Sort an array in wave form 1

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

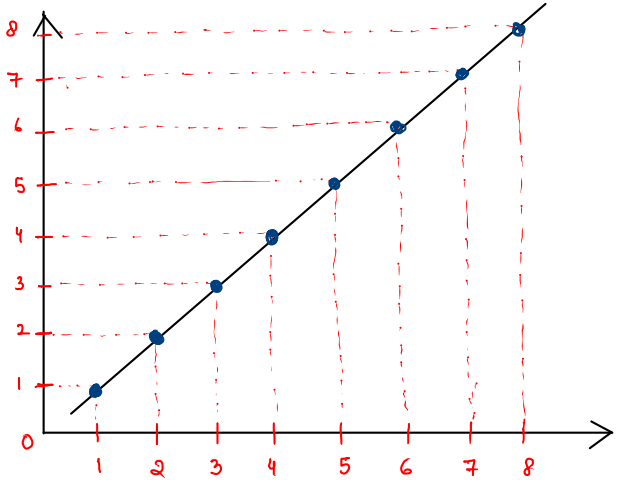
n= 7

arr

10	90	49	2	1	5	23

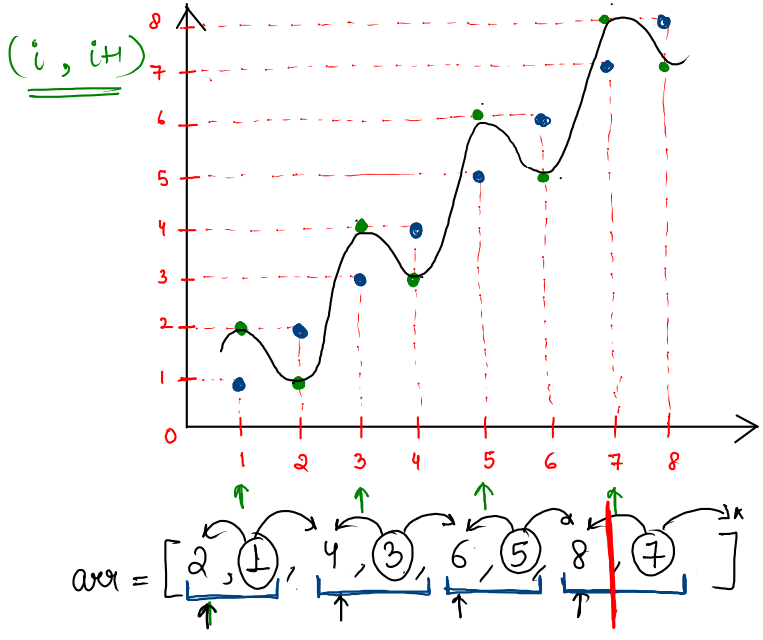
arr = [8 , 3 , 5 , 2 , 1 , 4 , 5 , 6]

1) Sort the array

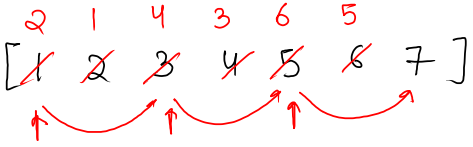


arr = [1, 2, 3, 4, 5, 6, 7, 8]

2) swap every adjacent pair



Note



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();
    }

    int[] ans = sortWave(arr);

    // printing
    for (int i = 0; i < n; i++) {
        System.out.print(ans[i] + " ");
    }
}
public static int[] sortWave(int[] arr) {
    int n = arr.length;
    // step 1
    Arrays.sort(arr);

    // step 2
    for (int i = 0; i < n - 1; i += 2) {
        swap(arr, i, i + 1);
    }

    return arr;
}
public static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

logic

O(n log(n))

O(n)

T.C ≃ O(n log n)

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

S.C = O(1)

Minimum difference 7

n = 4

arr =

0	1	2	3
9	4	1	7

k = 3

✗ 9, 4, 1 = 8

9, 4, 7 = 5

✗ 9, 1, 7 = 8

4, 1, 7 = 6

9 4 = 5

9 1 = 8

9 7 = 2

4 1 = 3

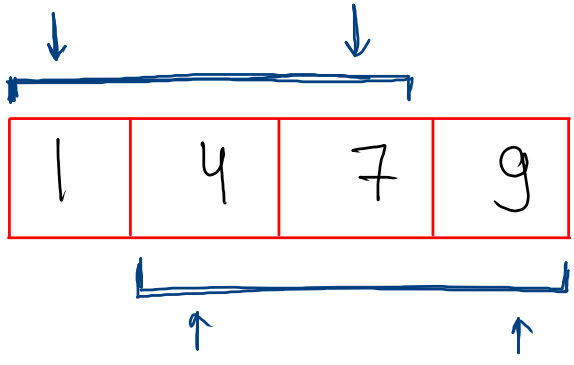
4 7 = 3

1 7 = 6

---

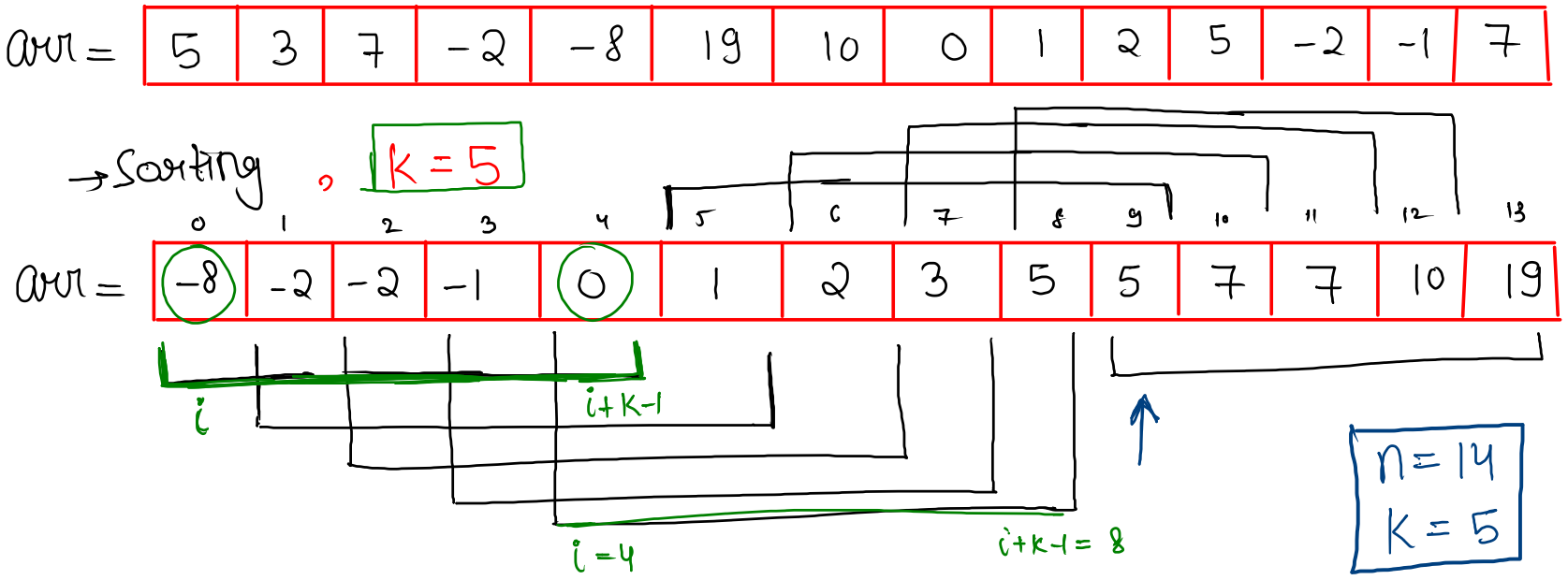
sorting

arr =



k = 3

ans = ~~6~~ 5



ans = ~~8~~ 3

loop

```
for( int i=0; i<= n-K ; i++) {
    first = arr[i];
    last = arr[i+k-1];

}
```

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();
    }
    int k = scn.nextInt();

    int ans = miniDiff(arr, n, k);
    System.out.println(ans);
}
public static int miniDiff(int[] arr, int n, int k) {
    Arrays.sort(arr);
    int ans = Integer.MAX_VALUE;
    for (int i = 0; i <= n - k; i++) {
        int smallest = arr[i];
        int largest = arr[i + k - 1];
        int diff = largest - smallest;
        if ( diff < ans ) {
            ans = diff;
        }
    }
    return ans;
}
```

T.C = O(nlogn + n)

≅ O(nlogn)

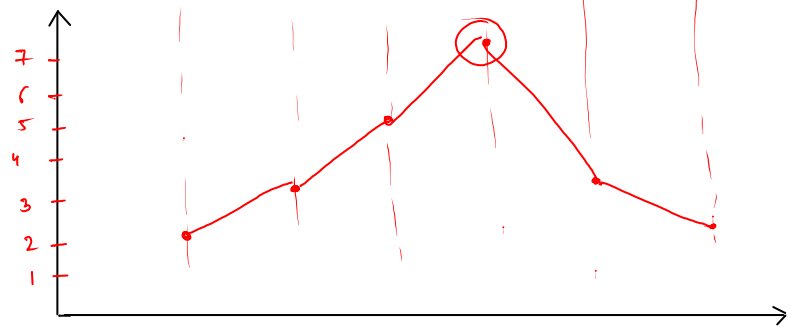
where 'n' is size of array

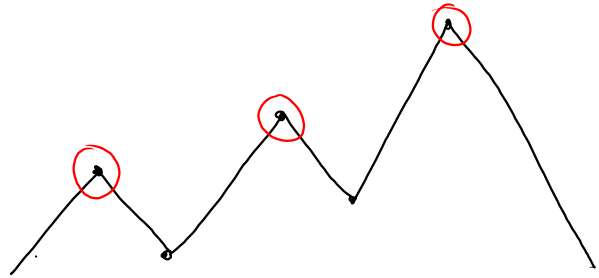
S.C = O(1)

Peak Elements

(element which is greater then both its nbr)

arr = [2 3 5 7 3 2]



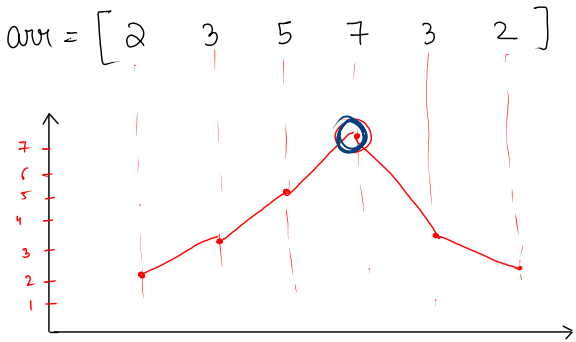


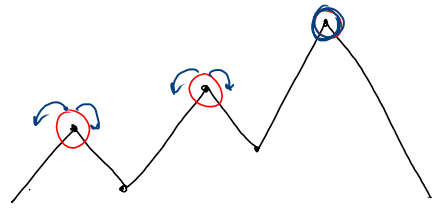
code

```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();
    }

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

Peak Index in a Mountain Array 2





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();
    }

    int ans = findAllPeakElements(arr, n);
    System.out.println(ans);
}
public static int findAllPeakElements(int[] arr, int n) {
    for (int i = 1; i < n - 1; i++) {
        int prev = arr[i - 1];
        int curr = arr[i];
        int next = arr[i + 1];
        if ( curr > prev && curr > next ) {
            return i;
        }
    }
    return -1;
}
```