Leetcode 220: Contains Duplicate III

Posted on July 6, 2018 · 3 minute read

Question

Given an array of integers, find out whether there are two distinct indices i and j in the array such that the absolute difference between nums[i] and nums[j] is at most t and the absolute difference between i and j is at most k.

Example 1:

```
Input: nums = [1,2,3,1], k = 3, t = 0
Output: true
```

Example 2:

```
Input: nums = [1,0,1,1], k = 1, t = 2
Output: true
```

Example 3:

```
Input: nums = [1,5,9,1,5,9], k = 2, t = 3
Output: false
```

Solution

Inspired by bucket sort, we can achieve **linear time complexity** in our problem using buckets as window.

Bucket sort is a sorting algorithm that works by distributing the elements of an array into a number of buckets. Each bucket is then sorted individually, using a different sorting algorithm. For example, we have 8 unsorted integers, 20, 13, 4, 49, 41, 32, 9, 18. We create 5 buckets covering the inclusive ranges of [0,9], [10,19], [20, 29], [30, 39], [40, 49] individually. Each of the eight elements is in a particular bucket. For element with value x, its bucket label is `x / w` and here we have `w = 10`. Sort each bucket using some other sorting algorithm and then collect all of them bucket by bucket.

Back to our problem, the critical issue we are trying to solve is: For a given element x is there an item in the window that is within the range of `[x-t, x+t]`?

Could we do this in constant time?

Let us consider an example where each element is a person's birthday. Your birthday, say some day in March, is the new element `x`. Suppose that each month has 30 days and you want to know if anyone has a birthday within `t = 30` days of

yours. Immediately, we can rule out all other months except February, March, April.

The reason we know this is because each birthday belongs to a bucket we called month! And the range covered by the buckets are the same as distance t which simplifies things a lot. Any two elements that are not in the same or adjacent buckets must have a distance greater than t.

We apply the above bucketing principle and design buckets covering the ranges of `..., [0,t], [t+1, 2t+1], ......,[0,t],[t+1,2t+1],...` We keep the window using this buckets. Then, each time, all we need to check is the bucket that x belongs to and its two adjacent buckets. Thus, we have a constant time algorithm for searching almost duplicate in the window.

One thing worth mentioning is the difference from bucket sort – Each of our buckets contains at most one element at any time, because two elements in a bucket means "almost duplicate" and we can return early from the function. Therefore, a HashMap with an element associated with a bucket label is enough for our purpose.

```
class Solution(object):
    def getId(self, n, w):
        if n >= 0:
            return n // w
        else:
            return n // w - 1

    def containsNearbyAlmostDuplicate(self, nums, k, t):
        """
        :type nums: List[int]
        :type k: int
        :type t: int
        :rtype: bool
        """
        if k <= 0 or t < 0:
            return False

        w = t + 1
        buckets = dict()
        for i in range(len(nums)):
            n = nums[i]
            id = self.getId(n, w)
            if id in buckets:
                return True
            if id - 1 in buckets and -t <= n - buckets[id - 1] <= t:
                return True
            if id + 1 in buckets and -t <= n - buckets[id + 1] <= t:
                return True
            buckets[id] = n
            if i >= k:
                del buckets[self.getId(nums[i - k], w)]

        return False
```

Tags: Leetcode Hard Review

← PREVIOUS POST

NEXT POST →