给定一个未排序的数组，判断这个数组中是否存在长度为 3 的递增子序列。

数学表达式如下:

如果存在这样的 *i, j, k,* 且满足 0 ≤ *i* < *j* < *k* ≤ *n*-1 ,

使得 *arr[i]* < *arr[j]* < *arr[k]* ， 返回 true ; 否则返回 false 。

说明: 要求算法的时间复杂度为 O(*n*)，空间复杂度为 O(*1*) 。

示例 1:

```
输入: [1,2,3,4,5]
输出: true
```

示例 2:

```
输入: [5,4,3,2,1]
输出: false
```

思路有点类似动态规划的思想，维护一个二元组(first,second)，记录第i个元素之前的“最小”递增二元子序列（对后续元素的要求最低如[5,6,2,3,4]会更新[5,6]为[2,3]此时只要后续满足大于3就可以）

- 当nums[i]小于first时，更新first的值
- 当nums[i]>first且nums[i]

by https://blog.csdn.net/whdAlive/article/details/80404875

```
class Solution {
public:
    bool increasingTriplet(vector<int>& nums) {
        if(nums.size()<3){
            return false;
        }
        int first = 2147483647,second = 2147483647;
        for(int i=0;i<nums.size();i++){
            if(nums[i]<=first){
                first=nums[i];
            }else if(nums[i]<=second){
                second=nums[i];
            }else if(first!=second){
                return true;
```

```
            }
        }
        return false;
    }
};
```