# SORTING

# SELECTION SORT →


Step 1 → 9 46 24 52 20 13

Step 2 → 9 13 24 52 20 46

Step 3 → 9 13 20 52 24 46

Step 4 → 9 13 20 24 52 46

Step 5 → 9 13 20 24 46 52

Steps →

1) Swap at index 0 & min index $\{0 - (n-1)\}$

2) Swap at index 0 & min index $\{1 - (n-1)\}$

⋮

TC : $O(n^2)$
SC : $O(1)$

Code →

```
for (i=0 ; i<=n-2 ; i++){
    mini = i
    for (j=i ; j<=n-1 ; j++){
        if(arr[j] < arr[mini]) mini = j;
    }
    swap(arr[i], arr[mini]);
}
```

# Bubble Sort →

push the max to the last by adjacent swapping

| 13 | 46 | 24 | 52 | 20 | 9 |
|---|---|---|---|---|---|

13 46 24 52 20 9

13 24 46 52 20 9

13 24 46 52 20 9

13 24 46 20 52 9

13 24 46 20 9 52

1st iteration

13 24

13 24 46

13 24 20 46

13 24 20 9 46 (52)

2nd iteration

13 24

13 20 24

13 20 9 24 (46) (52)

3rd iteration

13 20

13 9 20 (24) (46) (52)

4th iteration

9 13 (20) (24) (46) (52)

5th iteration

0 – (n-1)

0 – (n-2)

⋮

0 – 1

O(n) → Best

TC: O(n²) → Worst

SC: O(1)

```
for (i = n-1 ; i >= 1 ; i--) {
    swap = 0;
    for (j = 0 ; j <= i-1 ; j++) {
        if (a[j] > a[j+1]) {
            swap (a[j] , a[j+1]);
            swap = 1;
        }
        if (swap == 0) break;
    }
}
```

# Insertion Sort →

Take an element & place it in correct order.

| 14 | 9 | 15 | 12 | 6 | 8 | 13 |
|---|---|---|---|---|---|---|

9 14

9 14 15

9 12 14 15

6 9 12 14 15

6 8 9 12 14 15

6 8 9 12 13 14 15

$O(n) \rightarrow$ best

TC: $O(n^2) \rightarrow$ worst

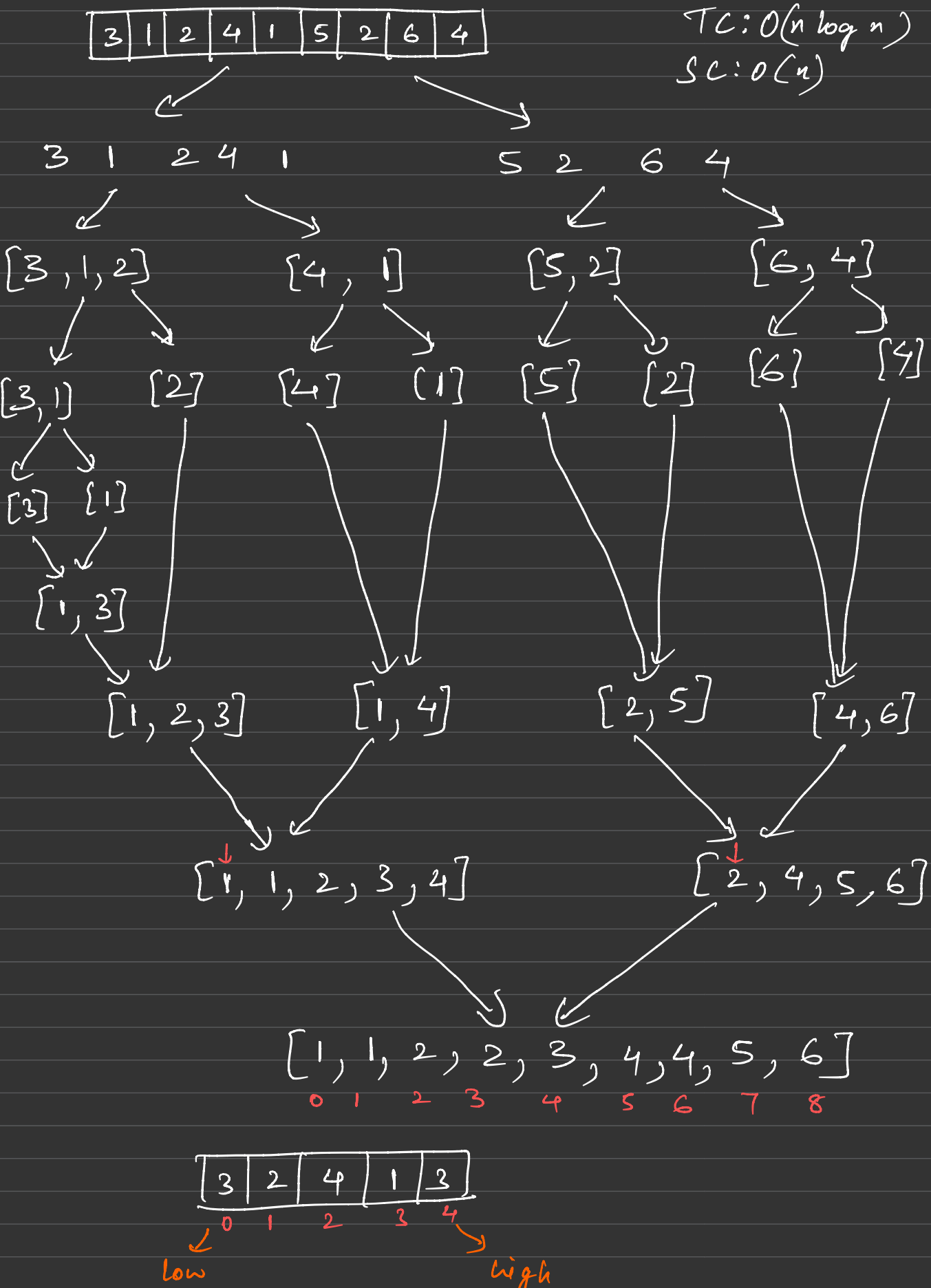
SC: $O(1)$

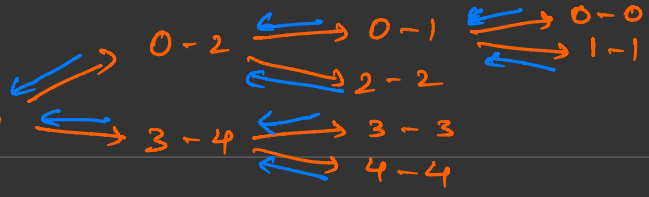
```
for (i=0; i<=n-1; i++){
    j=i
    while (j>0 && a[j-1] > a[j]){
        swap(a[j-1], a[j]);
        j--;
    }
}
```

# Merge Sort →

Divide & Conquer

TC: O(n log n)
SC: O(n)






Pseudocode →

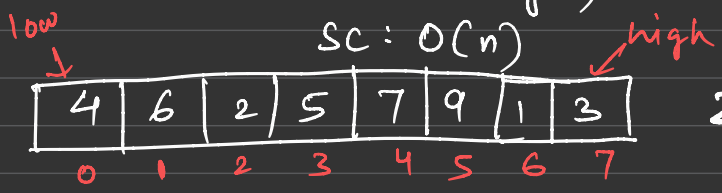
```
merge (arr, low, mid, high) {
    vector <int> temp;
    left = low;
    right = mid+1;
    while (left <= mid && right <= high) {
        if (arr [left] <= arr [right])
            temp.add (arr [left++]);
        else
            temp.add (arr [right++]);
    }
    while (left <= mid)
        temp.add (arr [left++]);
    while (right <= high)
        temp.add (arr [right++]);
    for (i = low → high)
        arr [i] = temp [i - low];
}
mergeSort (arr, low, high) {
    if (low >= high) return;
    mid = (low + high) / 2
    mergeSort (arr, low, mid);
    mergeSort (arr, mid+1, high);
    merge (arr, low, mid, high);
}
```

# Quick Sort →

TC: O(n log n)

SC: O(n)

1. Pick a pivot & place it in its correct place in the sorted array
2. Smaller on the left, larger on the right.

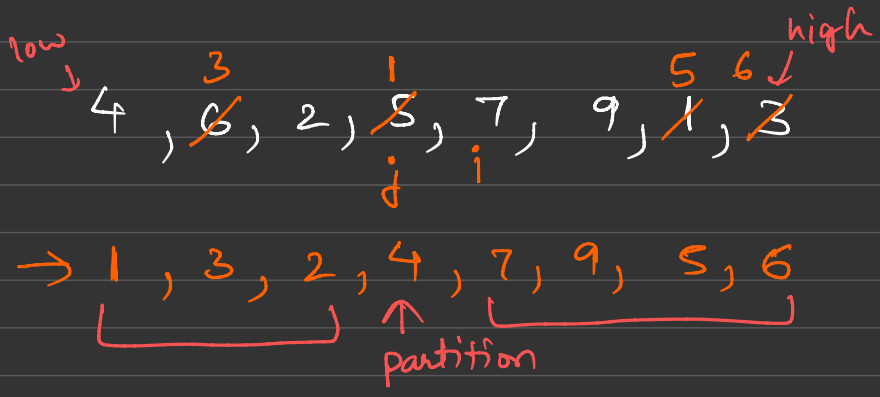



2 1 3 [4] 6 5 7 9

1 [2] 3 4 [5] 6 [7] 9

1 2 3 4 5 6 7 9

pivot = a[low]




Pseudo code →

```
int f(arr, low, high){
    pivot = arr[low];
    i = low;
    j = high;
    while(i<j){
        while(arr[i] <= pivot && i<high)
            i++;
        while(arr[j] > pivot && j>low)
            j--;
        if(i<j)  swap(arr[i], arr[j]);
    }
    swap(arr[i], arr[low]);
    return j;
}
```

```
QuickSort (arr, low, high){
    if (low < high){
        pInd = f (arr, low, high);
        QuickSort ( arr, low, pInd - 1);
        QuickSort (arr, pInd+1, high);
    }
}
```