Date: / /
Page:

Day - 85

## Quick Sort

⇉ It is a sorting algorithm.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 4 | 2 | 8 | 13 | 7 | 11 | 9 | 3 | (6) |

⇉ First we have to select a pivot element.
Pivot: Any element that is on their correct position in the sorted array.

⇉ For finding the correct position of the pivot, first we to find how many elements are equal or less than pivot element.

⇉ Suppose, in the above ex., we select 6 as pivot elem. then 4 elements are equal or less than 6.

⇉ [6 4 2 (3)] [6] [8 13 7 11 (9)]

[2] [3] [6 (4)] [6] [8 (7) 9 13 (11)]

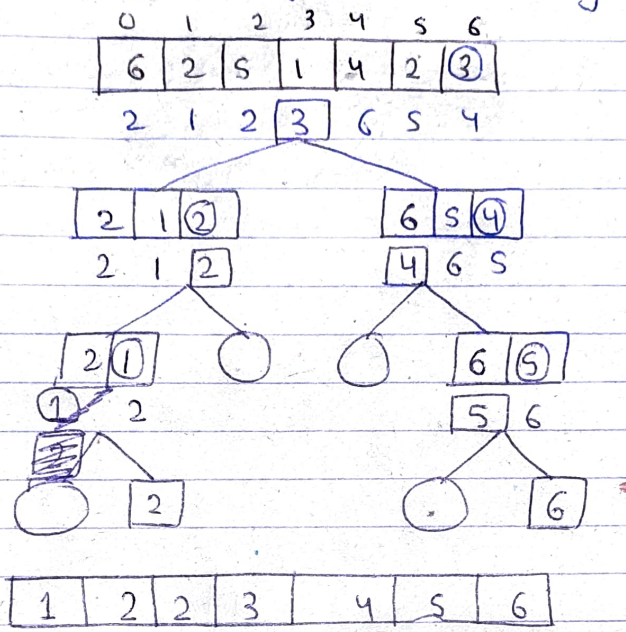
2 3 [4] [6] 6 [7] 8 9 [11] 13 } sorted array.

⇉ After putting the pivot element on the correct position, Then, you will have all the elements less than pivot on left & greater than pivot on right.

⇉ Now, again do this same thing with these two halves.

Date : / /
Page:

⇒ In the end, you will get sorted array.




⇒ The stopage condition is — empty or single element.

Code

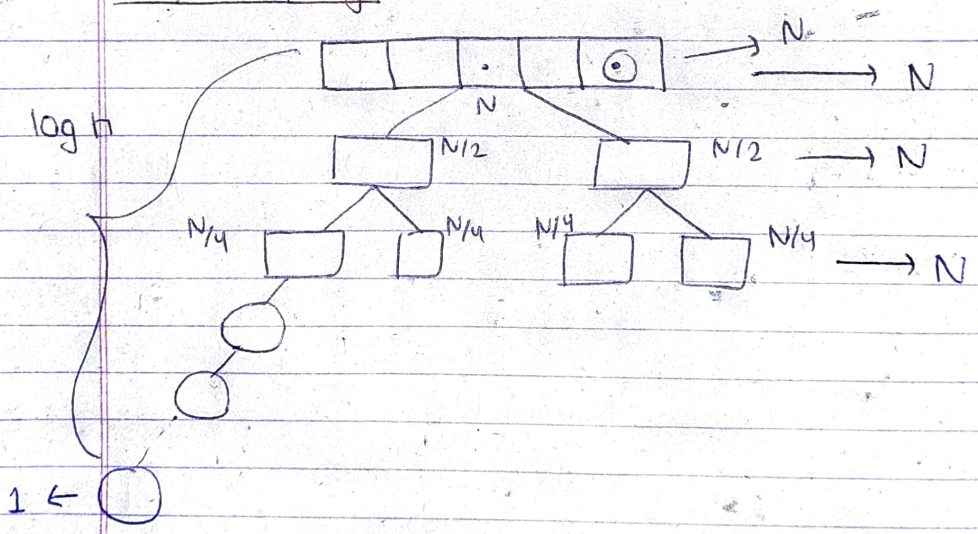
⇒
```
void quicksort(int arr[], int start, int end){

    if(start >= end){
        return;
    }
    int pivot = partition(arr, start, end);
    quick sort (arr, start, pivot -1);
    quicksort (arr, pivot+1, end);
}
```

Date: / /
Page:

```
int partition (int arr[], int start, int end){
    int pos = start;
    for(int i = start; i <= end; i++){
        if(arr[i] <= arr[end]){
            swap(arr[i], arr[pos]);
            pos++;
        }
    }
    return pos - 1;
}
```

* Time Complexity:




⇒ At every level,

$N + N + N + N + \dots + N$

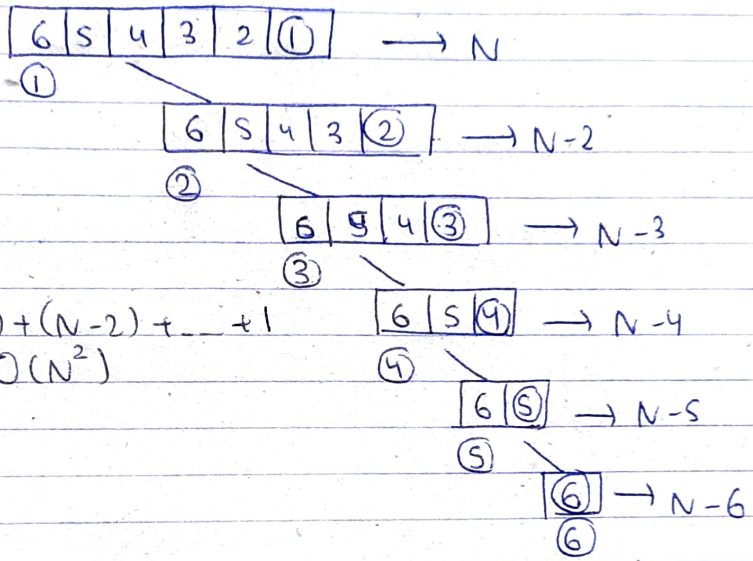
$N \log N$

⇒ But this is the avg. time complexity of Q.S..

⇒ S.C. → log N (in avg. case)

⇒ For this ex. → | 6 | 5 | 4 | 3 | 2 | 1 |

The cond. for Q.S. is worst case.




⇒ So,

⇒ $N+(N-1)+(N-2)+\dots+1$

⇒ $O(N^2)$

⇒ The worst case is when our array is in descending order & we have to sort it in ascending order.

⇒ S.C. → O(N).

⇒ Also, when our array is already sorted then the case is worst case.