Increasing Triplet Subsequence

Given an unsorted array return whether an increasing subsequence of length 3 exists or not in the array.

Formally the function should:

> Return true if there exists i, j, k
> such that $arr[i] < arr[j] < arr[k]$ given $0 \le i < j < k \le n-1$ else return false.

Your algorithm should run in $O(n)$ time complexity and $O(1)$ space complexity.

Examples:
Given `[1, 2, 3, 4, 5]`,
return `true`.

Given `[5, 4, 3, 2, 1]`,
return `false`.

Credits:
Special thanks to @DjangoUnchained for adding this problem and creating all test cases.

Solution 1

```
bool increasingTriplet(vector<int>& nums) {
    int c1 = INT_MAX, c2 = INT_MAX;
    for (int x : nums) {
        if (x <= c1) {
            c1 = x;           // c1 is min seen so far (it's a candidate for 1st
element)
        } else if (x <= c2) { // here when x > c1, i.e. x might be either c2 or c
3
            c2 = x;           // x is better than the current c2, store it
        } else {              // here when we have/had c1 < c2 already and x > c2
            return true;      // the increasing subsequence of 3 elements exists
        }
    }
    return false;
}
```

written by alveko original link here

Solution 2

```
   public boolean increasingTriplet(int[] nums) {
        // start with two largest values, as soon as we find a number bigger than
both, while both have been updated, return true.
        int small = Integer.MAX_VALUE, big = Integer.MAX_VALUE;
        for (int n : nums) {
            if (n <= small) { small = n; } // update small if n is smaller than b
oth
            else if (n <= big) { big = n; } // update big only if greater than sm
all but smaller than big
            else return true; // return if you find a number bigger than both
        }
        return false;
    }
```

written by sim5 original link here

Solution 3

The main idea is keep two values when check all elements in the array: the minimum value *min* until now and the second minimum value *secondMin* from the minimum value's position until now. Then if we can find the third one that larger than those two values at the same time, it must exists the triplet subsequence and return true.

What need to be careful is: we need to include the condition that some value has the same value with minimum number, otherwise this condition will cause the secondMin change its value.

```
public class Solution {
    public boolean increasingTriplet(int[] nums) {
        int min = Integer.MAX_VALUE, secondMin = Integer.MAX_VALUE;
        for(int num : nums){
            if(num <= min) min = num;
            else if(num < secondMin) secondMin = num;
            else if(num > secondMin) return true;
        }
        return false;
    }
}
```

The running time complexity is O(n).

written by billleeforfun original link here