Flip Columns For Maximum Number Of Equal Rows

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	1072
tags	Bit Manipulation Hash Map
language	C++
solved on	@22/11/2024
link	https://leetcode.com/problems/flip-columns-for-maximum-number-of-equal-rows/description/
Completion	☑

Intuition

The problem revolves around identifying rows in the matrix that can be made equal by flipping columns. A key observation is that for two rows to be equal after flipping, their relative pattern of 0s and 1s must match. This can be represented as a string where rows with the same pattern (either directly or after flipping) will be identical.

Approach

1. **Normalize Rows**:
   - For a given row, create a string representation where the first element is treated as the baseline (either 0 or 1).
   - Convert each element based on whether it matches the baseline (0 -> `0`, otherwise `1`).
2. **Hash and Count**:
   - Use a hash map to count the frequency of normalized row patterns.
   - The key is the normalized string representation of the row, and the value is the count of rows with the same pattern.
3. **Maximize Answer**:
   - Track the maximum frequency in the hash map, which corresponds to the maximum number of rows that can be made equal.

Complexity

Time Complexity:

- **Normalization**: For each row of size `m`, generating its normalized string takes **O(m)**.
- **Hashing and Counting**: Iterating through `n` rows takes . Thus, total complexity is **O(n m)**, where **n** is the number of rows and m is the number of columns.

Space Complexity:

- **Hash Map**: Stores at most unique patterns, each of size . Total space is **O(n m)**

Code

```
class Solution {
    string vec2str(vector<int>& v){
        string answer(v.size(),' ');
        for(int i = 0; i < v.size(); i++){
            answer[i] = (v[i]==v[0]) +'0';
        }
        return answer;
    }
public:
    int maxEqualRowsAfterFlips(vector<vector<int>>& matrix) {
        int answer = 1;
        unordered_map<string,int> mp;
        for(auto& row: matrix)
            answer = max(answer, ++mp[vec2str(row)]);

        return answer;
    }
};
```