Problem 1947: Maximum Compatibility Score Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 63.98%
Paid Only: No
Tags: Array, Dynamic Programming, Backtracking, Bit Manipulation, Bitmask

Problem Description

There is a survey that consists of `n` questions where each question's answer is either `0` (no) or `1` (yes).

The survey was given to `m` students numbered from `0` to `m - 1` and `m` mentors numbered from `0` to `m - 1`. The answers of the students are represented by a 2D integer array `students` where `students[i]` is an integer array that contains the answers of the `ith` student (**0-indexed**). The answers of the mentors are represented by a 2D integer array `mentors` where `mentors[j]` is an integer array that contains the answers of the `jth` mentor (**0-indexed**).

Each student will be assigned to **one** mentor, and each mentor will have **one** student assigned to them. The **compatibility score** of a student- mentor pair is the number of answers that are the same for both the student and the mentor.

* For example, if the student's answers were `[1, _0_ , _1_]` and the mentor's answers were `[0, _0_ , _1_]`, then their compatibility score is 2 because only the second and the third answers are the same.

You are tasked with finding the optimal student-mentor pairings to **maximize** the**sum of the compatibility scores**.

Given `students` and `mentors`, return _the**maximum compatibility score sum** that can be achieved._

Example 1:

Input: students = [[1,1,0],[1,0,1],[0,0,1]], mentors = [[1,0,0],[0,0,1],[1,1,0]] **Output:** 8 **Explanation:** We assign students to mentors in the following way: - student 0 to mentor 2 with a compatibility score of 3. - student 1 to mentor 0 with a compatibility score of 2. - student 2 to mentor 1 with a compatibility score of 3. The compatibility score sum is 3 + 2 + 3 = 8.

Example 2:

Input: students = [[0,0],[0,0],[0,0]], mentors = [[1,1],[1,1],[1,1]] **Output:** 0 **Explanation:** The compatibility score of any student-mentor pair is 0.

Constraints:

* `m == students.length == mentors.length` * `n == students[i].length == mentors[j].length` * `1 <= m, n <= 8` * `students[i][k]` is either `0` or `1`. * `mentors[j][k]` is either `0` or `1`.

Code Snippets

C++:

```
class Solution {
public:
int maxCompatibilitySum(vector<vector<int>>& students, vector<vector<int>>&
mentors) {

}
};
```

Java:

```
class Solution {
public int maxCompatibilitySum(int[][] students, int[][] mentors) {

}
}
```

Python3:

```
class Solution:
def maxCompatibilitySum(self, students: List[List[int]], mentors:
List[List[int]]) -> int:
```