UNIQUE NUMBER OF OCCURRENCES (LEETCODE-1207)

GitHub: **github.com/BCAPATHSHALA**
Date: 30-12-2023

EX1

arr

1	2	2	1	1	3
0	1	2	3	4	5

1 → 3 times
2 → 2 times
3 → 1 time

} All occurrences are unique

output **TRUE**

EX2

arr

1	2
0	1

1 → 1 time
2 → 2 times

} Diff.n occurrences

output **False**

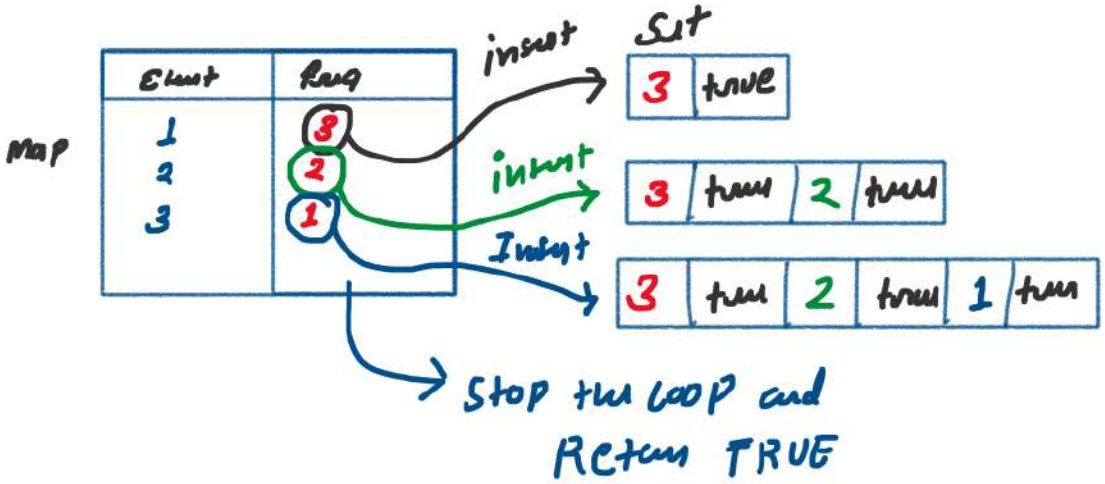
Algorithm

EX1

arr

1	2	2	1	1	3
0	1	2	3	4	5

DRY RUN STEP2



STEP1 Frequency counting using map

Map

Elmnt	Freq
1	3
2	2
3	1

STEP2 Checking for unique frequencies

In set Data Structure, emplace() method return a pair to newly inserted unique element and a value of true otherwise false. And Set store the unique values.

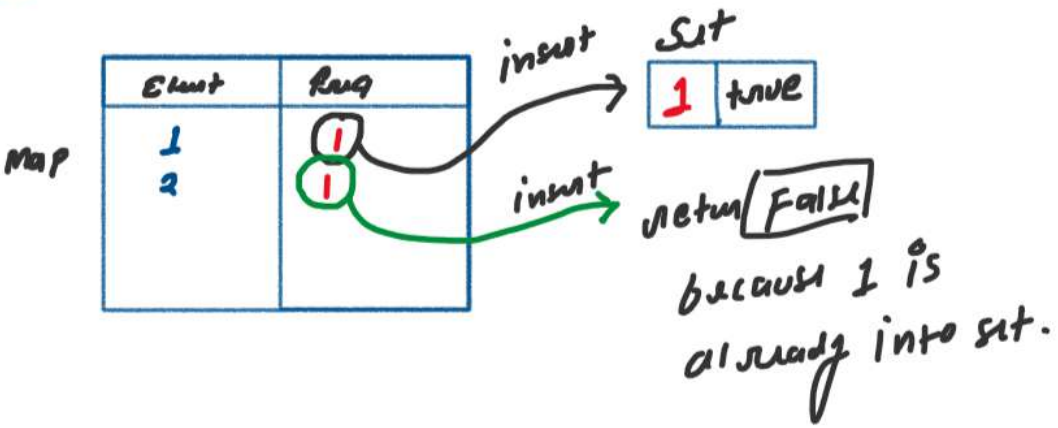
Resource: https://cplusplus.com/reference/set/set/

Ex2

arr

1	2
0	1

DRY RUN STEP2



STEP1 frequency counting using map

Map

elemt	freq
1	1
2	1

STEP2 checking for unique frequencies

```cpp
class Solution {
public:
    bool uniqueOccurrences(vector<int>& arr) {

        // Step 1: Frequency Counting using map
        unordered_map<int,int>mp;
        for(auto it:arr){
            mp[it]++;
        }

        // Step 2: Checking for Unique Frequencies
        unordered_set<int>st;
        for(auto it:mp){
            auto temp=st.emplace(it.second);
            if(!temp.second){
                return false;
            }
        }
        return true;
    }
};
```

Optimal Approach

Time complexity: O(N+M) = O(N)
Space complexity: O(N)

Where N is number of elements in input array, and M is the number of unique frequencies in the map.

Note 1: *In the worst case, where all elements are unique, the space complexity can be O(n) for unordered_map*
Note 2: *In the worst case, where all frequencies are unique, the space complexity can be O(n) for unordered_set*