发信人：Sempr (偶尔郁闷|(frk)styc's Fans No.1), 信区：Algorithm
标　题：Re: 翻转棋盘
发信站：水木社区（Sun Apr 20 10:39:18 2008），站内

http://www.topcoder.com/tc?module=Static&d1=match_editorials&d2=tco08qualRd3

最底下

EvenOnes

Used as: Division One - Level Three:
Value 1000
Submission Rate 175 / 1040 (16.83%)
Success Rate 122 / 175 (69.71%)
High Score nevenastan for 940.57 points (7 mins 13 secs)
Average Score 650.07 (for 122 correct submissions)

This problem involved very little coding, as the main challenge to overcome was figuring out the mathematical logic behind it. The first thing we may wonder is why the input always consists of an odd number of rows and columns. One important property that this restriction gives us is that if we, say, perform a move on a particular row, then the parity of the ones and zeros in that row is always changed, and the same holds for performing a move on a particular column. When we talk about the parity of ones or zeros, we mean the property that the number of ones or zeros in a row or column is even or odd. From this property, we have two important observations:

If we perform a move on row i, the parity of ones in row i is changed, and the parity of ones and zeros in every column is changed.
If we perform a move on column j, the parity of ones in column j is changed, and the parity of ones and zeros in every row is changed.
If we perform an even number of moves on the set of rows, then the parity of ones and zeros in every column remains unchanged, otherwise it is changed.
If we perform an even number of moves on the set of columns, then the parity of ones and zeros in every row remains unchanged, otherwise it is changed.
Now, let's define the number of rows with an even number of ones as Gr (good rows), the number of rows with an odd number of ones as Br (bad rows), the number of columns with an even number of ones as Gc (good columns), and the number of columns with an odd number of ones as Bc (bad columns). Also assume that there are N rows and M columns in our grid. We have two possible ways of making the partity of ones in the rows and columns to be even:

First, we can perform moves on every bad row. If Br is even, then after performing our moves on the bad rows, the parity of ones in all the columns remains constant. Now we complete our task by performing moves on every bad column, and the total number of moves required is Br + Bc. However, if Br is odd, then the partity of ones in all the columns will change after performing the moves on the bad rows. Thus, all the good columns become bad columns, and vice-versa. Therefore we must perform moves on all the originally good columns, and the total number of moves required is Br + Gc.

The other possible solution is to perform moves on every good row. If Gr is even, then after performing our moves on the good rows, the parity of ones in all the columns remains constant similar to the above. We again complete the task by performing moves on every bad column, and the total number of moves required is Gr + Bc. However, if Gr is odd, then the partity of ones in all the columns will change after performing the moves on the good rows, so all the good columns become bad columns, and vice-versa. Therefore we must perform moves on all the good columns, and the total number of moves required is Gr + Gc.

Of course, there may be situations where after performing the required operations on the chosen rows and columns, that the resulting grid could still have some row or column with an odd number of ones. However, it can be shown that this situation can never happen, so we will never return -1. The proof of this fact is left as an exercise to the reader (hint: consider a case-by-case analysis and a proof by contradiction). A sample C++ solution to this problem follows:

```
int EvenOnes::minOperations(vector <string> mat) {
    int badrow = 0, badcol = 0;
    int n = mat.size(); int m = mat[0].size();

    //get number of bad rows
    for (int i = 0; i < n; i++) {
        int ones = 0;
        for (int j = 0; j < m; j++) if (mat[i][j]=='1') ones++;
        if (ones % 2 == 1) badrow++;
   }

    //get number of bad cols
    for (int i = 0; i < m; i++) {
        int ones = 0;
        for (int j = 0; j < n; j++) if (mat[j][i] == '1') ones++;
        if (ones % 2 == 1) badcol++;
   }
    int a = 0, b = 0;
    //choose bad rows
    a = badrow;
    if (badrow % 2 == 1) a += m - badcol;
    else a += badcol;

    //choose good rows
    b = n - badrow;
    if ((n - badrow) % 2 == 1) b += m - badcol;
    else b += badcol;

    return min(a, b);
}
```

【 在 speedfirst (菜牛) 的大作中提到: 】
: 大家讨论下吧。程序员的算法题。
: 题目：翻转棋盘
: 一个N*M大小的棋盘，每个格子都是0或1，N和M都是奇数。你每次可以选择反转一行或
者一列，被反转的行或列的所有0变成1，所有1变成0。
: ....

--
淘宝，淘我喜欢！！
www.taobao.com

※ 来源:·水木社区 newsmth.net·[FROM: 121.0.31.*]