Table of Contents

Sorting

1. Definition

- Arranging in ascending or descending order according to parameter/scenario/condition.

2. Inbuilt library

- Array.sort() ➔ syntax to sort an array
- TC: O(nlogn), n = no of elements in an array

3. Noble Integer

- No of elements < element == element itself

a. Data is unique

Ex1: { -1 -5 3 5 -10 4 }

#less: { 2 1 3 5 0 4 }

Explanation:

- No's less than -1 are 2
- No's less than -5 are 1
- No's less than 3 are 3
- No's less than 5 are 5
- No's less than -10 are 0
- No's less than 4 are 4
- If we observe that no 3 has 3 no's less than itself , 4 has 4 no's less than itself and 5 has 5 no's less thanitself.
- So, from definition we can say that 3 and 4 are noble elements

Ex2: { -3 0 2 5 }

Index:{ 0 1 2 3 }

#less: { 0 1 2 3 }

Explanation:

- No's less than -3 are 0
- No's less than 0 are 1
- No's less than 2 are 2
- No's less than 5 are 3
- If we observe that no 2 has 2 no's less than itself.
- So, from definition we can say that 2 is a noble elements

Observation

- From the above example we can say that -ve numbers are not noble numbers.
- The less number is basically the index(0'th order) of number so we can say that if a[i] == i that no is noble number if the array is in ascending order.

Code

```
int noble(int[] arr){
        int n = arr.length;
        Array.sort(arr);
        for(int i=0; i<n; i++){
                if(a[i] == i){
                        System.out.print(a[i]+ " is noble element");
                }
        }
}
```

b. Data can be repeated

Ex1: { 0 2 2 3 3 6 }

Index:{ 0 1 2 3 4 5 }

#less: { 0 1 1 3 3 4 }

Explanation:

- No's less than 0 are 0
- No's less than 2 are 1
- No's less than 2 are 1 because previous no is same so less than value is same as previous no.
- No's less than 3 are 3
- No's less than 3 are 3 because previous no is same so less than value is same as previous no.
- No's less than 6 are 4
- If we observe that both no 3 has 3 no's less than itself and 0 has 0 no's less than itself.
- So, from definition we can say that both 3's and 0 are noble elements so total noble no's are 3.

Ex2: { -10 1 1 1 4 4 4 7 10 }

Index:{ 0 1 2 3 4 5 6 7 8 }

#less: { 0 1 1 1 4 4 4 7 8}

Explanation:

- No's less than -10 are 0
- No's less than 1 are 1
- No's less than 1 are 1 because previous no is same so less than value is same as previous no.
- No's less than 1 are 1 because previous no is same so less than value is same as previous no.
- No's less than 4 are 4

- No’s less than 4 are 4 because previous no is same so less than value is same as previous no.
- No’s less than 4 are 4 because previous no is same so less than value is same as previous no.
- No’s less than 7 are 7
- No’s less than 10 are 8
- If we observe that all no 1 has 1 no’s less than itself and 4 has 4 no’s less than itself and 7 has 7 no’s less than it self.
- So, from definition we can say that all 1’s, 4’s and 7 are noble elements so total noble no’s are 7

Observation

- If elements are coming for first time
  ```
  If(a[i]!=a[i-1])
  #less count = i
  ```
- If elements are repeated
  ```
  #count will be same
  ```

Code

```
int noble(int[] arr){
        int n = arr.length;
        Array.sort(arr);
        int ans = 0;
        if(arr[0] == 0){
                ans++;
        }
        int c = 0;
        for(int i=1; i<n; i++){
                if(a[i] != a[i-1]){
                        c = i;
                }
                if(a[i] == c){
                        ans++;
                }
        }
        return ans;
}
```

4. Comparator

a. Ascending order

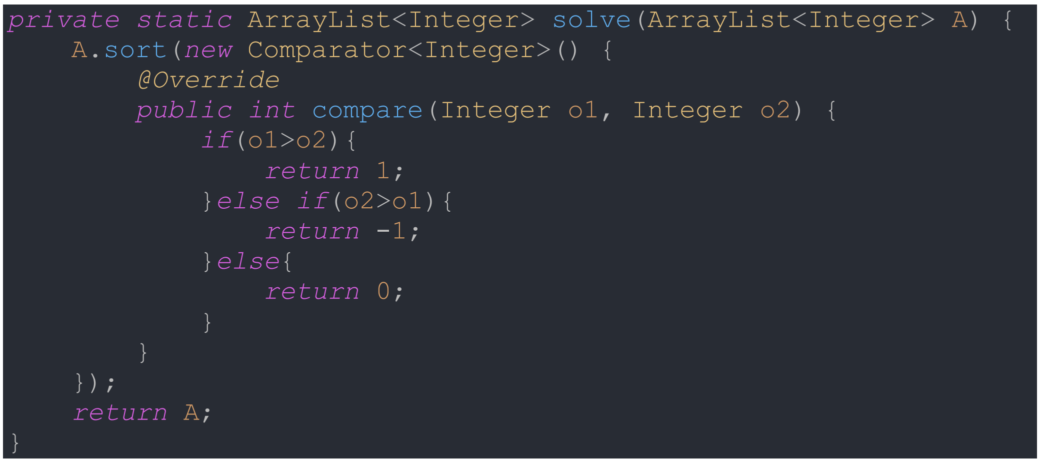
```
compare(a,b){
        if(a>b){
                return 1;
        }else if(a<b){
                return -1;
        }
        return 0;
}
```

b. Descending order

```
compare(a,b){
        if(a>b){
                return -1;
        }else if(a<b){
                return 1;
        }
        return 0;
}
```

Example:

Ascending order



```
private static ArrayList<Integer> solve(ArrayList<Integer> A) {
    A.sort(new Comparator<Integer>() {
        @Override
        public int compare(Integer o1, Integer o2) {
            if(o1>o2){
                return 1;
            }else if(o2>o1){
                return -1;
            }else{
                return 0;
            }
        }
    });
    return A;
}
```