Lec-8
01/09/25 2:15 PM

Sorting

bubble Sort

bubble
Selection
Insertion
merge
quick
Heap
Counting
Cycle Sort
DNF

```
public static void Sort(int[] arr) {
    // TODO Auto-generated method stub
    for (int turn = 1; turn < arr.length; turn++) {
        for (int i = 0; i < arr.length - turn; i++) {
            if(arr[i]>arr[i+1]) {
                int temp=arr[i];
                arr[i]=arr[i+1];
                arr[i+1]=temp;
            }
        }
    }
}
```

```
public static void Sort(int[] arr) {
    // TODO Auto-generated method stub
    for (int turn = 1; turn < arr.length; turn++) {
        for (int i = 0; i < arr.length - turn; i++) {
            if (arr[i] > arr[i + 1]) {
                int temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
        }
    }
}
```

```
int[] arr = { 4, -1, 5, 3, 1, 2 };
```

```
public static void Sort(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        int idx = Minimum_Index(arr, i);
        int temp = arr[idx];
        arr[idx] = arr[i];
        arr[i] = temp;
    }
}
```

```
public static void Insert_Element(int[] arr, int i) {// i-> last index
    int item = arr[i];
    int j = i - 1;
    while (j >= 0 && arr[j] > item) {
        arr[j + 1] = arr[j];
        arr[j] = item;
        j--;
    }
    // j+1 correct index;
```

```
public static int Maximum_Sum(int[] arr) {
    int ans = Integer.MIN_VALUE;
    int sum = 0;
    for (int i = 0; i < arr.length; i++) {
        sum += arr[i];
        ans = Math.max(ans, sum);
        if (sum < 0) {
            sum = 0;
        }
    }
}
```

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.