MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

LAB 7 Writeup(Maximum Element After Decreasing and Rearranging)

Aswin Sampath
saswin@uci.edu
53844684

Description | Editorial | Solutions | Submissions | Accepted ×

1846. Maximum Element After Decreasing and Rearranging

Medium | Topics | Companies | Hint

You are given an array of positive integers `arr`. Perform some operations (possibly none) on `arr` so that it satisfies these conditions:

- The value of the **first** element in `arr` must be `1`.
- The absolute difference between any 2 adjacent elements must be **less than or equal to** `1`. In other words, `abs(arr[i] - arr[i - 1]) <= 1` for each `i` where `1 <= i < arr.l` **indexed**). `abs(x)` is the absolute value of `x`.

There are 2 types of operations that you can perform any number of times:

- **Decrease** the value of any element of `arr` to a **smaller positive integer**.
- **Rearrange** the elements of `arr` to be in any order.

Return *the **maximum** possible value of an element in* `arr` *after performing the operations to satisfy the conditions.*

Example 1:

```
Input: arr = [2,2,1,2,1]
Output: 2
Explanation:
We can satisfy the conditions by rearranging arr so it becomes [1,2,2,2,1].
The largest element in arr is 2.
```

Example 2:

```
Input: arr = [100,1,1000]
```

Description | Editorial | Solutions | Submissions | Accepted ×

```
We can satisfy the conditions by rearranging arr so it becomes [1,2,2,2,1].
The largest element in arr is 2.
```

Example 2:

```
Input: arr = [100,1,1000]
Output: 3
Explanation:
One possible way to satisfy the conditions is by doing the following:
1. Rearrange arr so it becomes [1,100,1000].
2. Decrease the value of the second element to 2.
3. Decrease the value of the third element to 3.
Now arr = [1,2,3], which satisfies the conditions.
The largest element in arr is 3.
```

Example 3:

```
Input: arr = [1,2,3,4,5]
Output: 5
Explanation: The array already satisfies the conditions, and the largest element is 5.
```

Constraints:

- `1 <= arr.length <= 10^5`
- `1 <= arr[i] <= 10^9`

Understanding the Problem

The problem involves performing operations on an array of positive integers to satisfy two conditions:

- The first element in the array must be 1.
- The absolute difference between any two adjacent elements must be less than or equal to 1.

The goal is to return the maximum possible value of an element in the array after performing the operations to meet these conditions.

Identifying Edge Cases

1. An array with only one element: When the array contains only one element, the maximum value after operations will be 1.
2. An array already satisfying the conditions: If the array already satisfies the conditions, with elements sorted and having absolute differences of 1 or less, the maximum value will be the maximum element in the array.

Effective Test Cases

- An array with repeating elements:
  - Input: arr = [2, 2, 1, 2, 1]
  - Expected Output: 2
- An array with distinct elements:
  - Input: arr = [5, 10, 15, 20]
  - Expected Output: 20
- An array requiring rearrangement and value decrements:
  - Input: arr = [100, 1, 1000]
  - Expected Output: 3

Algorithmic Solution

The provided C++ code solves the problem by:

- Sorting the array.
- Setting the first element to 1.
- Iterating through the array from the second element onward, if the absolute difference between the current and previous elements is more than 1, setting the current element to be one more than the previous element.

Time and Space Complexity Analysis

The time complexity of this algorithm is O(n log n) due to the sorting operation, where n is the size of the input array.

The space complexity is O(1) as the algorithm sorts the input array in-place and uses only a constant amount of additional space.