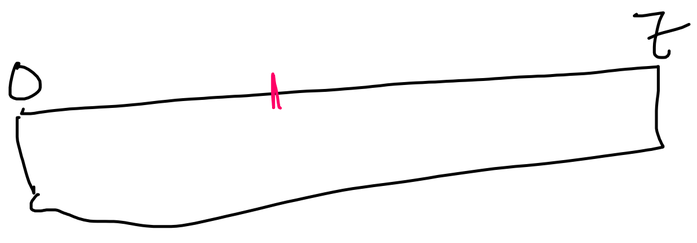
(a.length-1)

$\frac{0+7}{2} \Rightarrow 3$

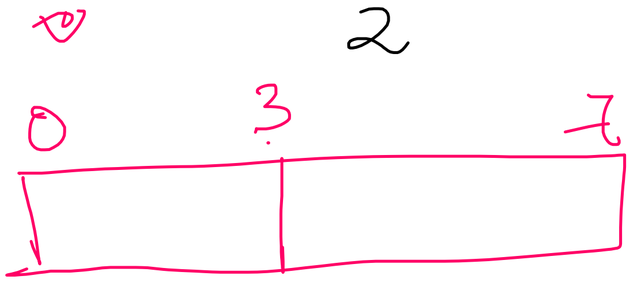
$0+5/2 \Rightarrow 2$



$\frac{0+3}{2} \Rightarrow 1$



$mid = \frac{0+7}{2} = 3$



$\frac{3+0}{2} \Rightarrow 1$

$\frac{3+2}{2} \Rightarrow 5/2 \Rightarrow 2$

$\frac{0+1}{2} \Rightarrow 0$

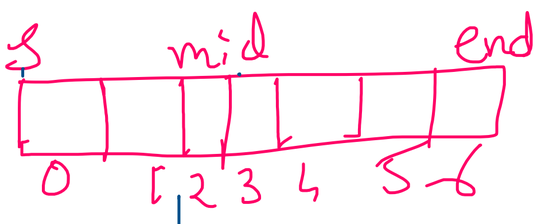
$\frac{2+3}{2} \Rightarrow 5/2 \Rightarrow 2$

$\frac{0+3}{2} \Rightarrow 1$

$\frac{7+0}{2} \Rightarrow 3$

$\frac{7+4}{2} \Rightarrow 5$

$\frac{5+7}{2} \Rightarrow 6$



let a1 = [| | |]

let a2 = [|]

mid-3

mid-start+1

3-0+1

end-mid

6-3

6-3

3

3+1+0

0+0

3+1+1

0+1

3+1+2

0+2

0+3

1

mid+1+j

3-0+1

4

```
MergeSort.js > ...
        let m1 = mid - start + 1;
        let m2 = end - mid;

        let a1 = new Array(m1);
        let a2 = new Array(m2);

        for(let i = 0;i<m1;i++){
            a1[i] = arr[start + 1];
        }
        for(let j = 0;j<m2;j++){
            a2[i] = arr[mid+1+j];
        }

    }   mergeSorted

    //start + i => 0+0 0+1 0+2
```