codility

Check out Codility training tasks

# Candidate Report: Anonymous

Test Name:

Summary Timeline

## Test Score

100 out of 100 points

100%

## Tasks in Test

| | Time Spent | Task Score |
|---|---|---|
| FlippingMatrix<br>Submitted in: Python | 1 min | 100% |

## 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 |
|---|---|---|
| 100% | 100% | 100% |

## 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* |

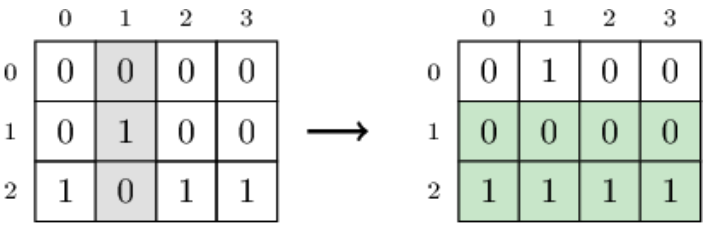
Write a function:

```
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.

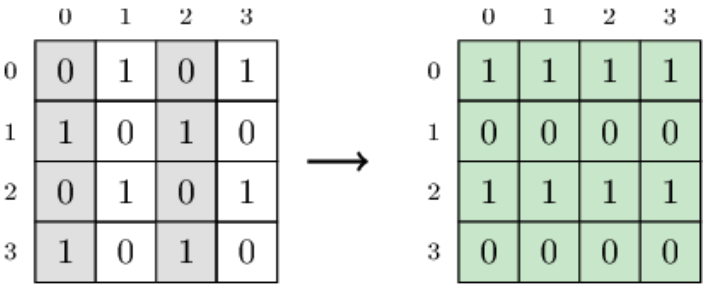
**Examples:**

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



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.



## Task timeline

00:13:19 00:13:39

Code: 00:13:39 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging pur
2  # print("this is a debug message")
3  def get_opposite(A):
4      result = ''
5      for item in A:
6          if item=='0':
7              result+='1'
8          else:
9              result+='0'
10     return result
11
12 def solution(A):
13     # write your code in Python 3.6
14     max_rows = 0
15     rows = len(A)
16     hash_table = {}
17     for row in A:
18         row_str = ''.join(str(e) for e in r
19         # print(row_str, get_opposite(row_s
20         if row_str in hash_table:
21             hash_table[row_str] += 1
22         elif get_opposite(row_str) in hash_
23             hash_table[get_opposite(row_str
24         else:
25             hash_table[row_str] = 1
26     for key in hash_table:
27         if hash_table[key] > max_rows:
28             max_rows = hash_table[key]
29     return max_rows
```

## Analysis summary

The solution obtained perfect score.

## Analysis

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

| expand all | Example tests | |
|---|---|---|
| ▶ example_1<br>First example. | | ✔ OK |

▶ example_2 ✔ OK
Second example.

expand all Correctness tests

▶ one_row ✔ OK
1 row, 5 columns.

▶ one_column ✔ OK
5 rows, 1 column.

▶ 2_x_2 ✔ OK
2 row, 2 columns.

▶ small_diagonal ✔ OK
Each row contains no more than one occurrence of 1.

▼ small_random ✔ OK
Random matrix, NM <= 40.

1. 0.036 s OK
2. 0.036 s OK
3. 0.036 s OK

▼ medium_diagonal ✔ OK
Each row contains no more than one occurrence of 1, NM <= 1,600.

1. 0.040 s OK
2. 0.036 s OK
3. 0.036 s OK
4. 0.036 s OK

▼ medium_random ✔ OK
Random matrix, NM <= 2,500.

1. 0.040 s OK
2. 0.036 s OK
3. 0.040 s OK

collapse all Performance tests

▼ large_diagonal ✔ OK
Each row contains no more than one occurrence of 1, NM <= 100,000.

1. 0.152 s OK
2. 0.160 s OK

| | | |
|---|---|---|
| 3. | 0.180 s | OK |
| 4. | 0.164 s | OK |
| 5. | 0.180 s | OK |

▼ large_random — ✔ OK
Random matrix, NM <= 100,000.

| | | |
|---|---|---|
| 1. | 0.160 s | OK |
| 2. | 0.168 s | OK |
| 3. | 0.200 s | OK |

▼ large_one_row — ✔ OK
1 row, 100,000 columns.

| | | |
|---|---|---|
| 1. | 0.176 s | OK |

▼ large_one_column — ✔ OK
100,000 row, 1 column.

| | | |
|---|---|---|
| 1. | 0.404 s | OK |