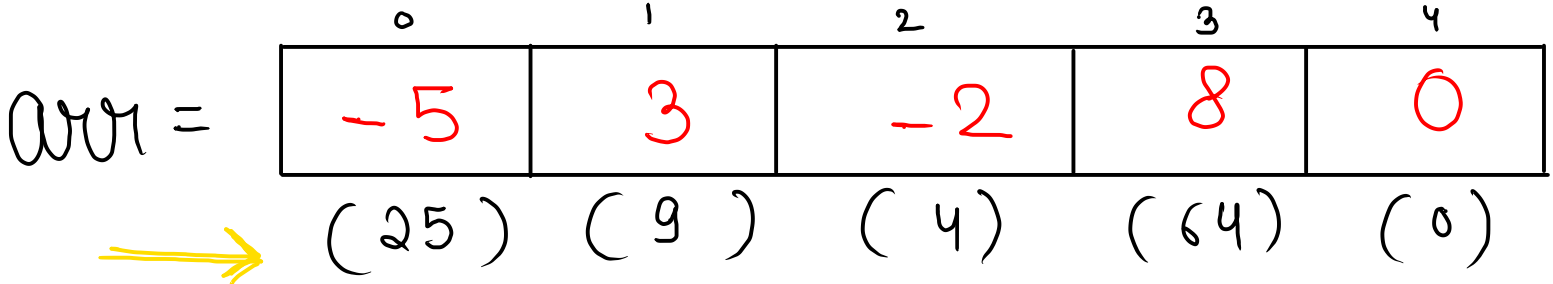
⇒ Custom sort (used to alter the properties of inbuilt function)

Arrays.sort(arr); → O(nlog(n))



ans ⇒ [0 , -2 , 3 , -5 , 8]

Note:- Custom sorting will never effect time complexity

→ Comparable and Comparator

syntax

```
Arrays.sort(arr, new myComparator());
```

→ implementation

annotation

```
public static class myComparator implements Comparator<Integer>{
    @Override
    public int compare(Integer a, Integer b){
        return a-b;
    }
}
```

← logic based on which elements of array will be rearranged

meaning cmp

return a - b ; // arrange the elements in ↑ing order

return b - a ; // arrange the elements in ↓ing order

Note:- if we use a value first then ↑ing
" " " b " " " ↓ing

Code

```
public static void main(String[] args) {
    Integer[] arr = {5, -4, 0, -1, 3};
    Arrays.sort(arr, new myComparator());
    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a - b; // increasing order
        // return b - a; // decreasing order
    }
}
```

Sort the array according to their Square of each element

arr = [4 , -1 , 0 , -5 , 6]

(16 , 1 , 0 , 25 , 36)

ans = [0 , -1 , 4 , -5 , 6]

Code

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

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    Integer[] arr = new Integer[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    Arrays.sort(arr, new myComparator());

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

public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a*a - b*b;
    }
}
```

⇒ Lambda function

Syntax :-

```
Arrays.sort (arr, (a, b) -> {
        return b - a;
});
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    Integer[] arr = new Integer[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    Arrays.sort(arr, (a, b) -> {
        return a*a - b*b;
    });

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

gmp

Ques

- ↑ing order :- $a - b$
- ↓ing order :- $b - a$
- ↑ing square :- $a^2 - b^2$
- ↓ing square :- $b*b - a*a$
- ↓ing cube :- $b*b*b - a*a*a$
- ↑ing cube :- $a*a*a - b*b*b$

Imp arr = [5 , 2 , 1 , -2 , 5 , -7]

```
Arrays.sort(arr, (a,b)->{
    return a-b;
});
```

a = myself

b = other

Note:- 1) whenever we return a -ve value then smaller values should appear first

2) whenever we return a +ve value then larger values should appear first

arr = [5 , 2 , 1 , -2 , 5 , -7]

M. gmp

return a-b :- [-7 , 5]

return b-a :- [5 , -7]

return -1 :- [5 , -7]

return +1 :- [-7 , 5]

V. Imp Note :-

return a-b :- arrange in ↑ing order based on values

return b-a :- arrange in ↓ing order based on values

return -1 :- a value will appear first

return +1 :- b value will appear first

Sort Array By Parity

↳ sort elements so that
  ↳ all even values appear first, all odd values appear last
  ↳ all even values and all odd values should be in ↑ing order

arr =

3	4	2	8	7	1	5
0	1	2	3	4	5	6

step I

4	2	8	3	7	1	5
0	1	2	3	4	5	6

Step II

2	4	8	1	3	5	7
0	1	2	3	4	5	6

diff

increasing :- 1, 2, 3, 4, 7, 9, 10, ----

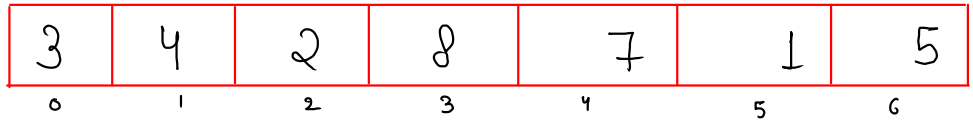
non-decreasing :- 1 2 2 2 3 3 4 5 5....

---

decreasing :- 5, 4, 3, 1, 0, -2, -4, ---

non-increasing :- 5, 4, 4, 4, 4, 1, 0, -2, -2, -4, ---

Logic



a = 4
b = 2

a	b	return
even	even	a − b
odd	odd	a − b
even	odd	−1
odd	even	1

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    Integer[] arr = new Integer[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    Arrays.sort( arr, (a, b) -> {
        if ( a % 2 == 0 && b % 2 == 0 ) {
            return a - b;
        } else if ( a % 2 != 0 && b % 2 != 0 ) {
            return a - b;
        } else if ( a % 2 == 0 && b % 2 != 0 ) {
            return -1;
        } else {
            return 1;
        }
    } );

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

T.C = O(n log(n))
where n is size of array

Qu) arrange the element where all odd values should appear first followed by even values & odd values are arranged in ↑ing order based on square values & even values are arranged in ↓ing order based on cube values: