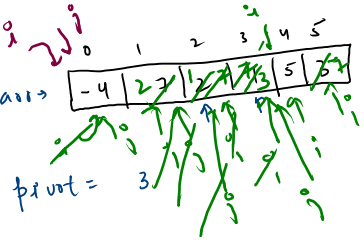
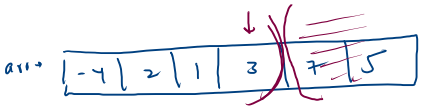
→ Two pointer

→ Partition an Array

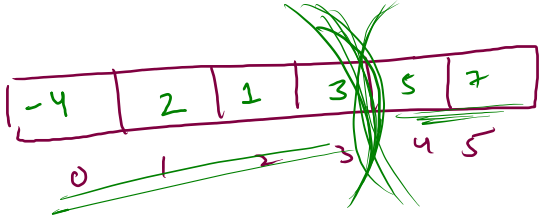


→ <= pivot
left side of array

> pivot
right side of array.



→ Maintain the order



i=0, j=0

```
if(arr[i] > pivot)
    ↳ i++;

else
    ↳ swap(i, j)
      i++;
      j++;
```

Sort 01 (Day 32)

Problem | Submissions | Leaderboard | Discussions

1. You are given an array(arr) containing only 0's and 1's.
2. You have to sort the given array in increasing order and in linear time.

Sample Input 0

```
5
0 1 0 1 0
```

Sample Output 0

```
0 0 0 1 1
```

n = 5

arr → | 0 | 1 | 0 | 1 | 0 |
0 1 2 3 4

arr → | 0 | 0 | 0 | 1 | 1 |
0 1 2 3 4

↳ linear time complexity

↳ ① Bubble sort
② Insertion sort
③ Selection sort
} → O(n²)

↳ ④ Merge sort } → O(n log n)

↳ ⑤ (a) count zero , count one

(b)

| 0 | 0 | 0 | 1 | 1 |
0 1 2 3 4

↳ sort

countZero = ~~1~~ 2 ~~1~~ 0
countOne = ~~1~~ 2 ~~1~~ 0

T.C → O(n)

2 iteration

↳ (a) Count no. of zeroes & ones

↳ (b) Filling zeroes & ones in the array

= = 0

1 iteration → i

↳ ⑥ Dutch National Flag Algo"

if (arr[i] = = 0) → ① swap(i,j) ② i++ ③ j++

else → ① → i++

while(i<n)

arr → | 0 | 0 | ~~1~~ | ~~0~~ | 1 | ~~0~~ | 1 |
0 1 2 3 4 5 6

i, j

i = 0, j = 0

if(arr[i] ≤ = pivot)

↳ swap(i,j)

↳ i++

↳ j++;

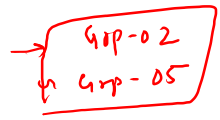
else i++;

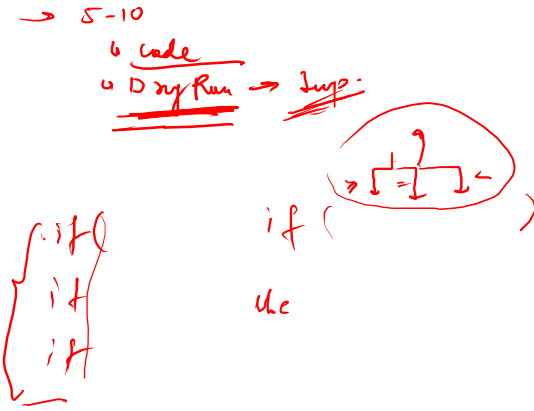
arr → | 0 | 0 | 0 | 1 | 1 |

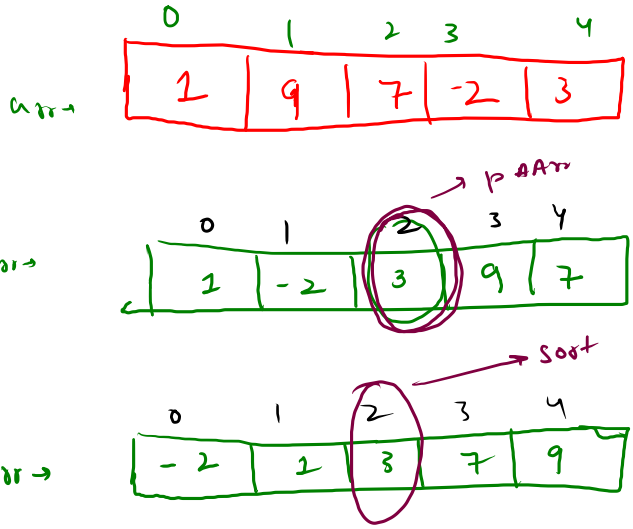
↳ T.C → O(n)

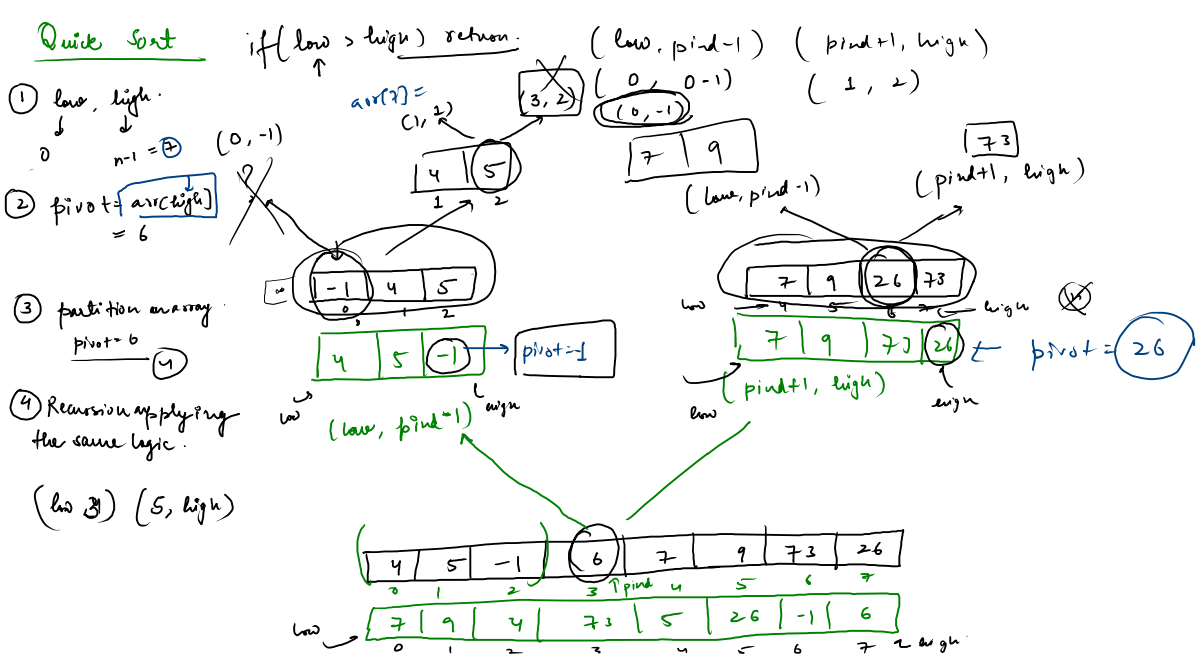
↳ 1 iteration

```
public class Solution {
    public static void swap(int arr[],int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void sort01(int arr[],int n){
        int i=0,j=0;

        while(i<n){
            if(arr[i] == 0){
                swap(arr,i,j);
                i++;
                j++;
            }else{
                // Value is 1
                i++;
            }
        }
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print outpu
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        sort01(arr,n);

        for(int i=0;i<n;i++){
            System.out.print(arr[i] + " ");
        }
    }
```






0 1 2 3 4
arr
1 9 7 -2 3
p = 3
partition an array
& p = 3 → exactly
at a same position as
it is in sorted
array.
p = 3
0 1 2 3 4
arr
1 -2 3 9 7
sort
0 1 2 3 4
arr
-2 1 3 7 9



1. low = 0, high = n-1
2. pivot = arr[high]
3. pind = partition an array
4. a) qs (arr, low, pind-1) → if(low > high) return
   b) qs (arr, pind+1, high)

Code + DryRun

10 min → 5 Coding + 5 DryRun

Imp.

```java
public class Solution {
    public static void swap(int arr[],int i, int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static int partitionAnArray(int arr[],int low, int high,int pivot){
        int i=low,j=low;
        while(i<=high){
            if(arr[i]<=pivot){
                swap(arr,i,j);
                i++;
                j++;
            }else{
                i++;
            }
        }

        return j-1;
    }

    public static void quickSort(int arr[], int low, int high){
        if(low > high) return;

        int pivot = arr[high];
        int pind = partitionAnArray(arr,low,high,pivot);

        quickSort(arr,low,pind-1);
        quickSort(arr,pind+1,high);
    }
```

Imp.

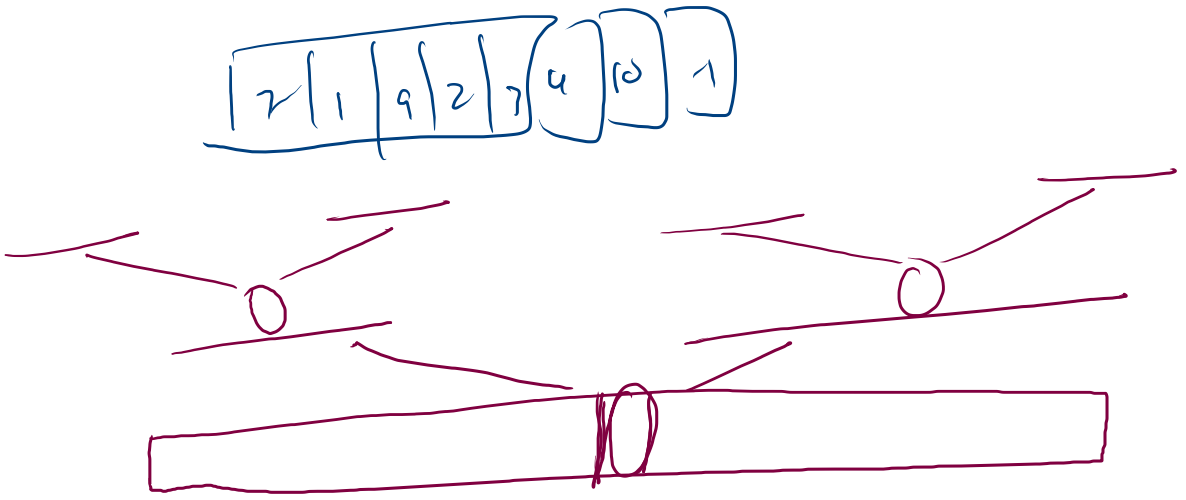
```java
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDI
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        quickSort(arr,0,n-1);

        for(int i=0;i<n;i++){
            System.out.print(arr[i] + " ");
        }
    }
```

T.C = ?

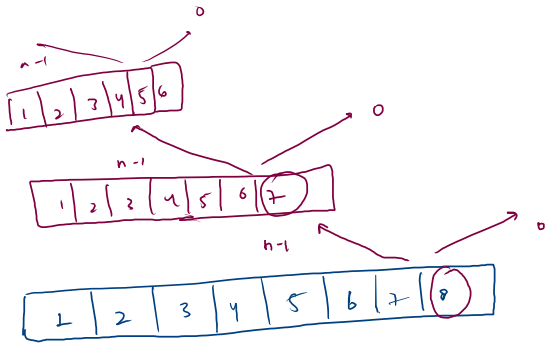
$T(n) = T(n/2) + T(n/2) + n^{\text{partition of an array}} + K$

$$T(n) = 2T(n/2) + n + K$$

↳ $T.C = O(n \log n)$ → when elements are arranged in random order

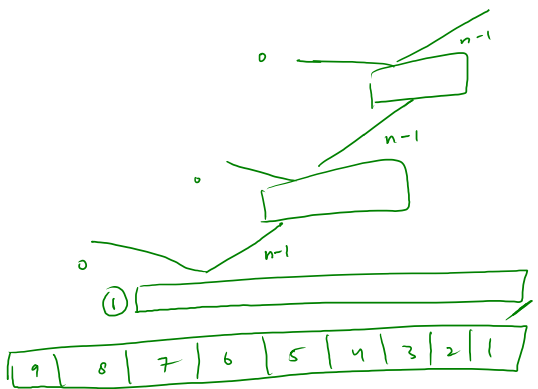


Sorted → Increasing order ⇒ $T(n) = T(n-1) + n + K$



(2) sorted in decreasing order

$T(n) = T(n-1) + n + K$



Sorted in Increasing | Decreasing order

partition

$T(n) = T(n-1) + n + k$

↳ $O(n^2)$

Quick sort

partition → $O(n)$

T.C.

↳ (1) When elements are arranged in random order

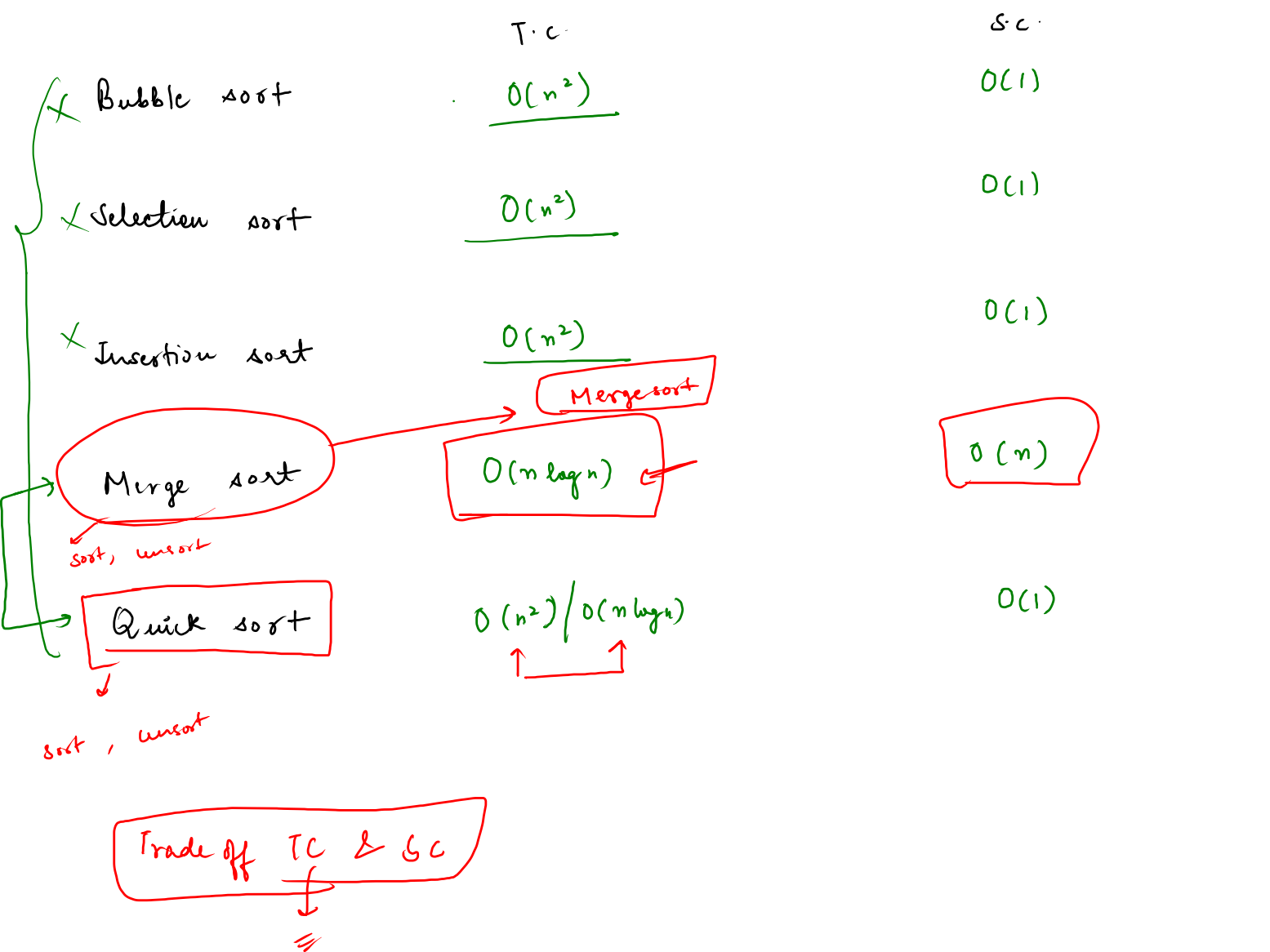
$T.C. = O(n \log n)$

↳ (2) When elements are arranged in increasing | decreasing order.

$T.C. = O(n^2)$

S.C.

$O(1)$ → no extra space


T.C
S.C
Bubble sort
O(n²)
O(1)
Selection sort
O(n²)
O(1)
Insertion sort
O(n²)
O(1)
Merge sort
Merge sort
O(n log n)
O(n)
sort, unsort
Quick sort
O(n²) / O(n log n)
O(1)
sort, unsort
Trade off TC & SC