Codility_

Alexandru Dascal

Candidate

E-mail:
alexandrudascal94@gmail.com

Session

ID: K2S9K7-JCA
Time limit: 145 min.
Report recipients: No one
Accessed from: 86.81.229.9, 86.81.229.9
Invited by:
raimonda.fetingyte.external@zalando.de

Status: completed

Invitation: sent
Created on: 2022-07-16 10:48 UTC
Started on: 2022-07-17 08:57 UTC
Finished on: 2022-07-17 11:19 UTC

Notes:
N/A

Similarity Check

Status: not found
No similar solutions have been detected.

Test score

65%

Tasks in test	Score
1 BugfixingBisection Submitted in: Java 8	55%
2 SkylinePaint Submitted in: Java 11	100%
3 ScheduleTrainings Submitted in: Java 8	40%

Tasks Details

Easy

1. BugfixingBisection

Find and correct bugs in the bisection algorithm.

Task Score	Correctness	Performance
55	55	Not assessed

Task description

You are given an implementation of a function:

```
class Solution { public int solution(int[] A, int X); }
```

This function, when given an array A of N integers, sorted in non-decreasing order, and some integer X, looks for X in A. If X is present in A, then the function returns position of (some) occurrence of X in A. Otherwise, the function returns −1.

For example, given the following array:

```
A[0] = 1    A[1] = 2    A[2] = 5
A[3] = 9    A[4] = 9
```

and X = 5, the function should return 2, as A[2] = 5.

The attached code is still **incorrect** for some inputs. Despite the error(s), the code may produce a correct answer for the example test cases. The goal of the exercise is to find and fix the bug(s) in the implementation. You can modify at most **three** lines.

Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,000,000,000..2,000,000,000];
- array A is sorted in non-decreasing order;
- X is an integer within the range [−2,000,000,000..2,000,000,000].

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Solution

See Live Version

Programming language used:	Java 8
Total time used:	15 minutes
Effective time used:	15 minutes
Notes:	*not defined yet*

Source code

Code: 09:11:49 UTC, java, final, score: **55**

```
1   class Solution {
2       int solution(int[] A, int X) {
3           int N = A.length;
4           if (N == 0) {
5               return -1;
6           }
7           int l = 0;
8           int r = N - 1;
9           while (l < r) {
10              int m = (l + r) / 2;
+               if (A[m] == X) {
+                   return m;
+               }
11              if (A[m] > X) {
12                  r = m - 1;
```

```
13              } else {
14                  l = m;
15              }
16          }
17  -       if (A[l] == X) {
18  -           return l;
19  -       }
20          return -1;
21      }
22  }
```

Analysis summary

The following issues have been detected: wrong answers, timeout errors.

For example, for the input `([3], 3)` the solution returned a wrong answer (got -1, expected 0).

Analysis

Example tests		
	example example from the problem statement	✔ OK
Correctness tests		
	simple simple correctness test	✘ TIMEOUT ERROR Killed. Hard limit reached: 11.000 sec.
	empty empty array	✔ OK
	single_double single and double element	✘ WRONG ANSWER got -1, expected 0
	not_present X not present in A	✘ TIMEOUT ERROR Killed. Hard limit reached: 11.000 sec.
	negative negative numbers	✔ OK
	constant_segment all elements in A are equal	✔ OK
	random_small_positive random test with small positive values	✔ OK
	random_medium_values medium random test with medium values	✔ OK
	random_large_values maximum size random test	✘ TIMEOUT ERROR Killed. Hard limit reached: 11.000 sec.

Medium

2. SkylinePaint

Compute the minimum number of horizontal brushstrokes needed to paint a skyline shape.

Task Score	Correctness	Performance
100	100	100

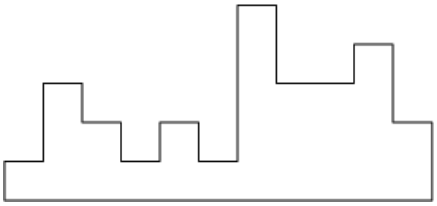
Task description

Your room is being decorated. On the largest wall you would like to paint a skyline. The skyline consists of rectangular buildings arranged in a line. The buildings are all of the same width, but they may have different heights. The skyline shape is given as an array A whose elements specify the heights of consecutive buildings.

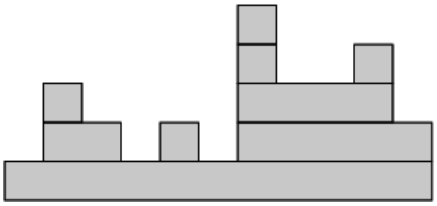
For example, consider array A such that:

```
A[0]  = 1
A[1]  = 3
A[2]  = 2
A[3]  = 1
A[4]  = 2
A[5]  = 1
A[6]  = 5
A[7]  = 3
A[8]  = 3
A[9]  = 4
A[10] = 2
```

The shape specified by this array is represented by the figure below.



You would like to paint the skyline using continuous horizontal brushstrokes. Every horizontal stroke is one unit high and arbitrarily wide. The goal is to calculate the minimum number of horizontal strokes needed. For example, the above shape can be painted using nine horizontal strokes.



Starting from the bottom, you can paint the skyline in horizontal stripes with 1, 3, 2, 2, 1 strokes per respective stripe.

Write a function:

```
class Solution { public int solution(int[] A); }
```

that, given a non-empty array A consisting of N integers, returns the minimum number of horizontal brushstrokes needed to paint the shape represented by the array.

The function should return −1 if the number of strokes exceeds 1,000,000,000.

For example, given array A as described above, the function should return 9, as explained above.

On the other hand, for the following array A:

```
A[0] = 5
A[1] = 8
```

the function should return 8, as you must paint one horizontal stroke at each height from 1 to 8.

For the following array:

```
A[0] = 1
A[1] = 1
```

```
A[2] = 1
A[3] = 1
```

the function should return 1, as you can paint this shape using a single horizontal stroke.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [1..1,000,000,000].

Solution

See Live Version

Programming language used:	Java 11
Total time used:	49 minutes
Effective time used:	34 minutes
Notes:	*not defined yet*

Source code

Code: 09:48:31 UTC, java11, final, score: **100**

```java
// you can also use imports, for example:
// import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {

    // time O(n), space o(1)
    public int solution(int[] A) {
        int strokes = A[0];
        int maxStrokes = 1000000000;

        for(int i = 0; i < A.length - 1; i++){
            int current = A[i];
            int next = A[i + 1];
            int difference = next - current;

             if(difference > 0){
                 strokes += difference;
             }

             if(strokes > maxStrokes){
                 return -1;
             }
        }

        return strokes;
    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N)**

Example tests		
	example1 first example test	✔ OK
	example2 second example test	✔ OK
	example3 third example test	✔ OK
Correctness tests		
	single single element	✔ OK
	double two elements	✔ OK
	triple three elements	✔ OK
	small small functional tests	✔ OK
	small_random small random, N <= 100, max(A) <= 100	✔ OK
Performance tests		
	medium_random medium random, N <= 800	✔ OK
	large_random large random	✔ OK
	large_maximum_result large, maximum result	✔ OK
	large_arithmetic large, arithmetic sequences	✔ OK
	special result exceeds 1,000,000,000 after last element	✔ OK

Easy

3. ScheduleTrainings

Given a list of the free days of each employee, output the maximum number of employees available on at least one of two arbitrary days.

Task Score	Correctness	Performance
40	40	Not assessed

Task description

A one-day-long training session will be conducted twice during the next 10 days. There are N employees (numbered from 0 to N−1) willing to attend it. Each employee has provided a list of which of the next 10 days they are able to participate in the training. The employees' preferences are represented as an array of strings. E[K] is a string consisting of digits ('0'-'9') representing the days the K-th employee is able to attend the training.

The dates during which the training will take place are yet to be scheduled. What is the maximum number of employees that can attend during at least one of the two scheduled days?

Write a function:

```
class Solution { public int solution(String[] E); }
```

that, given an array E consisting of N strings denoting the available days for each employee, will return the maximum number of employees that can attend during at least one of the two scheduled days.

Examples:

1. Given E = ["039", "4", "14", "32", "", "34", "7"], the answer is 5. It can be achieved for example by running training on days 3 and 4. This way employees number 0, 1, 2, 3 and 5 will attend the training.

2. Given E = ["801234567", "180234567", "0", "189234567", "891234567", "98", "9"], the answer is 7. It can be achieved by running training on days 0 and 9. This allows all employees to attend the training.

3. Given E = ["5421", "245", "1452", "0345", "53", "354"], the answer is 6. It can be achieved just by running training once on day 5, when every employee is available.

Assume that:

- N is an integer within the range [1..100];
- each string in array E consists only of digits (0−9);
- each string in array E has length within the range [0...10];
- characters in every string in array E are distinct.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Solution

See Live Version

Programming language used:	Java 8
Total time used:	140 minutes
Effective time used:	95 minutes
Notes:	*not defined yet*

Source code

Code: 11:19:45 UTC, java, final, score: **40**

```java
// you can also use imports, for example:
import java.util.*;

// you can write to stdout for debugging purposes, e.g.
// System.out.println("this is a debug message");

class Solution {
```

```
 8
 9  public int solution(String[] E) {
10      int DAYS = 10;
11      // day, employees idx
12      HashMap<Integer, ArrayList<Integer>> distribution = new HashMap<>();
13      fillHashMap(distribution, DAYS);
14
15      for(int i = 0; i < E.length; i++){
16          char[] employeeAvailabilities = E[i].toCharArray();
17
18          for(int j = 0; j < employeeAvailabilities.length; j++){
19              int availableDay = Character.getNumericValue(employeeAvailabilities[j]);
20              ArrayList<Integer> dayAvailabilies = distribution.get(availableDay);
21              dayAvailabilies.add(i);
22              distribution.put(availableDay, dayAvailabilies);
23          }
24      }
25
26  // This solutions will check the number of max employees that can attend the
27  // training thake take place 2 consecutive days. released at the end that
28  // I missed that the day order does't matter
29
30      int maxAttendance = 0;
31      for(int i = 0; i < DAYS - 1; i++){
32
33          ArrayList<Integer> dayOne = distribution.get(i);
34          ArrayList<Integer> dayTwo = distribution.get(i + 1);
35
36          dayOne.addAll(dayTwo);
37          int attendance = (int)dayOne.stream().distinct().count();
38
39          if(attendance > maxAttendance){
40              maxAttendance = attendance;
41          }
42      }
43
44      return maxAttendance;
45  }
46
47  private void fillHashMap(HashMap<Integer, ArrayList<Integer>> map, int entryCount){
48      for(int i = 0; i < entryCount; i++){
49          map.put(i, new ArrayList<Integer>());
50      }
51  }
52}
```

Analysis summary

The following issues have been detected: wrong answers.

Analysis

Example tests	
example1 First example test.	✔ OK
example2 Second example test.	✘ WRONG ANSWER got 6 expected 7
example3 Third example test.	✔ OK
Correctness tests	
no_intersection Each employee is available at most once.	✔ OK
most_popular Two most popular days are the ones to be chosen.	✔ OK
least_popular Two least popular days are the ones to be chosen.	✘ WRONG ANSWER got 11 expected 12

middle_days Two averagely popular days are the ones to be chosen. Score x 2.	✘ WRONG ANSWER got 11 expected 12
random_tests Random tests. Score x 2.	✘ WRONG ANSWER got 8 expected 9
one_day There is a day on which all employees are available.	✔ OK
mixed Mixed tests.	✘ WRONG ANSWER got 69 expected 75
corner Corner cases.	✔ OK