414 Third Maximum Number (link)

Description

Given an integer array `nums`, return *the **third distinct maximum** number in this array. If the third maximum does not exist, return the **maximum** number*.

Example 1:

```
Input: nums = [3,2,1]
Output: 1
Explanation:
The first distinct maximum is 3.
The second distinct maximum is 2.
The third distinct maximum is 1.
```

Example 2:

```
Input: nums = [1,2]
Output: 2
Explanation:
The first distinct maximum is 2.
The second distinct maximum is 1.
The third distinct maximum does not exist, so the maximum (2) is returned instead.
```

Example 3:

```
Input: nums = [2,2,3,1]
Output: 1
Explanation:
The first distinct maximum is 3.
The second distinct maximum is 2 (both 2's are counted together since they have the sa
The third distinct maximum is 1.
```

Constraints:

- `1 <= nums.length <= 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

Follow up: Can you find an `O(n)` solution?

(scroll down for solution)

Solution

Language: cpp

Status: Accepted

```cpp
#include <vector>
#include <unordered_set>
using namespace std;

class Solution {
public:
    int thirdMax(vector<int>& nums) {
        long long first = LLONG_MIN, second = LLONG_MIN, third = LLONG_MIN;
        unordered_set<int> uniqueNums;

        for (int num : nums) {
            uniqueNums.insert(num);
            if (num > first) {
                third = second;
                second = first;
                first = num;
            } else if (num > second && num < first) {
                third = second;
                second = num;
            } else if (num > third && num < second) {
                third = num;
            }
        }

        return uniqueNums.size() >= 3 ? third : first;
    }
};
```