Sort Array By Parity

arr =

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

observation

↳ all even values will appear first and then all odd values will appear later

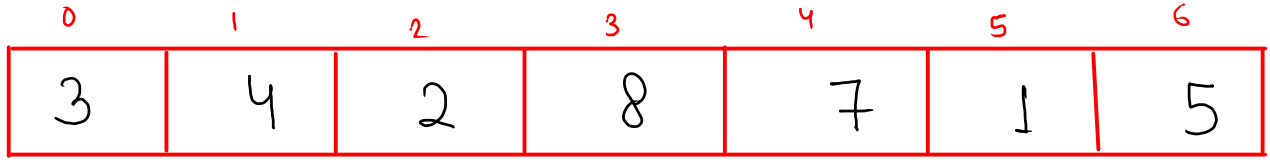
↳ in non-decreasing order (↑ing order)

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

strictly increasing :- 1 2 3 5 7 9 12 ....

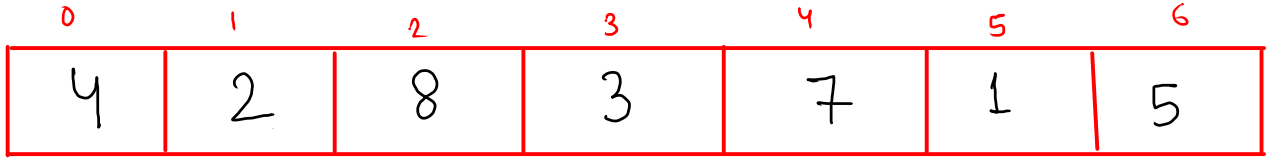
Explanation

arr =



a

arr =

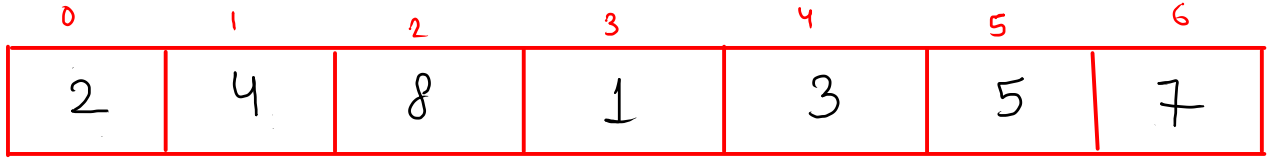


step I

even

odd

arr =



step II

✓✓

Logic

a	b	
even	odd	return -1
odd	even	return +1
even	even	return a-b
odd	odd	return a-b

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();
    }
    //logic
    Arrays.sort(arr, (a, b) -> {

        if ( a % 2 == 0 && b % 2 == 0 ) { // a = even, b = even
            return a - b;
        } else if ( a % 2 != 0 && b % 2 != 0 ) {  // a = odd, b = odd
            return a - b;
        } else if ( a % 2 == 0 && b % 2 != 0 ) { // a = even, b = odd
            return -1;
        } else if ( a % 2 != 0 && b % 2 == 0 ) { // a = odd, b = even
            return 1;
        }
        return 0;
    });

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

Code

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

Qu) Arrange, odd values should be first & then even values. also, odd values should be sorted in ↑ing order based on square values & even values should be in ↑ing order based on cube values.

a → even b → even = return b*b*b − a*a*a

a → odd b → odd = return a*a − b*b

a → even , b → odd = return +1

a → odd b → even = return −1

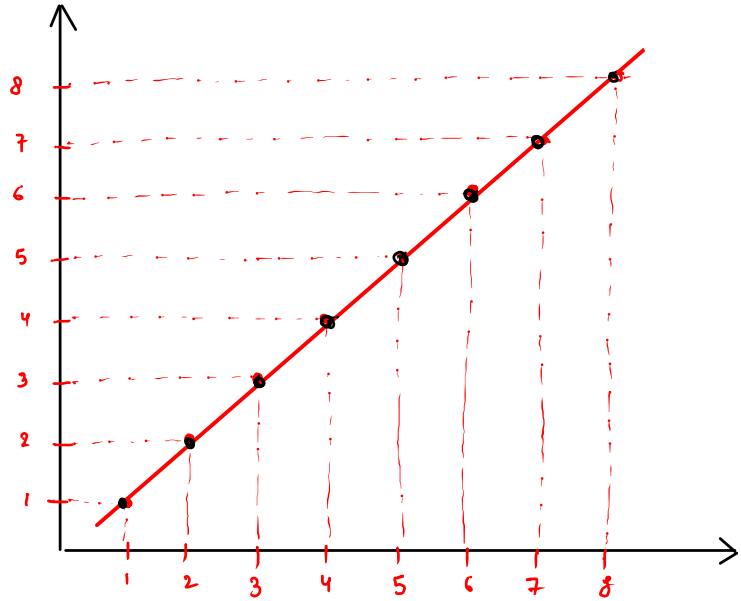
Sort an array in wave form 1

```
arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= …..
```

arr = [8 (0), 3 (1), 5 (2), 2 (3), 1 (4), 4 (5), 5 (6), 6 (7)]

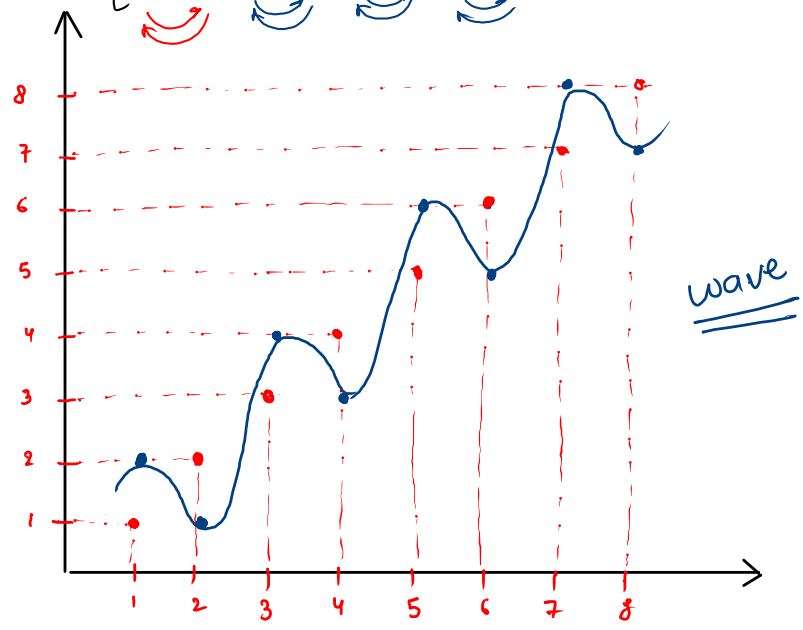
Ex:- 5 >= 3 <= 4 >= 2 <= 8 >= 1 <= 6 >= 5

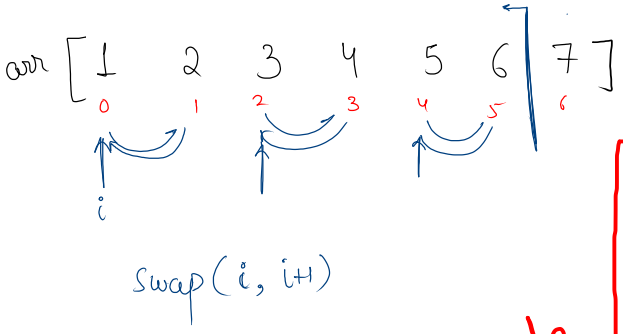
1) sort array



2) swap adjacent elements

[1 2 3 4 5 6 7 8]





Code

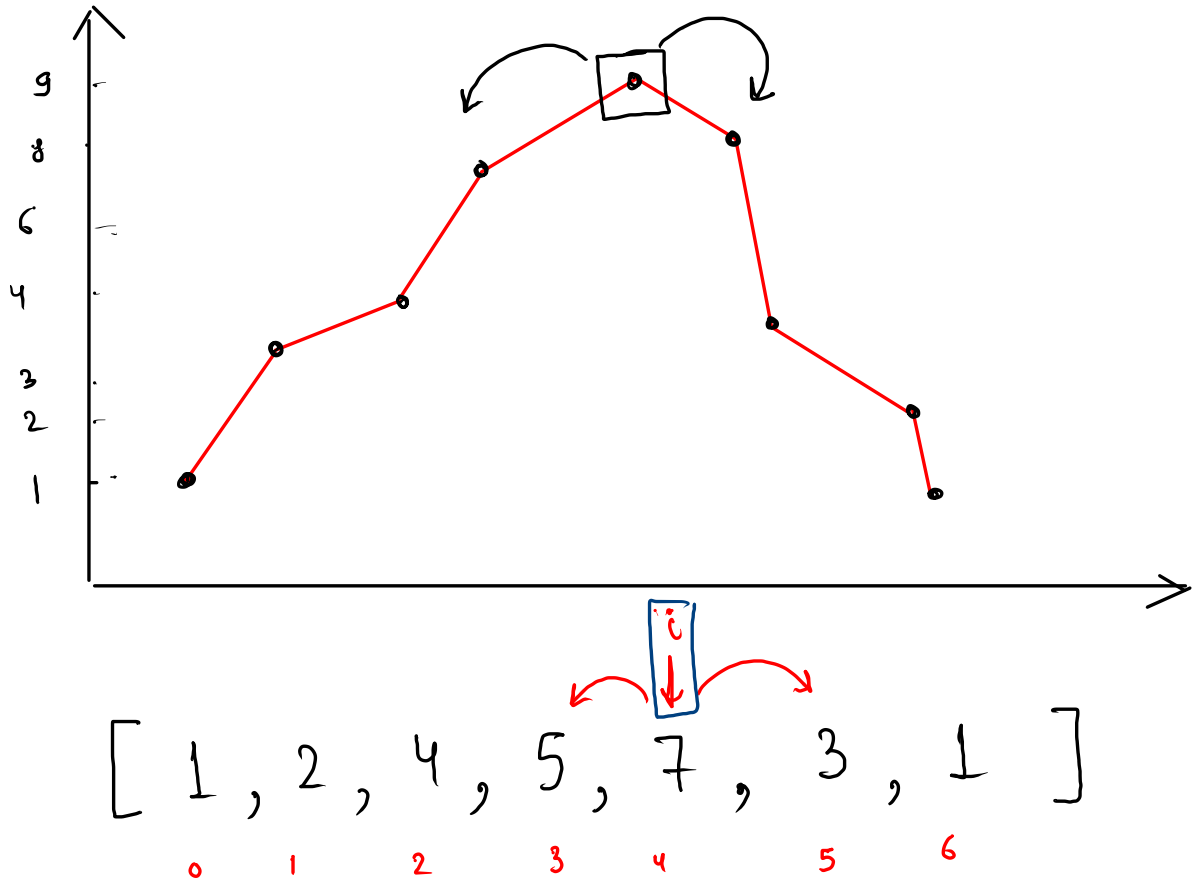
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    createWave(arr, n);

    //print
    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
public static void createWave(int[] arr, int n) {
    //logic
    //step1
    Arrays.sort(arr);

    //step2
    for (int i = 0; i < n - 1; i += 2) {
        swap(arr, i, i + 1);
    }
}
public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

$T.C = \frac{n}{2} + n\log(n)$

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

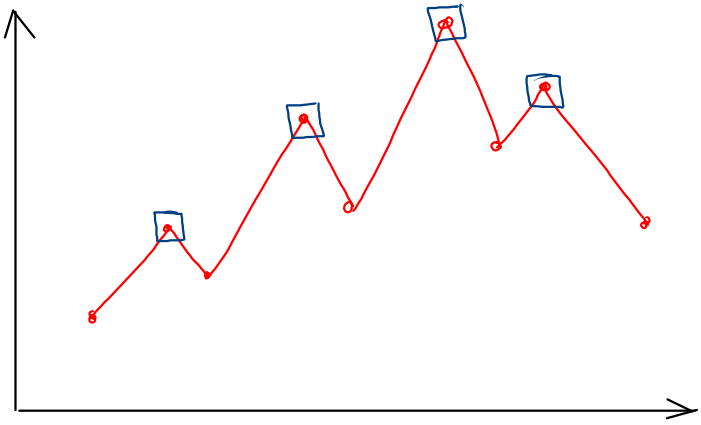
$T.C = O(n\log(n))$

Peak Index in a Mountain Array 2



```java
public static void main(String[] args) {
    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();
    }

    System.out.println(peakIndex(arr, n));
}

public static int peakIndex(int[] arr, int n) {
    for (int i = 1; i < n - 1; i++) {
        int curr = arr[i];
        int left = arr[i - 1];
        int right = arr[i + 1];
        if ( curr > left && curr > right ) {
            return i;
        }
    }
    return -1;
}
```

Peak Elements



Code

```java
public static void main(String[] args) {
    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();
    }

    peakIndex(arr, n);
}

public static void peakIndex(int[] arr, int n) {
    for (int i = 1; i < n - 1; i++) {
        int curr = arr[i];
        int left = arr[i - 1];
        int right = arr[i + 1];
        if ( curr > left && curr > right ) {
            System.out.print(arr[i] + " ");
        }
    }
}
```