1072. Flip Columns For Maximum Number of Equal Rows

You are given an `m x n` binary matrix `matrix`.
You can choose any number of columns in the matrix and flip every cell in that column (i.e., Change the value of the cell from `0` to `1` or vice versa).

Return *the maximum number of rows that have all values equal after some number of flips*.

Example 1:

```
Input: matrix = [[0,1],[1,1]]
Output: 1
Explanation: After flipping no values, 1 row has all values equal.
```

Example 2:

```
Input: matrix = [[0,1],[1,0]]
Output: 2
Explanation: After flipping values in the first column, both rows have equal
values.
```

Example 3:

```
Input: matrix = [[0,0,0],[0,0,1],[1,1,0]]
Output: 2
Explanation: After flipping values in the first two columns, the last two rows have
equal values.
```

Constraints:

- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 300`
- `matrix[i][j]` is either `0` or `1`.

1072. Flip Columns For Maximum Number of Equal Rows

```
/*
    Brute Force
    Time compelxity: O(m*(n+m))=O(m.n+m^2)
    Space complexity: O(n)
*/
class Solution {
public:
    int maxEqualRowsAfterFlips(std::vector<std::vector<int>>& matrix) {
        int m=matrix.size();
        int n=matrix[0].size();

        int ans=0;
        for(auto& cur_row: matrix){
            std::vector<int> flipped_row(n);
            for(int i=0;i<n;++i) flipped_row[i]=1-cur_row[i];
            int count_identical_rows=0;
            for(auto& compare_row: matrix){
                if(cur_row==compare_row || flipped_row==compare_row) count_identical_rows++;
            }
            ans=std::max(ans,count_identical_rows);
        }
        return ans;
    }
};
```

1072. Flip Columns For Maximum Number of Equal Rows

```
/*
    Hash map
    Time compelxity: O(m*n+m)
    Space complexity: O(n)
*/
class Solution {
public:
    int maxEqualRowsAfterFlips(std::vector<std::vector<int>>& matrix) {
        int m=matrix.size();
        int n=matrix[0].size();

        std::unordered_map<std::string,int> count;

        for(auto& cur_row: matrix){
            std::string s="";
            for(int i=1;i<n;++i) s+=((cur_row[0]-'0')^(cur_row[i]-'0'))+'0';
            count[s]++;
        }
        int ans=0;
        for(auto& [s,cnt]: count) ans=std::max(ans,cnt);
        return ans;
    }
};
```