codility

Check out Codility training tasks

TASKS DETAILS

MEDIUM

1. **FlippingMatrix**
A matrix of binary values is given. We can flip the values in selected columns. What is the maximum number of rows that we can obtain that contain all the same values?

Task Score	Correctness	Performance
26%	18%	33%

Task description

Matrix A, consisting of N rows and M columns, is given, with each cell containing the value 0 or 1. Rows are numbered from 0 to N−1 (from top to bottom). Columns are numbered from 0 to M−1 (from left to right). The values inside the matrix can be changed: you can select as many columns as you want, and in the selected column(s), every value will be flipped (from 0 to 1, or from 1 to 0).

The goal is to obtain the maximum number of rows whose contents are all the same value (that is, we count rows with all 0s and rows with all 1s).

Write a function:

```
def solution(A)
```

that, given matrix A, returns the maximum number of rows containing all the same values that can be obtained after flipping the selected columns.

Examples:

1. Given matrix A with N = 3 rows and M = 4 columns:

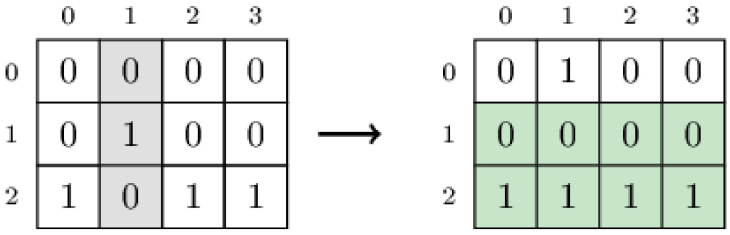
Solution

Programming language used:	Python
Total time used:	6 minutes
Effective time used:	6 minutes
Notes:	*not defined yet*

Task timeline

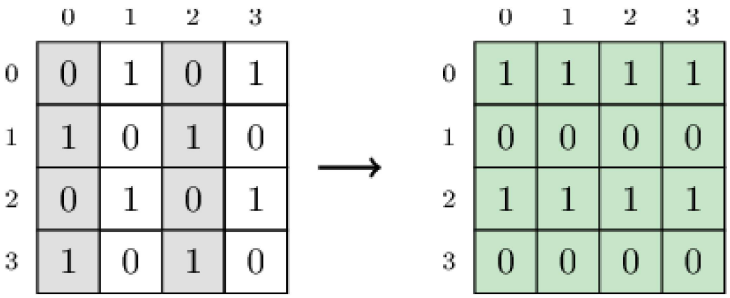
19:52:20 19:58:00

Code: 19:58:00 UTC, py, final, score: **26** show code in pop-up



the function should return 2. After flipping the values in column 1, the two last rows contain all equal values. Row 1 contains all 0s and row 2 contains all 1s.

2. Given matrix A with N = 4 rows and M = 4 columns:



the function should return 4. After flipping the values in two of the columns (columns 0 and 2), all the rows have the same value. Rows number 0 and 2 contain all 1s, and rows number 1 and 3 contain all 0s.

Write an **efficient** algorithm for the following assumptions:

- N and M are integers within the range [1..100,000];
- N * M is not greater than 100,000.


```
def dm(mat) :
    N = len(mat)
    for i in range(0, N):
        for j in range(0, N) :
            if ((i != j) and
             (mat[i][j] != 0)) :
                return False

    return True
def solution(A):
    c = 0
    n = len(A)
    m = len(A[0])
    if n==1 or m==1:
        return n
    if dm(A):
        return 1
    i = 0
    while i<n-1:
        if len(set(A[i])) <= 1:
            i = i+1
            continue
        j = i+1
        f=0
        while j<n:
            l=[]
            l.extend(A[i])
            l.extend(A[j])
            #print(l)
            if l.count(1) == l.count(0):
                c=c+2
                f = 1
                break
            j = j+1
        if f:
            i = i+2
        else:
            i = i+1
    return c
```

Analysis summary

The following issues have been detected: wrong answers, runtime errors, timeout errors.

For example, for the input `[[1, 1], [1, 0]]` the solution returned a wrong answer (got 0 expected 1).

Analysis

expand all	Example tests	
▶	example_1 First example.	✓ OK
▶		

example_2 Second example.		✓ OK
expand all	Correctness tests	
▶	one_row 1 row, 5 columns.	✓ OK
▶	one_column 5 rows, 1 column.	✓ OK
▶	2_x_2 2 row, 2 columns.	✗ WRONG ANSWER got 0 expected 1
▶	small_diagonal Each row contains no more than one occurrence of 1.	✗ WRONG ANSWER got 0 expected 1
▶	small_random Random matrix, NM <= 40.	✗ WRONG ANSWER got 0 expected 1
▶	medium_diagonal Each row contains no more than one occurrence of 1, NM <= 1,600.	✗ WRONG ANSWER got 0 expected 1
▶	medium_random Random matrix, NM <= 2,500.	✗ WRONG ANSWER got 42 expected 1
expand all	Performance tests	
▶	large_diagonal Each row contains no more than one occurrence of 1, NM <= 100,000.	✗ TIMEOUT ERROR running time: 0.696 sec., time limit: 0.592 sec.
▶	large_random Random matrix, NM <= 100,000.	✗ WRONG ANSWER got 278 expected 1
▶	large_one_row 1 row, 100,000 columns.	✓ OK
▶	large_one_column 100,000 row, 1 column.	✓ OK