Flatland Space Stations

Flatland is a country with a number of cities, some of which have space stations. Cities are numbered consecutively and each has a road of $1km$ length connecting it to the next city. It is not a circular route, so the first city doesn't connect with the last city. Determine the maximum distance from any city to its nearest space station.

Example
$n = 3$
$c = [1]$

There are $n = 3$ cities and city 1 has a space station. They occur consecutively along a route. City 0 is $1 - 0 = 1$ unit away and city 2 is $2 - 1 = 1$ units away. City 1 is 0 units from its nearest space station as one is located there. The maximum distance is 1.

Function Description

Complete the *flatlandSpaceStations* function in the editor below.

flatlandSpaceStations has the following parameter(s):

- *int n:* the number of cities
- *int c[m]:* the indices of cities with a space station

Returns
- *int:* the maximum distance any city is from a space station

Input Format

The first line consists of two space-separated integers, n and m.
The second line contains m space-separated integers, the indices of each city that has a space-station. These values are *unordered* and distinct.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq m \leq n$
- There will be at least 1 city with a space station.
- No city has more than one space station.

Output Format

Sample Input 0

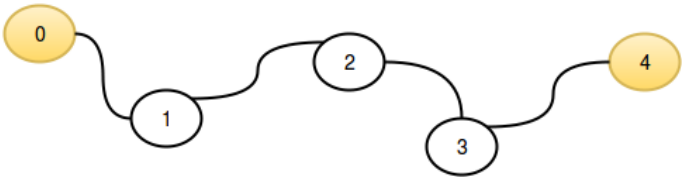
```
STDIN   Function
-----   --------
5 2     n = 5, c[] size m = 2
0 4     c = [0, 4]
```

Sample Output 0

```
2
```

Explanation 0

This sample corresponds to following graphic:



The distance to the nearest space station for each city is listed below:

- $c[0]$ has distance $0\ km$, as it contains a space station.
- $c[1]$ has distance $1\ km$ to the space station in $c[0]$.
- $c[2]$ has distance $2\ km$ to the space stations in $c[0]$ and $c[4]$.
- $c[3]$ has distance $1\ km$ to the space station in $c[4]$.
- $c[4]$ has distance $0\ km$, as it contains a space station.

We then take $max(0, 1, 2, 1, 0) = 2$.

Sample Input 1

```
6 6
0 1 2 4 3 5
```

Sample Output 1

```
0
```

Explanation 1

In this sample, $n = m$ so every city has space station and we print 0 as our answer.

Submissions: 19
Max Score: 25
Difficulty: Easy

Rate This Challenge:

More

Java 7

```
1  import java.io.*;
2  import java.math.*;
3  import java.security.*;
4  import java.text.*;
5  import java.util.*;
6  import java.util.concurrent.*;
7  import java.util.regex.*;
8
9  public class Solution {
10
11     // Complete the flatlandSpaceStations function below.
12     static int flatlandSpaceStations(int n, int[] c) {
13         if(n == c.length)
14             return 0;
15
16         Arrays.sort(c);
17         int maxDis = 0;
18         maxDis = Math.max(maxDis, c[0]-0);
```

```
        maxDis = Math.max(maxDis, (n-1)-c[c.length-1]);
        for(int i=1;i<c.length;i++){
            maxDis = Math.max(maxDis, (c[i]-c[i-1])/2);
        }
        return maxDis;
    }

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) throws IOException {
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        String[] nm = scanner.nextLine().split(" ");

        int n = Integer.parseInt(nm[0]);

        int m = Integer.parseInt(nm[1]);

        int[] c = new int[m];

        String[] cItems = scanner.nextLine().split(" ");
        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

        for (int i = 0; i < m; i++) {
            int cItem = Integer.parseInt(cItems[i]);
            c[i] = cItem;
        }

        int result = flatlandSpaceStations(n, c);

        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();

        bufferedWriter.close();

        scanner.close();
    }
}

```

Line: 24 Col: 6

Upload Code as File Test against custom input Run Code Submit Code