Todays Content

1. Re-arrange array
2. Quick sort

Q. Given an ar[N]: re-arrange it such that

Bring last element to its correct sorted position:

All values < lastele are continously on leftside of lastelement

All Value >= last ele are continously on rightside of lastelement

Note: Without extra space

	0	1	2	3	4	5	6	7
Ex: ar[8]:	9	8	1	6	5	11	4	7

	0	1	2	3	4	5	6	7
ans1: ar[8]:	1	4	5	6	7	9	8	11
ans2: ar[8]:	6	4	1	5	7	11	8	9
ans3: ar[8]:	1	4	5	6	7	8	9	11

#Note: Multiple ans are possible return any one of them.

	0	1	2	3	4	5	6	7
Ex2: ar[9]:	5	13	7	8	25	20	23	10

	0	1	2	3	4	5	6	7
ans1: ar[8] =	7	5	8	10	25	20	23	13
ans2: ar[8] =	5	7	8	10	13	20	23	25

#Ideal: Sort ar[]

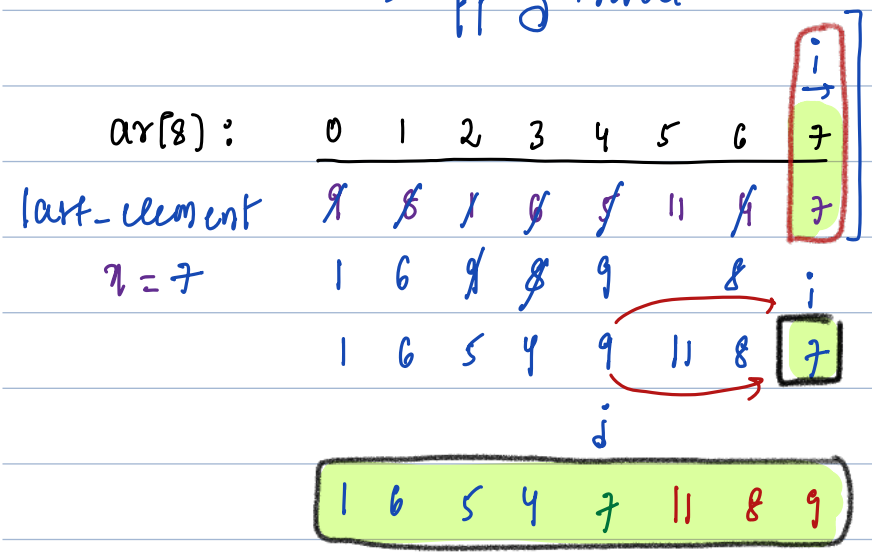
1. Using BS/SS/QS TC: O(N^2)
2. Inbuild/Merge TC: O(NlogN)

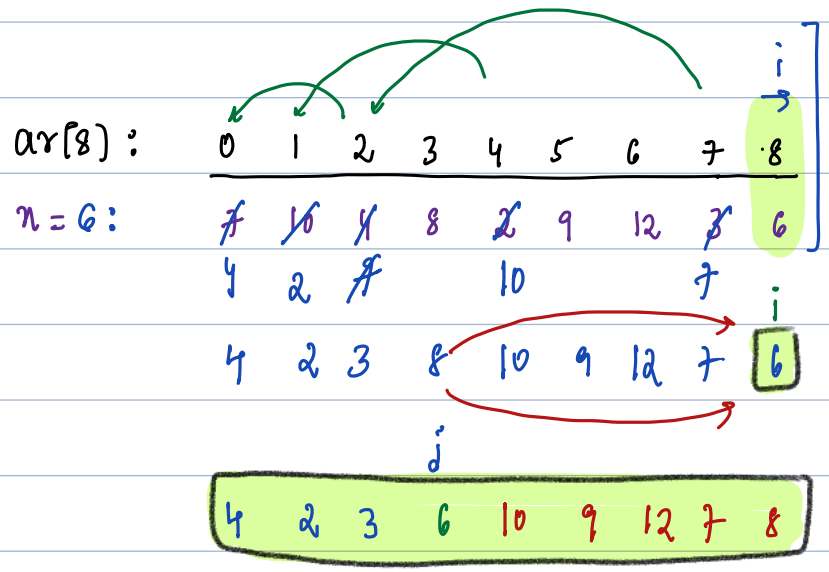
#Idea2:

Take 2 indices i & j

i: Iterate m array

j: To track swapping index





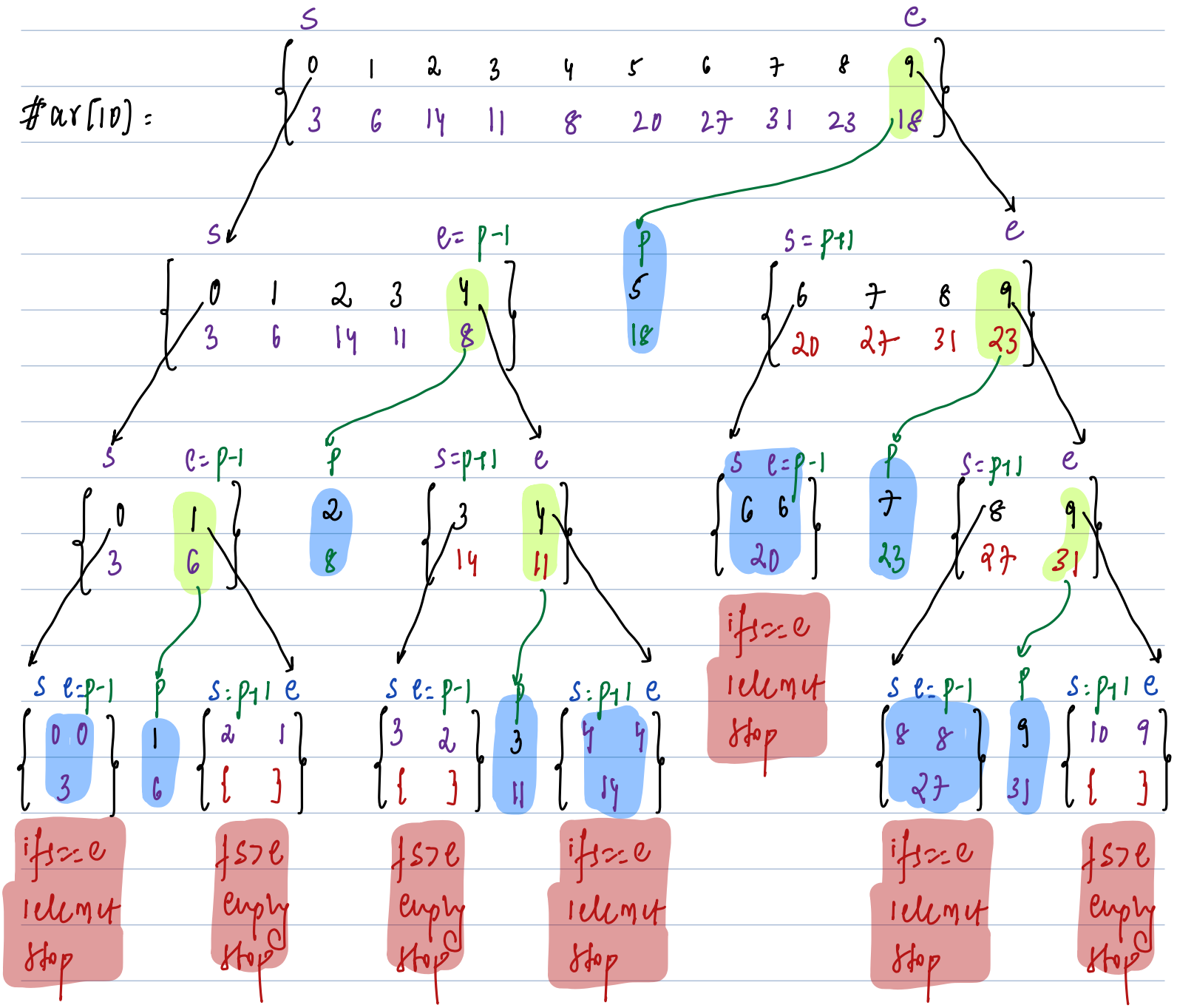
#obs:

if(ar[i]<n) { swap ar[i] & ar[j]; j++; }

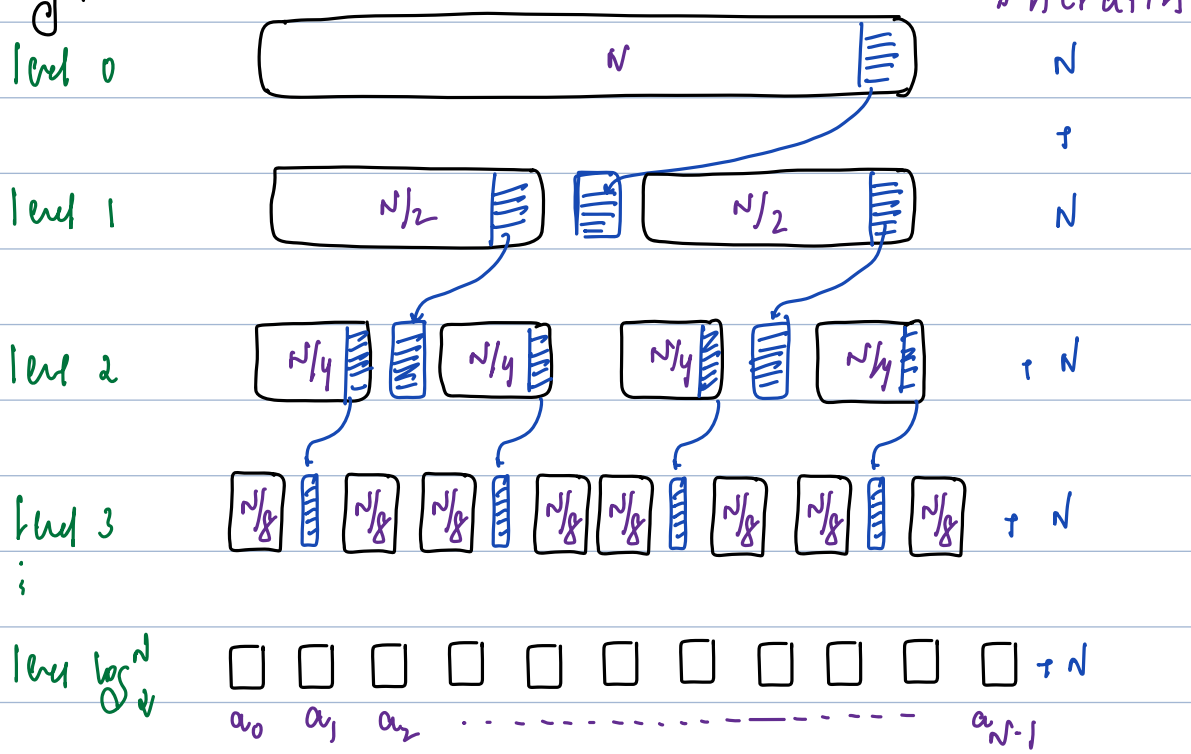
if i == n-1: { swap ar[i] & ar[j] }

```
int re-arrange(vector<int> &arr) { TC: O(N) SC: O(1)
    int N = arr.size();
    int n = arr[N-1], j = 0;
    for(int i = 0; i < N-1; i++) {
        if(arr[i] < n) { # arr[i] should come at left
            swap arr[i] & arr[j];
            j++;
        }
    }

    swap arr[N-1] & arr[j]; # Bringing last element to it's correct pos
}
```

Quick Sort Idea:



#Dry Run TC:



Note: for single element or empty elements we return, which will take O(1), Doing it N Time = O(N)

#Obs1: At every level we will have N iterating.

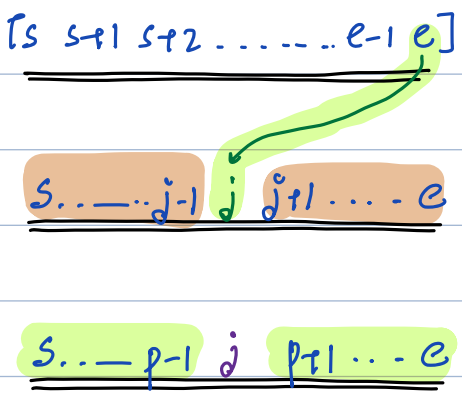
Total no. of levels = $\log_2^N$

Total TC = $O(N \log_2^N)$

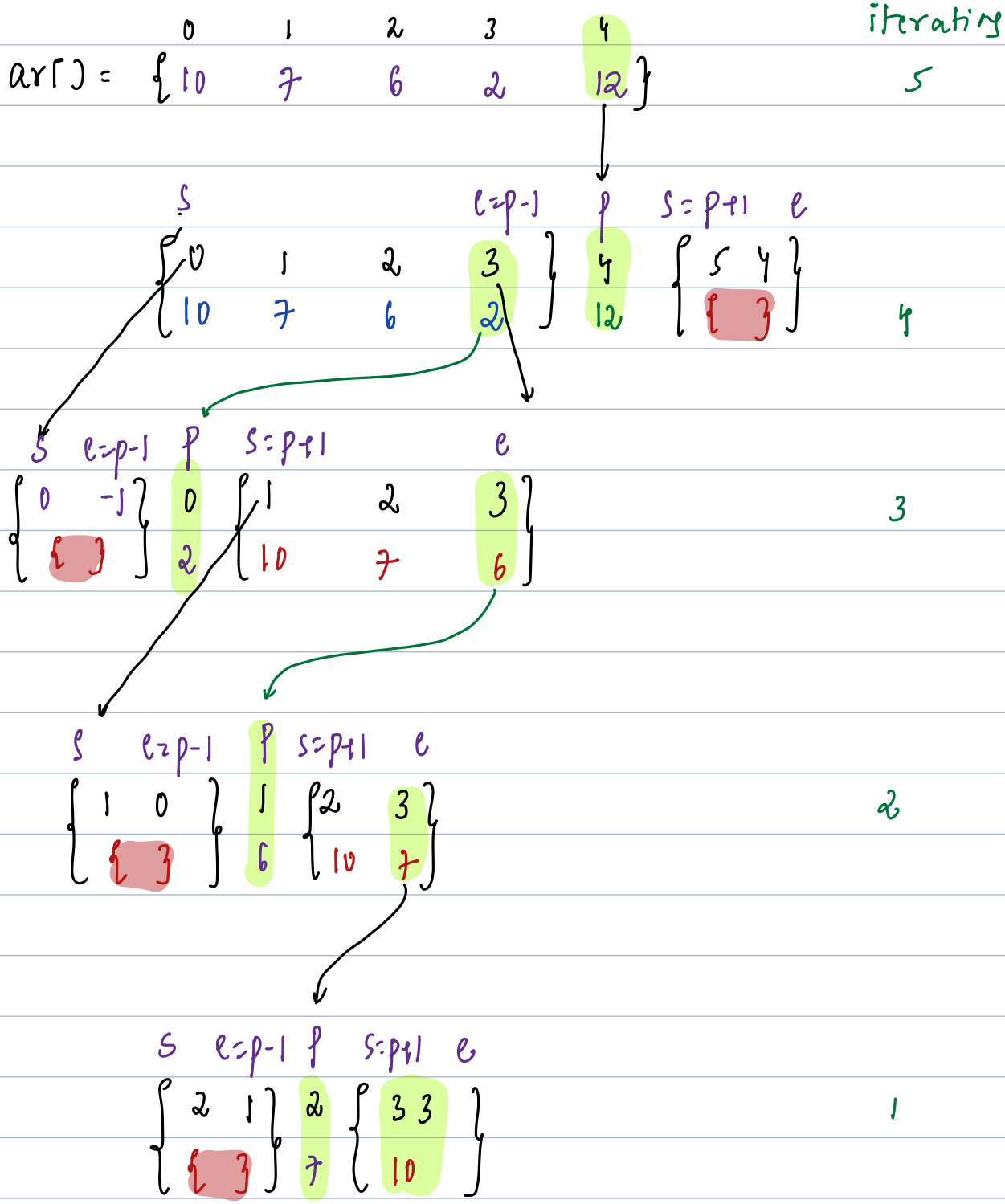
Ass: Given ar[], s, e : Sort ar[] from s..e & retn nothing.

	Best	Avg	Wrst
	O(NlogN)	O(NlogN)	O(N²)
sc:	O(log N)	O(log N)	O(N)

```
void Quicksort(vector<int>& aw, int s, int e){
    if(s >= e){retn;}

    #Step1: Bring last ele to it's correct p.
    int x = ar[e], j = s;
    for(int i = s; i < e; i++){
        if(ar[i] < x){
            swap ar[i] & ar[j];
            j++;
        }
    }

    swap ar[e] & ar[j];
    Quicksort(aw, s, j-1);
    Quicksort(aw, j+1, e);
}

vector<int> solve(vector<int> aw){
    int N = aw.size();
    Quicksort(aw, 0, N-1);
}
```



QuickSort: Worst Case:



#Con: In above example total iterations = 1+2+3+4+5 = 15

#Worst Case:

Total Iterations = $1+2+3+\dots+N-1+N = O(N^2)$

Probability that above case occurs is less, hence we consider

avg = $O(N \log N)$

#Note: If Time permits I more small questions.