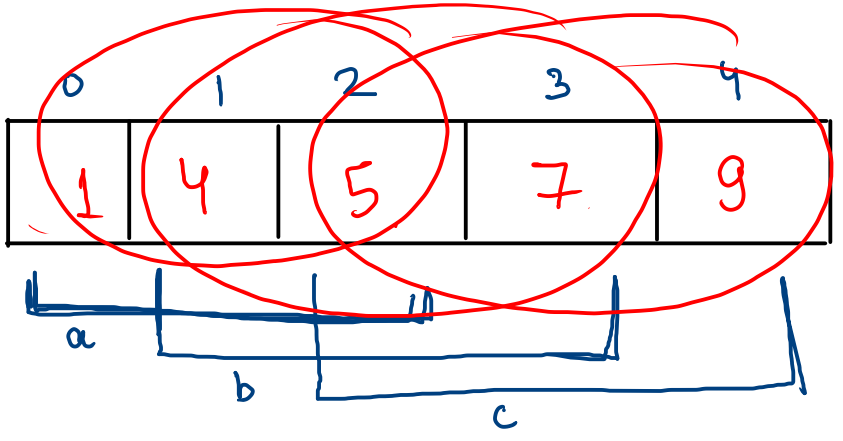
Minimum difference 7

9	1	4	7	5

K = 3

Sorted

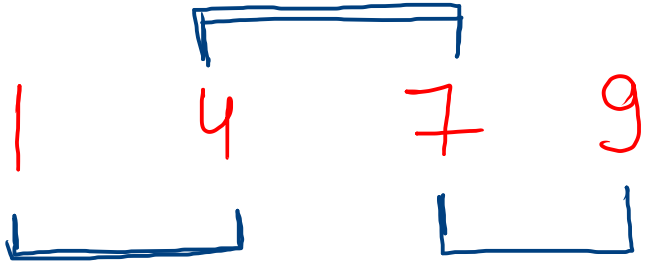


ans = ~~4~~ 3 → 4 5 7

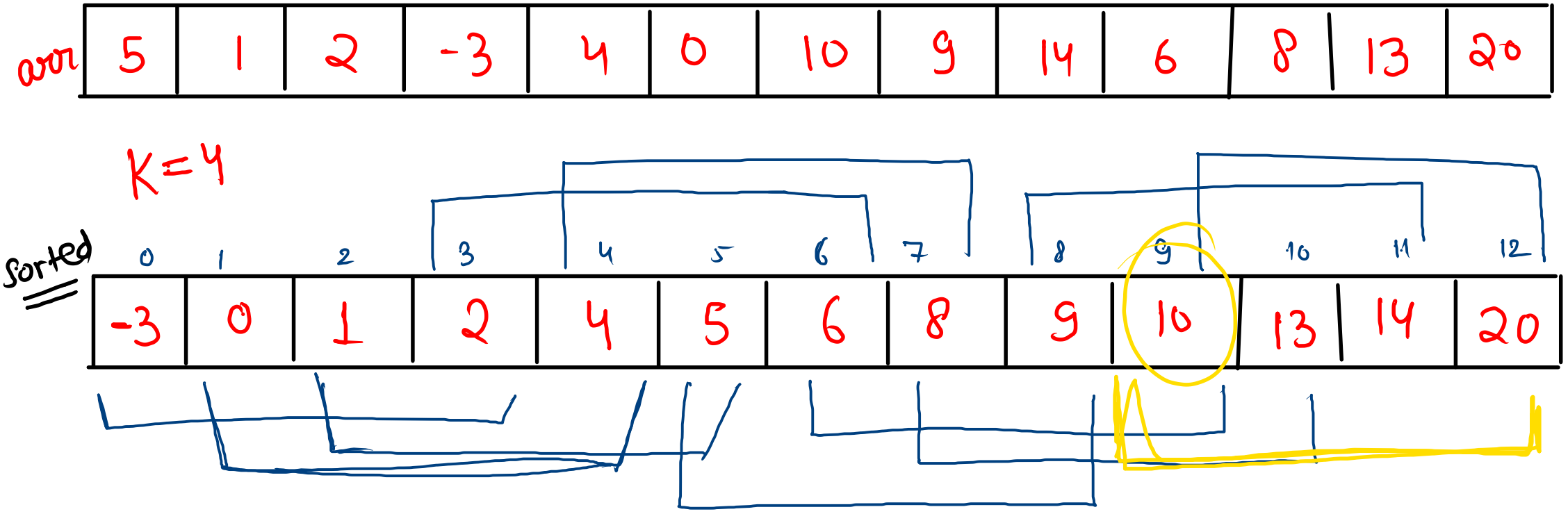
4

9 4 1 7

K=2



ans = ~~∞~~ ~~3~~ ~~3~~ 2



ans = ~~∞~~ ~~5~~ 4

i = 0, 3
i = 1, 4
i = 2, 5

i = 3, 6
i = 4, 7
i = 5, 8

i = 6, 9
i = 7, 10
i = 8, 11

i = 9, 12

mini idx = 0 → 9

maxi idx = 3 → 12

mini $i \rightarrow i+k-1$ maxi

k = 4

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();

    System.out.println(minDiff(arr, k));
}

public static int minDiff(int[] arr, int k) {
    // sort the array
    Arrays.sort(arr);

    int ans = Integer.MAX_VALUE;
    // find diff of mini and maxi value
    for (int i = 0; i <= arr.length - k; i++) {
        int miniValue = arr[i];
        int maxiValue = arr[i + k - 1];
        int diff = maxiValue - miniValue;
        if (diff < ans) {
            ans = diff;
        }
    }

    return ans;
}
```

psudo code :-

1) Sort the array

2) find diff. b/w mini and maxi value of each window of size 'k'.

Sort an array in wave form 1

arr

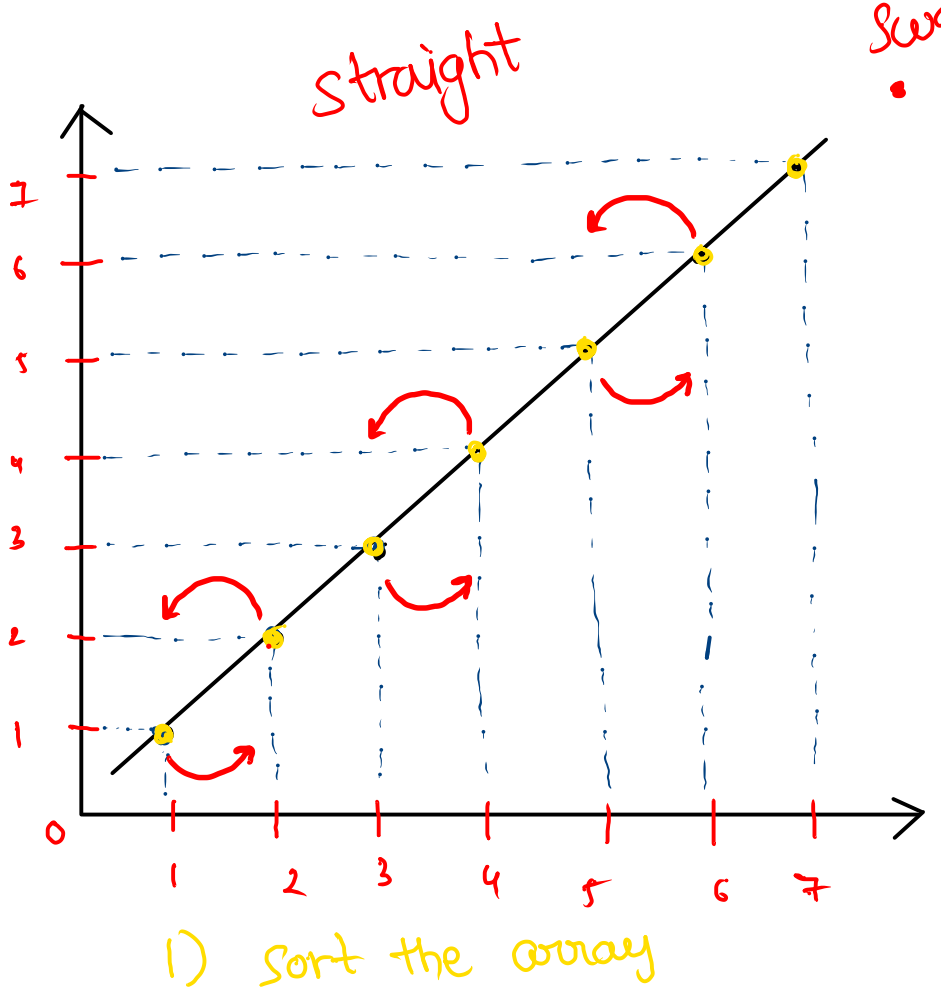
3	5	1	2	7	6	4

1) Sorted

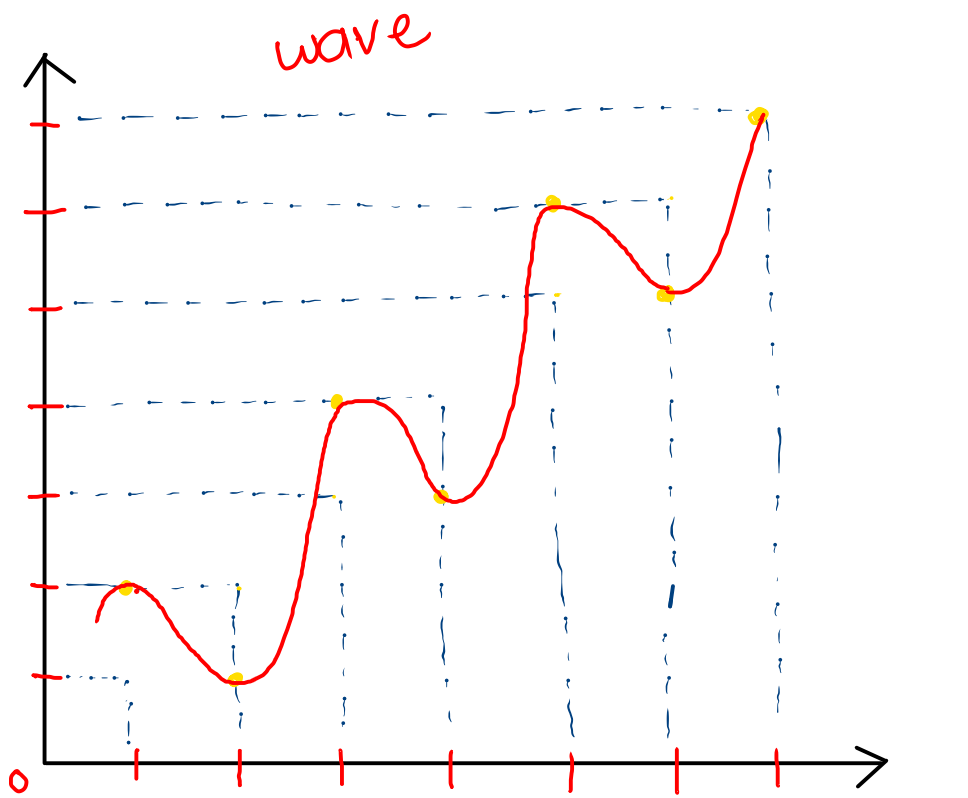
swap

1 2 3 4 5 6 7

2 >= 1 <= 4 >= 3 <= 6 >= 5 <= 7 wave



1) Sort the array

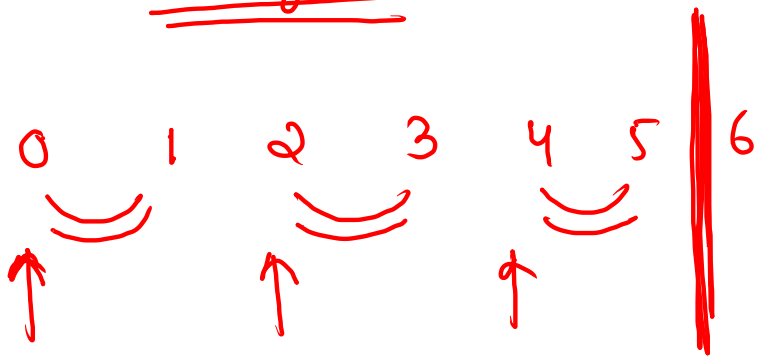


2) Swap each adjacent element

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();
    }

    waveForm(arr);
}

public static void waveForm(int[] arr) {
    // step1: sort the array
    Arrays.sort(arr);

    // step2: swap adj. elements
    for (int i = 0; i < arr.length - 1; i += 2) {
        swap(arr, i, i + 1);
    }

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

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

arr.length = 7 - 1 = 6



Form the largest number

```
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();
    }

    String[] arr1 = new String[n];
    for (int i = 0; i < n; i++) {
        arr1[i] = String.valueOf(arr[i]);
    }

    Arrays.sort(arr1, new myComparator());

    String ans = "";
    for (int i = 0; i < n; i++) {
        ans += arr1[i];
    }

    System.out.println(ans);
}

public static class myComparator implements Comparator<String> {
    @Override
    public int compare(String a, String b) {
        String num1 = a + b;
        String num2 = b + a;
        return num2.compareTo(num1);
    }
}
```

Input

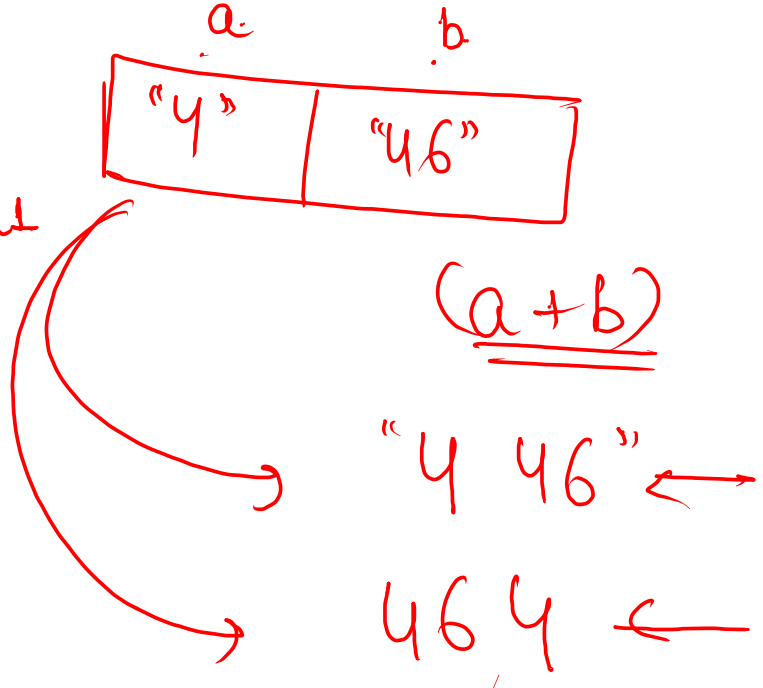
convert Integer value into String

Sort array acc. to largest no.

concatinating the String type arrl

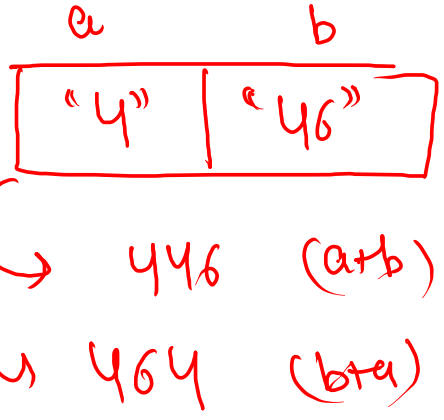
a+b

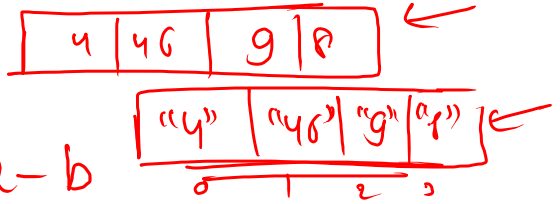
b+a



code in Lambda function

```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();
    }

    String[] arr1 = new String[n];
    for (int i = 0; i < n; i++) {
        arr1[i] = String.valueOf(arr[i]);
    }

    Arrays.sort(arr1, (a, b) -> {
        String num1 = a + b;
        String num2 = b + a;
        return num2.compareTo(num1);
    });

    String ans = "";
    for (int i = 0; i < n; i++) {
        ans += arr1[i];
    }

    System.out.println(ans);
}
```





num1 - num2 //inc.

num2 - num1 //dec

num1.compareTo(num2)

num2.compareTo(num1)

4 46 9 8
4 46 8 9
46 4 9 8
46 4 8 9
9 8 46 4
9 8 4 46