Check out Codility training tasks

Candidate Report: Anonymous

Test Name:

Summary Timeline

Test Score

82 out of 100 points

82%

Tasks in Test

	Time Spent	Task Score
FlippingMatrix Submitted in: Python	1 min	82%

TASKS DETAILS

		Task Score	Correctness	Performance
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?	82%	100%	66%

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).

Solution

Programming language used:	Python
Total time used:	1 minutes
Effective time used:	1 minutes
Notes:	*not defined yet*

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:

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 1 & 1 \end{array} \longrightarrow \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 1 \end{array}$$

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:

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 0 & 1 \\ 3 & 1 & 0 & 1 & 0 \end{array} \longrightarrow \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 1 \\ 3 & 0 & 0 & 0 & 0 \end{array}$$

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.

Task timeline

23:29:02

23:29:19

Code: 23:29:19 UTC, py, final, score: **82**

show code in pop-up

```
# you can write to stdout for debugging pur
# print("this is a debug message")
def is_opposite(A, B):
    for i in range(len(A)):
        if A[i]==B[i]:
            return False
    return True

def solution(A):
    # write your code in Python 3.6
    max_rows = 0
    rows = len(A)
    cols = len(A[0])
    for row in range(rows):
        count = 0
        for i in range(rows-row):
            if A[row]==A[i+row] or is_oppos
                count+=1
        if count>max_rows:
            max_rows = count
        if max_rows==(rows-row):
            break
    return max_rows
```

Analysis summary

The following issues have been detected: timeout errors.

Analysis

Detected time complexity: **O(N*M*log(N+M) or O(N*M) or O(N**2*M)**

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.	✔ OK
▶	small_diagonal Each row contains no more than one occurrence of 1.	✔ OK
▶	small_random Random matrix, NM <= 40.	✔ OK
▶	medium_diagonal Each row contains no more than one occurrence of 1, NM <= 1,600.	✔ OK
▶	medium_random Random matrix, NM <= 2,500.	✔ OK

expand all	Performance tests	
▶	large_diagonal Each row contains no more than one occurrence of 1, NM <= 100,000.	✔ OK
▶	large_random Random matrix, NM <= 100,000.	✘ TIMEOUT ERROR Killed. Hard limit reached: 6.000 sec.
▶	large_one_row 1 row, 100,000 columns.	✔ OK
▶	large_one_column 100,000 row, 1 column.	✔ OK