Minimum difference 7

You are given a 0-indexed integer array nums, where `nums[i]` represents the score of the **ith** student. You are also given an integer **k**.

Pick the scores of any **k** students from the array so that the difference between the **highest** and the **lowest** of the **k** scores is minimized.

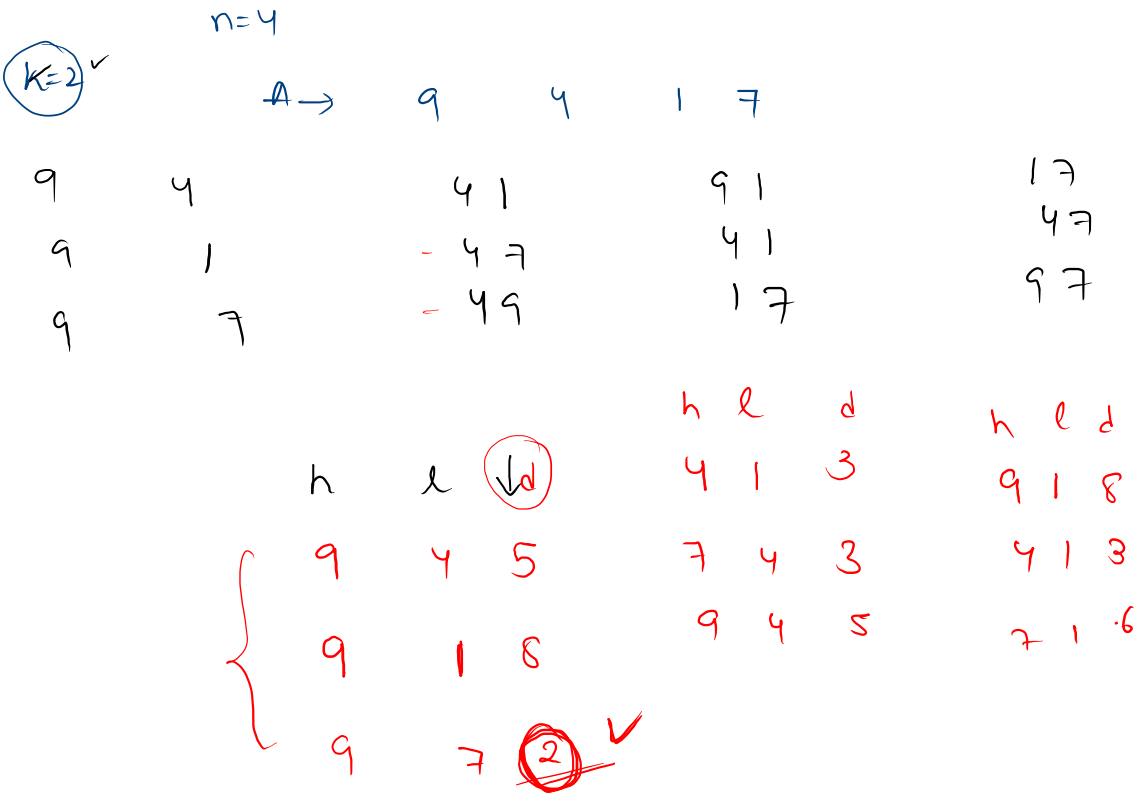
Return the minimum possible difference.

Sample Input 0

```
4
-9 4 1 7
2
```

Sample Output 0

```
2
```



n = 4

k = 2

ans = ?

9 4 1 7

h	l	d
9	7	2 ✓
4	1	3
7	1	6

k=3

9 4 1 7

0 1 2 3

9 4 1

9 1 7

4 1 7

4 7 9

4 9 7

h l d

9 1 8

9 1 8

7 4 3

9 4 5

9 4 5

K=3.

9 4 1 7

h e = d

9 4 = 5

9 1 = 8

7 1 = 6

9 1 = 8

9 4 7

9 1 7

4 1 7

1 7 9

k=4

9 1 4 7

h l d

ans=? 1 4 9 7 9 1 8

k = 2

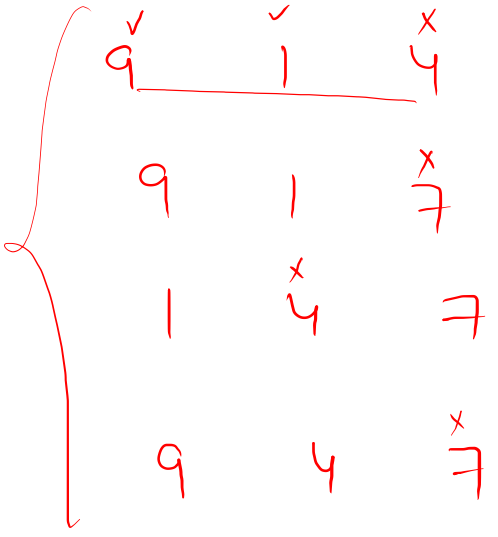
9 1 4 7

		h − l	d
9	1	9 − 1	8
9	4	9 − 4	5
9	7	9 − 7	(2)
1	4	4 − 1	3
1	7	7 − 1	6
4	7	7 − 4	3

k=3.

9 1 4 7

k=3.



h - l	d
9-1	8
9-1	8
7-1	6
9-4	5

k=4

k = 3

sorted

9 1 4 7

1 4 7 9

= 8

$k = 3$

1	4	7	9
0	1	2	3

i

A. $\begin{cases} i \le n-k \\ i < n-k-1 \end{cases}$

$n-k$

$4-3+1$

B. $\{$ $n-k+1$

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i< n; i++){
            A[i] = scn.nextInt();
        }
        int k = scn.nextInt();
        Arrays.sort(A);
        int d = Integer.MAX_VALUE;
        for(int i = 0; i <= n-k ; i++){ //? a.
            int h = A[i+k-1] ; //??b.
            int l = A[i] ;
            d = Math.min(h-l, d);
        }
        System.out.println(d);
    }
}
```

9 1 4 7 → 1 4 7 9

K=2

K=3

K=4

1 4 7 9

0 1 2 3

k=3

k=3 → dry run.

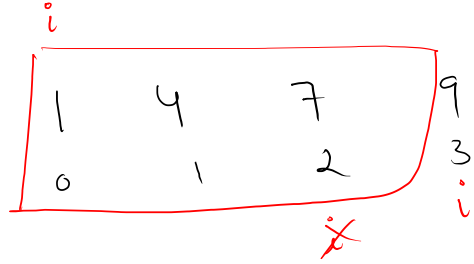
$i \le n-k$

$0 \le 2$

$1 \le 2$

$2 \le 2$

$3 \le 2$ ✗



k=2

d = ∞ → 8 → 2

```
5
6       public static void main(String[] args) {
7           Scanner scn = new Scanner(System.in);
8           int n = scn.nextInt();
9           int [] A = new int[n];
10          for(int i = 0; i< n; i++){
11              A[i] = scn.nextInt();
12          }
13          int k = scn.nextInt();
14          Arrays.sort(A);
15          int d = Integer.MAX_VALUE;
16          for(int i = 0; i <= n-k ; i++){ //? a.
17              int h = A[i+k-1] ; //??b.
18              int l = A[i] ;
19              d = Math.min(h-l, d);
20          }
21          System.out.println(d);
22      }
23  }
```

h = A[i+k-1] = A[2+2-1] = A[3] = 9

l = A[i] = A[2] = 7

2, 3

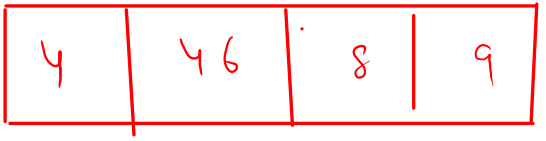
Form the largest number

Meet Sarah, an enthusiastic programmer who loves to solve challenging problems. She was recently given an array of **non-negative** integers and was asked to arrange its elements in such a way that they form the **largest** possible number.

Solve the problem by comparing the values of the elements in a way that produced the **maximum** possible number.

Sample Input 0

```
4
4 46 8 9
```

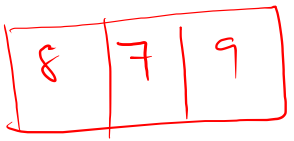


Sample Output 0

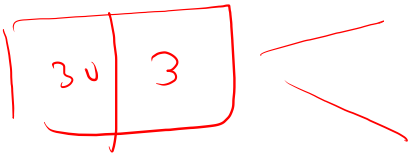
```
98464
```

eg.

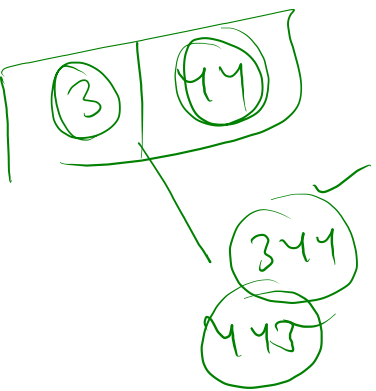
4 46 8 9 < 8 4469 ✓

eg. 

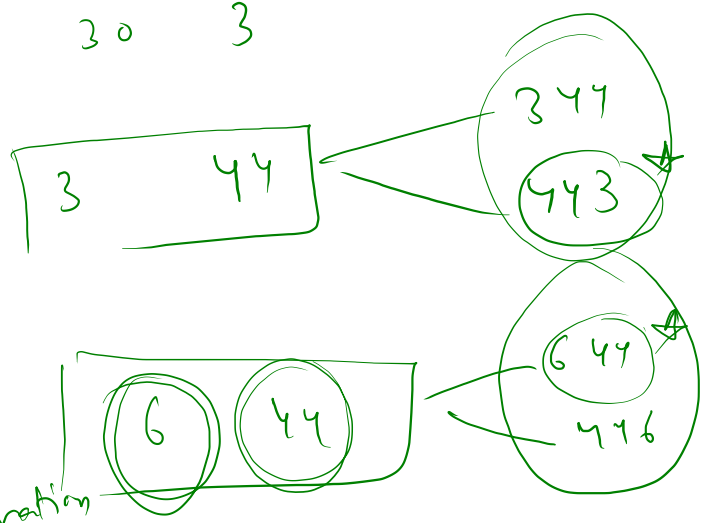
→ ? ⇒ 987

eg. 

303
330



30 3



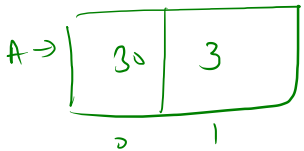
a + b.

String concatenation

```
Comparator<String> myComp = new Comparator<String>(){
    public int compare(String a, String b){
        String x = a + b;
        String y = b + a;

        // int v1 = Integer.parseInt(x);
        // int v2 = Integer.parseInt(y);

        // return v2-v1;
        return y.compareTo(x);  //compareTo method
    }
};
Arrays.sort(S, myComp);
```



+ve
-ve
0

x = "303"
y = "330"

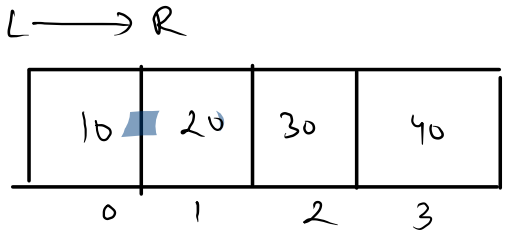
y - x

x.compareTo(y) = -ve.
303 330

y.compareTo(x) = +ve
330 303

a == b

a.equals(b)



Sub-array → contiguous part of an Array.

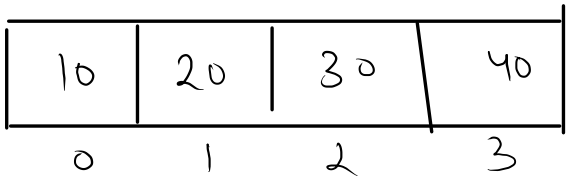
20 30 40 → sa ✓

10 20 → sa ✓

10 30 40 → sa ✗

10 20 30 40 → sa ✓

20 10 → sa ✗



$n = 4$

Count subarray = $n + n-1 + \dots + 1$

$= \frac{n(n+1)}{2}$

10_0

$10\ 20_1$

$10\ 20\ 30_2$

$10\ 20\ 30\ 40_3$

20_1

$20\ 30_2$

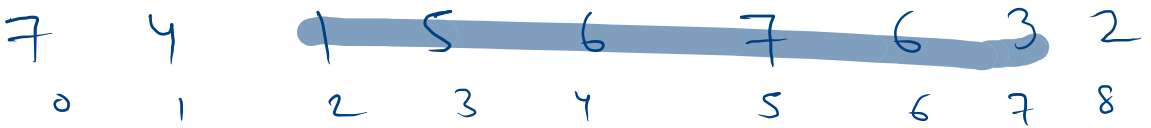
$20\ 30\ 40_3$

30_2

$30\ 40_3$

40_3

Start	end
0	0 1 2 3
1	1 2 3
2	2 3
3	3



start = 2

end = 7


```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        //logic
        for(int start = 0; start < n; start++){
            for(int end = start; end < n; end++){

                for(int k = start; k <= end; k++){
                    System.out.print(A[k] + " ");
                }
                System.out.println();
            }
        }

```