Codeforces Round #663 (Div. 2)

A. Suborrays

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2,3,1,5,4]$ is a permutation, but $[1,2,2]$ is not a permutation (2 appears twice in the array) and $[1,3,4]$ is also not a permutation ($n=3$ but there is 4 in the array).

For a positive integer n, we call a permutation p of length n **good** if the following condition holds for every pair i and j ($1 \le i \le j \le n$) —

- $(p_i \text{ OR } p_{i+1} \text{ OR } \ldots \text{ OR } p_{j-1} \text{ OR } p_j) \ge j-i+1$, where OR denotes the bitwise OR operation.

In other words, a permutation p is **good** if for every subarray of p, the OR of all elements in it is not less than the number of elements in that subarray.

Given a positive integer n, output any **good** permutation of length n. We can show that for the given constraints such a permutation always exists.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first and only line of every test case contains a single integer n ($1 \le n \le 100$).

Output
For every test, output any **good** permutation of length n on a separate line.

Example
input
```
3
1
3
7
```
output
```
1
3 1 2
4 3 5 2 7 1 6
```

Note
For $n=3$, $[3,1,2]$ is a good permutation. Some of the subarrays are listed below.

- $3 \text{ OR } 1 = 3 \ge 2$ $(i=1, j=2)$
- $3 \text{ OR } 1 \text{ OR } 2 = 3 \ge 3$ $(i=1, j=3)$
- $1 \text{ OR } 2 = 3 \ge 2$ $(i=2, j=3)$
- $1 \ge 1$ $(i=2, j=2)$

Similarly, you can verify that $[4,3,5,2,7,1,6]$ is also good.

B. Fix You

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a conveyor belt represented using a grid consisting of n rows and m columns. The cell in the i-th row from the top and the j-th column from the left is labelled (i,j).

Every cell, except (n,m), has a direction R (Right) or D (Down) assigned to it. If the cell (i,j) is assigned direction R, any luggage kept on that will move to the cell $(i,j+1)$. Similarly, if the cell (i,j) is assigned direction D, any luggage kept on that will move to the cell $(i+1,j)$. If at any moment, the luggage moves out of the grid, it is considered to be lost.

There is a counter at the cell (n,m) from where all luggage is picked. A conveyor belt is called **functional** if and only if any luggage reaches the counter regardless of which cell it is placed in initially. More formally, for every cell (i, j), any luggage placed in this cell should eventually end up in the cell (n, m).

This may not hold initially; you are, however, allowed to **change** the directions of some cells to make the conveyor belt functional. Please determine the minimum amount of cells you have to change.

Please note that it is always possible to make any conveyor belt functional by changing the directions of some set of cells.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10$). Description of the test cases follows.

The first line of each test case contains two integers n, m ($1 \le n \le 100$, $1 \le m \le 100$) — the number of rows and columns, respectively.

The following n lines each contain m characters. The j-th character in the i-th line, $a_{i,j}$ is the initial direction of the cell (i, j). Please note that $a_{n,m} =$ C.

Output

For each case, output in a new line the minimum number of cells that you have to change to make the conveyor belt functional.

Example

input

```
4
3 3
RRD
DDR
RRC
1 4
DDDC
6 9
RDDDDDRRR
RRDDRRDDD
RRDRDRRDR
DDDDRDDRR
DRRDRDDDR
DDRDRRDDC
1 1
C
```

output

```
1
3
9
0
```

Note

In the first case, just changing the direction of $(2, 3)$ to D is enough.

You can verify that the resulting belt is functional. For example, if we place any luggage at $(2, 2)$, it first moves to $(3, 2)$ and then to $(3, 3)$.

In the second case, we have no option but to change the first 3 cells from D to R making the grid equal to RRRC.

C. Cyclic Permutations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Consider a permutation p of length n, we build a graph of size n using it as follows:

- For every $1 \le i \le n$, find the **largest** j such that $1 \le j < i$ and $p_j > p_i$, and add an undirected edge between node i and node j
- For every $1 \le i \le n$, find the **smallest** j such that $i < j \le n$ and $p_j > p_i$, and add an undirected edge between node i and node j

In cases where no such j exists, we make no edges. Also, note that we make edges between the corresponding indices, not the values at those indices.

For clarity, consider as an example $n = 4$, and $p = [3, 1, 4, 2]$; here, the edges of the graph are $(1, 3), (2, 1), (2, 3), (4, 3)$.

A permutation p is **cyclic** if the graph built using p has at least one simple cycle.

Given n, find the number of cyclic permutations of length n. Since the number may be very large, output it modulo $10^9 + 7$.

Please refer to the Notes section for the formal definition of a simple cycle

Input

The first and only line contains a single integer n ($3 \le n \le 10^6$).

Output

Output a single integer $0 \le x < 10^9 + 7$, the number of cyclic permutations of length n modulo $10^9 + 7$.

Examples

input

```
4
```

output

```
16
```

input

```
583291
```

output

```
135712853
```

Note

There are 16 cyclic permutations for $n = 4$. $[4, 2, 1, 3]$ is one such permutation, having a cycle of length four: $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 4$
.

Nodes v_1, v_2, $\ldots$, v_k form a simple cycle if the following conditions hold:

- $k \geq 3$.
- $v_i \neq v_j$ for any pair of indices i and j. ($1 \leq i < j \leq k$)
- v_i and v_{i+1} share an edge for all i ($1 \leq i < k$), and v_1 and v_k share an edge.

D. 505

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A binary matrix is called **good** if every **even** length square sub-matrix has an **odd** number of ones.

Given a binary matrix a consisting of n rows and m columns, determine the minimum number of cells you need to change to make it good, or report that there is no way to make it good at all.

All the terms above have their usual meanings — refer to the Notes section for their formal definitions.

Input

The first line of input contains two integers n and m ($1 \leq n \leq m \leq 10^6$ and $n \cdot m \leq 10^6$) — the number of rows and columns in a, respectively.

The following n lines each contain m characters, each of which is one of `0` and `1`. If the j-th character on the i-th line is `1`, then $a_{i,j} = 1$. Similarly, if the j-th character on the i-th line is `0`, then $a_{i,j} = 0$.

Output

Output the minimum number of cells you need to change to make a good, or output -1 if it's not possible at all.

Examples

input

```
3 3
101
001
110
```

output

```
2
```

input

```
7 15
000100001010010
100111010110001
101101111100100
010000111111010
111010010100001
000011001111101
111111011010011
```

output

```
-1
```

Note

In the first case, changing $a_{1,1}$ to 0 and $a_{2,2}$ to 1 is enough.

You can verify that there is no way to make the matrix in the second case good.

Some definitions —

- A binary matrix is one in which every element is either 1 or 0.
- A sub-matrix is described by 4 parameters — r_1, r_2, c_1, and c_2; here, $1 \le r_1 \le r_2 \le n$ and $1 \le c_1 \le c_2 \le m$.
- This sub-matrix contains all elements $a_{i,j}$ that satisfy both $r_1 \le i \le r_2$ and $c_1 \le j \le c_2$.
- A sub-matrix is, further, called an even length square if $r_2 - r_1 = c_2 - c_1$ and $r_2 - r_1 + 1$ is divisible by 2.

E. Pairs of Pairs

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a simple and connected undirected graph consisting of n nodes and m edges.

Consider any way to pair some subset of these n nodes such that no node is present in more than one pair.

This pairing is **valid** if for every pair of pairs, the induced subgraph containing all 4 nodes, two from each pair, has at most 2 edges (out of the 6 possible edges). More formally, for any two pairs, (a, b) and (c, d), the induced subgraph with nodes $\{a, b, c, d\}$ should have at most 2 edges.

Please note that the subgraph induced by a set of nodes contains nodes only from this set and edges which have both of its end points in this set.

Now, do one of the following:

- Find a simple path consisting of at least $\lceil \frac{n}{2} \rceil$ nodes. Here, a path is called simple if it does not visit any node multiple times.
- Find a valid pairing in which at least $\lceil \frac{n}{2} \rceil$ nodes are paired.

It can be shown that it is possible to find at least one of the two in every graph satisfying constraints from the statement.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

The first line of each test case contains 2 integers n, m ($2 \le n \le 5 \cdot 10^5$, $1 \le m \le 10^6$), denoting the number of nodes and edges, respectively.

The next m lines each contain 2 integers u and v ($1 \le u, v \le n$, $u \neq v$), denoting that there is an undirected edge between nodes u and v in the given graph.

It is guaranteed that the given graph is connected, and simple — it does not contain multiple edges between the same pair of nodes, nor does it have any self-loops.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$.

It is guaranteed that the sum of m over all test cases does not exceed 10^6.

Output

For each test case, the output format is as follows.

If you have found a pairing, in the first line output "PAIRING" (without quotes).

- Then, output k ($\lceil \frac{n}{2} \rceil \le 2 \cdot k \le n$), the number of pairs in your pairing.
- Then, in each of the next k lines, output 2 integers a and b — denoting that a and b are paired with each other. **Note that the graph does not have to have an edge between a and b!**
- This pairing has to be valid, and every node has to be a part of at most 1 pair.

Otherwise, in the first line output "PATH" (without quotes).

- Then, output k ($\lceil \frac{n}{2} \rceil \le k \le n$), the number of nodes in your path.
- Then, in the second line, output k integers, $v_1, v_2, \ldots, v_k$, in the order in which they appear on the path. Formally, v_i and v_{i+1} should have an edge between them for every i ($1 \le i < k$).
- This path has to be simple, meaning no node should appear more than once.

Example

input

```
4
```

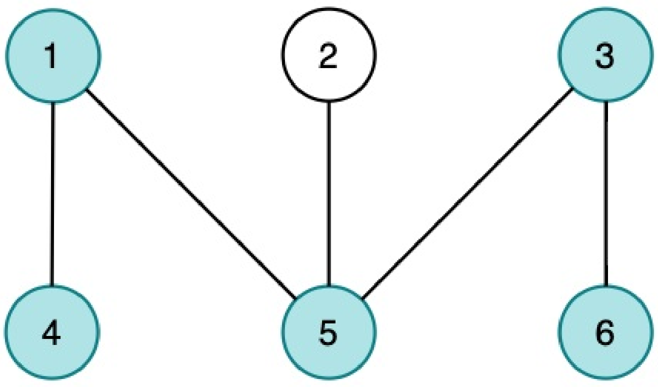
```
6 5
1 4
2 5
3 6
1 5
3 5
6 5
1 4
2 5
3 6
1 5
3 5
12 14
1 2
2 3
3 4
4 1
1 5
1 12
2 6
2 7
3 8
3 9
4 10
4 11
2 4
1 3
12 14
1 2
2 3
3 4
4 1
1 5
1 12
2 6
2 7
3 8
3 9
4 10
4 11
2 4
1 3
```

output

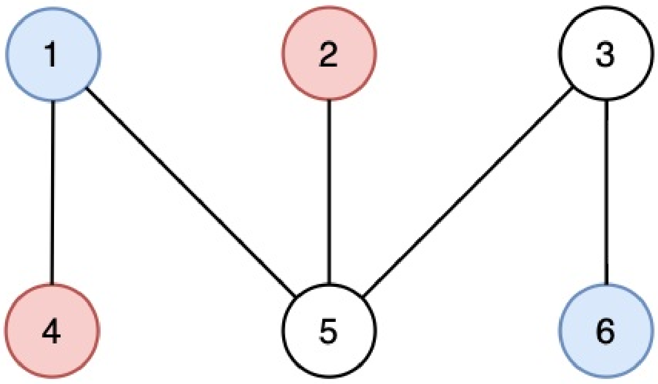
```
PATH
4
1 5 3 6
PAIRING
2
1 6
2 4
PAIRING
3
1 8
2 5
4 10
PAIRING
4
1 7
2 9
3 11
4 5
```

Note

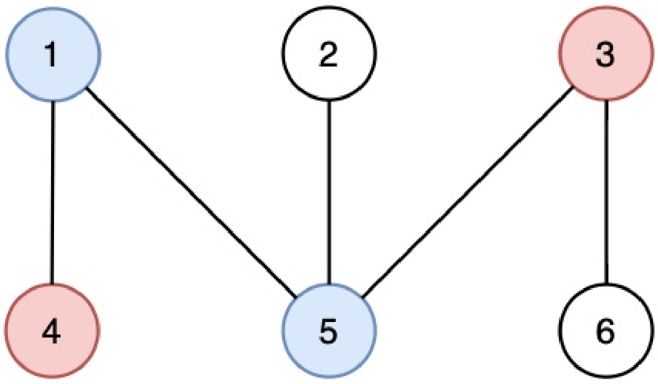
The path outputted in the first case is the following.



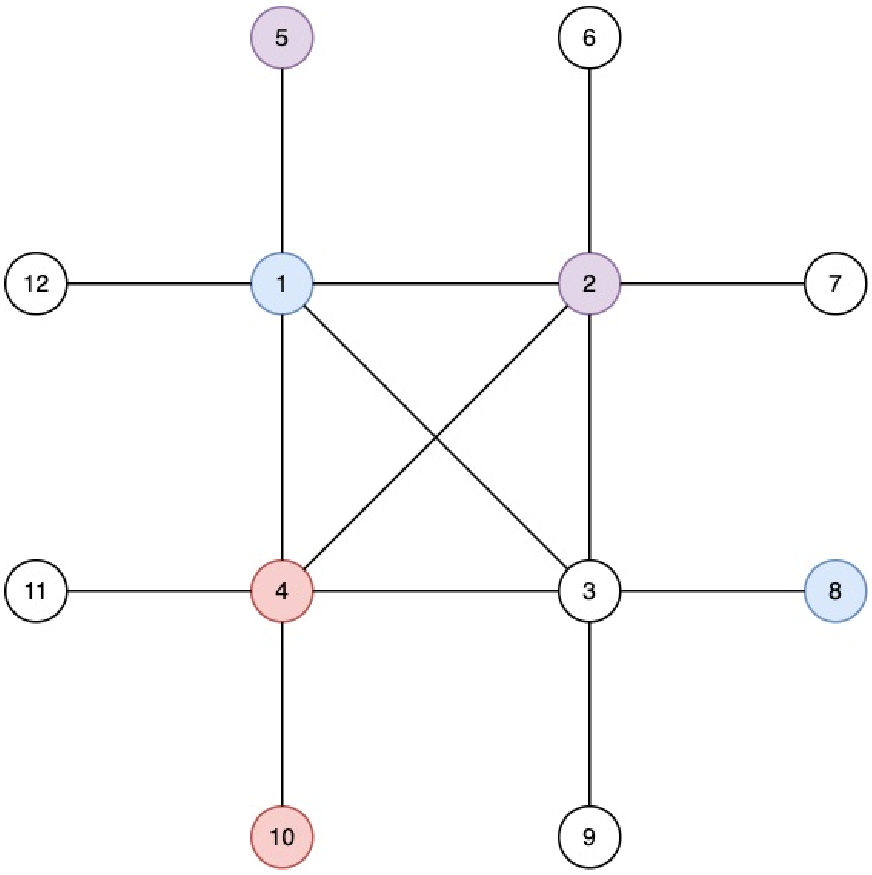
The pairing outputted in the second case is the following.



Here is an **invalid** pairing for the same graph — the subgraph $\{1,3,4,5\}$ has 3 edges.



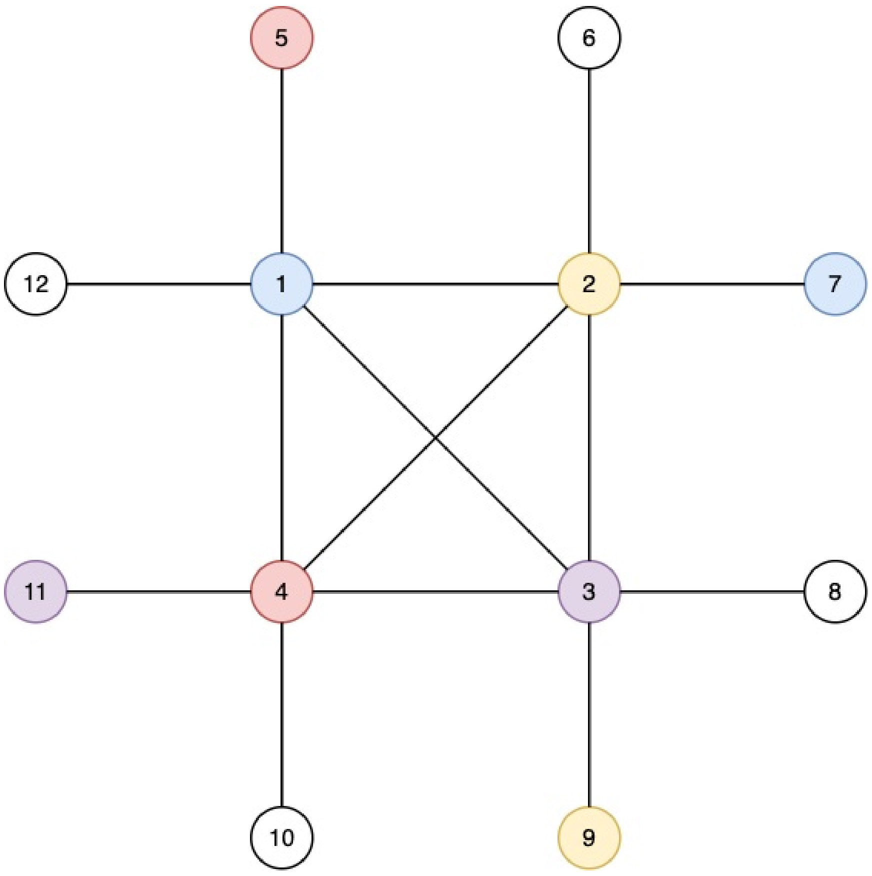
Here is the pairing outputted in the third case.



It's valid because —

- The subgraph $\{1,8,2,5\}$ has edges (1,2) and (1,5).
- The subgraph $\{1,8,4,10\}$ has edges (1,4) and (4,10).
- The subgraph $\{4,10,2,5\}$ has edges (2,4) and (4,10).

Here is the pairing outputted in the fourth case.


5
6
12
1
2
7
11
4
3
8
10
9