Contents

Show Answers

Interview Questions

Leetcode – 164. Maximum Gap

Posted on Jun 4 by Deepanshu Jain

Given an integer array nums, return the maximum difference between two successive elements in their sorted form. If the array contains less than two elements, return 0.

You must write an algorithm that runs in linear time and you can linear extra space.

Example 1:

```
Input: nums = [3,6,9,1]
Output: 3
Explanation: The sorted form of the array is
[1,3,6,9], either (3,6) or (6,9) has the maximum
difference 3.
```

Example 2:

```
Input: nums = [10]
Output: 0
Explanation: The array contains less than 2 elements,
therefore return 0.
```

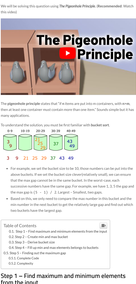
Constraints:

- 1 <= nums.length <= 10^5
- 0 <= nums[i] <= 10^9

Solution:

The main problem is we can't sort the array as it will take n log(n) complexity, but as asked we can't do that.

We will be solving this using the **The Pigeonhole Principle**. **Discovered**by Walsh this called



> For example, we are left the bucket size is size 10, these numbers can be put into the above buckets. If we set the bucket size (heretofore) smaller, we can ensure that the max gap, must be found in the same bucket. In the worst case, each successive numbers have the same gap. For example, we have 1, 3, 5 the gap and the max gap is (5 - 1) / 2 = 2 apart . Smallest, first gaps.
>
> Based on this, we only need to compare the max number in this bucket and the min number in the next bucket to get the relatively large gap and find out which two buckets have the largest gap.

Table of Contents

0.1. Step 1 – Find maximum and minimum elements from the input
0.2. Step 2 – Create min and max bucket
0.3. Step 3 – Derive bucket size
0.4. Step 4 – Fill up min and max elements belongs to buckets
0.5. Step 5 – Finding out the maximum gap
0.6.1. Complete Code
0.6.2. Complexity

Step 1 – Find maximum and minimum elements from the input

```
Java
int min = nums[0];
int max = nums[0];
for(int num:nums){
    min = Math.min(num, min);
    max = Math.max(num, max);
}
```

Step 2 – Create min and max bucket

```
Java
int[] minBucket = new int[n];
int[] maxBucket = new int[n];

Arrays.fill(minBucket, Integer.MIN_VALUE);
Arrays.fill(maxBucket, Integer.MIN_VALUE);
```

Step 3 – Derive bucket size

```
Java
int bucketSize = (int)Math.ceil((double)(max-min)/(n - 1));
```

Step 4 – Fill up min and max elements belongs to buckets

As we can't have the maximum gap lies in the same bucket (because of maximum gap we can have is bucketSize-1), so we will be maintaining only min and max elements in the respective buckets.

```
Java
for(int num:nums){
    int bucketIndex = (num-min) / bucketSize;
    minBucket[bucketIndex] = Math.min(num,minBucket[bucketIndex]);
    maxBucket[bucketIndex] = Math.max(num,maxBucket[bucketIndex]);
}
```

Step 5 – Finding out the maximum gap

```
Java
int maxGap = 0;
int previous = minBucket[0];

for(int i=1;i<n;i++){
    if(maxBucket[i] == Integer.MIN_VALUE)
        continue;
    maxGap = Math.max(maxGap, minBucket[i] - previous);
    previous = maxBucket[i];
}
```

Complete Code

```
Java
class Solution {
    public int maximumGap(int[] nums) {
        int n = nums.length;

        if(n < 2)
            return 0;
        int min = nums[0];
        int max = nums[0];

        for(int num:nums){
            min = Math.min(num, min);
            max = Math.max(num, max);
        }

        if(max == min)
            return 0;

        int[] minBucket = new int[n];
        int[] maxBucket = new int[n];

        Arrays.fill(minBucket, Integer.MIN_VALUE);
        Arrays.fill(maxBucket, Integer.MIN_VALUE);

        int bucketSize = (int)Math.ceil((double)(max-min)/(n - 1));

        for(int num:nums){
            int bucketIndex = (num-min) / bucketSize;
            minBucket[bucketIndex] = Math.min(num,minBucket[bucketIndex]);
            maxBucket[bucketIndex] = Math.max(num,maxBucket[bucketIndex]);
        }

        int maxGap = 0;
        int previous = minBucket[0];

        for(int i=1;i<n;i++){
            if(maxBucket[i] == Integer.MIN_VALUE)
                continue;
            maxGap = Math.max(maxGap, minBucket[i] - previous);
            previous = maxBucket[i];
        }

        return maxGap;
    }
}
```

Complexity

Time complexity: O(n)

Space complexity: O(n)

Share on Social Media

Twitter
Facebook
Reddit
Tumblr
email

Related posts:

Two non-adjacent Solution
In "Interview Questions"

Max Area of Island Leetcode 695
In "Interview Questions"

Leetcode 1423. Maximum Gain Leetcode
In "Interview Questions"

leetcode • leetcode solution • Maximum Gap Leetcode

← OFF CAMPUS DRIVES

MAX AREA OF ISLAND LEETCODE 695 →

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Please enter an answer in digits:

thirteen + twenty =

Post Comment

Email

Subscribe

Recent Posts

Introducing – Manager, Software Engineering Management

Internship Interview Experience – Microsoft

Phonon Phoxels – Intern to Hire

Insufficient get lock for Redis resource temporarily unavailable

ServiceNow Hiring Project/Program Manager – Scrum Master

Follow us

Facebook
Linkedin
Twitter
Youtube
Tiktok
email

Latest News

Posted on Jun 4
SERVICENOW – MANAGER, SOFTWARE ENGINEERING MANAGEMENT

Posted on Jun 4
PHONON PHOXELS – INTERN TO HIRE

Posted on Jun 4
SERVICENOW HIRING PROJECT/PROGRAM MANAGER – SCRUM MASTER

Posted on Jun 4
AMAZON – SOFTWARE DEVELOPMENT INTERSHIP

Posted on Jun 4
VISUAL 3D – PRESENT – SOFTWARE ENGINEER

Proudly powered by WordPress | Theme: TrendNews | By ThemeXpert.com.