Day 6. Matchings

A. Bipartite graph

1 second, 256 megabytes

The bipartite graph is an undirected graph $\langle V, E\rangle$ whose vertices can be divided into two disjoint sets L and R ($L \cap R = \emptyset$ and $L \cup R = V$), such that for every edge $(u, v) \in E$, either $u \in L, v \in R$ or $v \in L, u \in R$.

You are given an undirected graph. Check if this graph is bipartite.

Input

The first line contains two integers n and m ($1 \le n \le 100\,000$; $0 \le m \le 200\,000$) — number of vertices and edges in the graph, respectively.

Next m lines contain the description of edges; i-th of these lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) — the vertices connected by i-th edge.

Output

Output "YES", if the given graph is bipartite, or "NO", otherwise.

If the graph is bipartite output the proper division into two sets: for each vertex output 1, if this vertex in set L, otherwise output 2.

input

```
4 4
1 2
1 3
2 4
4 2
```

output

```
YES
1 2 2 1
```

input

```
3 3
1 2
2 3
3 1
```

output

```
NO
```

B. Add Some Edges

2 seconds, 256 megabytes

You are given a simple connected undirected graph.

Your task is to find the maximum number of edges you can add to this graph, so the graph remains simple and bipartite.

Recall that the graph is called bipartite if its vertices can be divided into two sets U and V such that all edges of the graph connect some vertex from U and some vertex from V.

If the graph is already is not bipartite, print `-1`.

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of vertices in the graph and the number of edges in the graph, respectively.

The following m lines contain the edges: edge i is given as a pair of the vertices v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$). There are no multiple edges in the given graph, i.e. for each pair (v_i, u_i) there are no other pairs (v_i, u_i) and (u_i, v_i) in the list of edges.

It is guaranteed that the given graph is simple, connected and undirected.

Output

If the graph is already is not bipartite, print `-1`. Otherwise, print the maximum number of edges you can add to this graph, so the graph is simple and bipartite.

input

```
4 4
1 2
2 3
3 4
4 1
```

output

```
0
```

input

```
4 5
1 2
2 3
3 4
4 1
1 3
```

output

```
-1
```

C. Maximal by Inclusion Matching

2 seconds, 256 megabytes

You are given a bipartite graph (i.e. such an undirected graph that vertices can be split into two such sets that there are no edges between vertices of the same set). This graph has n_1 vertices in the first part and n_2 vertices in the second part. Edges in this graph only exist between vertices of different parts. The i-th edge connects the vertex x_i in the first part and the vertex y_i in the second part ($1 \le x_i \le n_1; 1 \le y_i \le n_2$).

Your task is to find maximal by inclusion matching in the given bipartite graph. Maximal by inclusion matching is a matching for which it is impossible to add an edge from the given graph preserving a matching property. I.e. a maximal by inclusion matching is not included in other matching.

Consider using faster input/output methods in this problem.

Input

The first line of the input contains three integers n_1, n_2 and m ($1 \le n_1, n_1 \le 10^5; 0 \le m \le min(5 \cdot 10^5; n_1 \cdot n_2)$) — the number of vertices in the first part of the graph, the number of vertices in the second part of the graph and the number of edges in the graph.

The next m lines describe edges of the graph. The i-th line contains two integers x_i and y_i ($1 \le x_i \le n_1; 1 \le y_i \le n_2$), where x_i is the vertex of the first part and y_i is the vertex of the second part.

It is guaranteed that all edges are distinct (i.e. all pairs (x_i, y_i) are distinct).

Output

In the first line, print one integer k ($0 \le k \le m$) — the size of the maximal by inclusion matching in the given graph. In the next k lines, print the maximal by inclusion matching itself. The i-th of these lines should contain the i-th edge of the found maximal by inclusion matching (a_i, b_i).

If there are several answers, you can print any.

Note that the size of the maximal by inclusion matching can be less than the size of the maximum matching.

input
```
2 2 3
1 1
1 2
2 2
```
output
```
2
1 1
2 2
```

input
```
5 5 12
4 5
3 1
1 3
4 3
1 1
2 1
2 5
4 4
1 2
2 4
3 2
5 1
```
output
```
4
4 5
3 1
1 3
2 4
```

input
```
5 6 8
1 2
4 4
1 3
5 3
2 6
3 3
4 6
2 3
```
output
```
4
1 2
4 4
5 3
2 6
```

D. Maximum Matching

1 second, 256 megabytes

You are given a bipartite graph (i.e. such an undirected graph that vertices can be split into two such sets that there are no edges between vertices of the same set). This graph has n_1 vertices in the first part and n_2 vertices in the second part. Edges in this graph only exist between vertices of different parts. The i-th edge connects the vertex x_i in the first part and the vertex y_i in the second part ($1 \le x_i \le n_1; 1 \le y_i \le n_2$).

Your task is to find the maximum matching in the given graph. Recall that the maximum matching is such a set of edges that there are no two edges in this set connected to the same vertex, and the size of this set is **maximum possible**.

Input

The first line of the input contains three integers n_1, n_2 and m ($1 \le n_1, n_2 \le 300$; $0 \le m \le n_1 \cdot n_2$) — the number of vertices in the first part of the graph, the number of vertices in the second part of the graph and the number of edges in the graph.

The next m lines describe edges of the graph. The i-th line contains two integers x_i and y_i ($1 \le x_i \le n_1; 1 \le y_i \le n_2$), where x_i is the vertex of the first part and y_i is the vertex of the second part.

It is guaranteed that all edges are distinct (i.e. all pairs (x_i, y_i) are distinct).

Output

In the first line, print one integer k ($0 \le k \le m$) — the size of the maximum matching in the given graph. In the next k lines, print the maximum matching itself. The i-th of these lines should contain the i-th edge of the found maximum matching (a_i, b_i).

If there are several answers, you can print any.

input
```
2 2 3
1 1
1 2
2 2
```
output
```
2
1 1
2 2
```

input
```
5 5 12
4 5
3 1
1 3
4 3
1 1
2 1
2 5
4 4
1 2
2 4
3 2
5 1
```
output
```
5
5 1
3 2
1 3
4 4
2 5
```

input
```
5 6 8
1 2
4 4
1 3
5 3
2 6
3 3
4 6
2 3
```

output

```
4
1 2
3 3
4 4
2 6
```

E. Compose a Word

1 second, 256 megabytes

You are given n cubes. The i-th cube has 6 letters written on it, one letter per cube face. The exact positions of letters doesn't matter in this problem, so each cube can be described with the string of length 6 consisting of lowercase Latin letters.

You are also given a word s of length m. Your task is to say if you can compose this word using the given cubes. You can choose exactly one letter in each cube, you can place cubes in any order you want, and it is not necessary to use all the given cubes at all.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 200$) — the number of cubes and the length of the given word, respectively.

The second line contains one string s, consisting of m lowercase Latin letters — the given word.

The next n lines describe cubes. The i-th line contains the string t_i consisting of 6 lowercase Latin letters — the description of the i-th cube.

Output

If it is impossible to compose the given word using the given cubes, print `NO` in the only line.

Otherwise, print `YES` in the first line and m **distinct** integers $p_1, p_2, \dots, p_m$ $(1 \le p_i \le n)$, where p_i is the index of the cube that should be used to obtain the i-th letter of the given word.

If there are several answers, you can print any.

input

```
4 3
ann
annnnn
bcdefg
hijklm
nopqrs
```

output

```
NO
```

input

```
6 5
helen
abcdef
ghijkl
mnopql
stuvwn
eiuozk
eiuozk
```

output

```
YES
2 5 3 1 4
```

F. Parquet Covering

5.0 s, 256 megabytes

There is a room that can be represented as a matrix of size $n \times m$. The room is covered with parquet, but it is pretty old, so some parts of the parquet are already broken.

Cells of the room where the parquet is **not broken** are marked with dots '`.`', and cells of the room where the parquet is **broken** are marked with stars '`*`'.

You have two possible opportunities: the first one is to buy a parquet piece of size 2×1 (or 1×2 if you rotate it) for a coins or buy a parquet piece of size 1×1 for b coins. Note that you can not break the parquet piece of size 2×1.

Your task is to cover **all** cells with the broken parquet for the minimum possible cost.

Input

The first line of the input contains four integers n, m, a and b ($1 \le n, m \le 300; 1 \le a, b \le 1000$) — the size of the room and prices of parquet pieces.

The next n lines contain m characters each. The j-th character in the i-th line is dot '`.`' if the parquet in the cell (i, j) of the room is **not broken** and star '`*`' otherwise.

Output

Print one integer — the minimum possible cost of covering **all** cells with the broken parquet using only pieces described in the problem statement.

input

```
2 3 3 2
.**
.*.
```

output

```
5
```

input

```
4 4 5 2
..*.
..*.
***.
....
```

output

```
10
```

G. Matching vs Independent Set

1 second, 256 megabytes

You are given a graph with $3 \cdot n$ vertices and m edges. You are to find a matching of n edges, **or** an independent set of n vertices.

A set of edges is called a matching if no two edges share an endpoint.

A set of vertices is called an independent set if no two vertices are connected with an edge.

Input

The first line contains a single integer $T \ge 1$ — the number of graphs you need to process. The description of T graphs follows.

The first line of description of a single graph contains two integers n and m, where $3 \cdot n$ is the number of vertices, and m is the number of edges in the graph ($1 \le n \le 10^5$, $0 \le m \le 5 \cdot 10^5$).

Each of the next m lines contains two integers v_i and u_i ($1 \le v_i, u_i \le 3 \cdot n$), meaning that there is an edge between vertices v_i and u_i.

It is guaranteed that there are no self-loops and no multiple edges in the graph.

It is guaranteed that the sum of all n over all graphs in a single test does not exceed 10^5, and the sum of all m over all graphs in a single test does not exceed $5 \cdot 10^5$.

Output

Print your answer for each of the T graphs. Output your answer for a single graph in the following format.

If you found a matching of size n, on the first line print "`Matching`" (without quotes), and on the second line print n integers — the indices of the edges in the matching. The edges are numbered from 1 to m in the input order.

If you found an independent set of size n, on the first line print "`IndSet`" (without quotes), and on the second line print n integers — the indices of the vertices in the independent set.

If there is no matching and no independent set of the specified size, print "`Impossible`" (without quotes).

You can print edges and vertices in any order.

If there are several solutions, print any. In particular, if there are both a matching of size n, and an independent set of size n, then you should print exactly one of such matchings **or** exactly one of such independent sets.

input

```
4
1 2
1 3
1 2
1 2
1 3
1 2
2 5
1 2
3 1
1 4
5 1
1 6
2 15
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 6
```

output

```
Matching
2
IndSet
1
IndSet
2 4
Matching
1 15
```

The first two graphs are same, and there are both a matching of size 1 and an independent set of size 1. Any of these matchings and independent sets is a correct answer.

The third graph does not have a matching of size 2, however, there is an independent set of size 2. Moreover, there is an independent set of size 5: `2 3 4 5 6`. However such answer is not correct, because you are asked to find an independent set (or matching) of size **exactly** n.

The fourth graph does not have an independent set of size 2, but there is a matching of size 2.

H. Matching of Maximum Weight

1 second, 256 megabytes

Given a tree of n vertices. Each edge has a weight. Matching is a set of edges without common vertices. For a given tree find the matching with the maximum total weight.

Input

The first line contains the integer n, the number of tree nodes ($1 \le n \le 10^5$). The next $n - 1$ lines describe the edges of the tree. Each line contains three integers u_i, v_i, w_i, the indices of the vertices that the edge connects, and its weight ($1 \le v_i, u_i \le n$, $1 \le w_i \le 1000$). It is guaranteed that the described graph is a tree.

Output

Print one number, the weight of maximal matching.

input

```
6
1 2 2
1 6 3
4 2 5
6 5 1
6 3 4
```

output

```
9
```

I. Move items

2 seconds, 512 megabytes

There are n objects on the plane and one should move them to n specified location. Each location should contain one object at the end. Each object has its moving speed, while one could move objects simultaneously.

Find the minimal time to move all objects in the desired manner.

Input

The first line of the input contains one integer n $(1 \le n \le 50)$ — the number of objects. Next n lines contain the descriptions of objects: coordinates x_i, y_i $(1 \le x_i, y_i \le 1000)$ and the maximal speed v_i $(1 \le v_i \le 10)$. The last n lines contain the coordinates a_i, b_i $(1 \le a_i, b_i \le 1000)$ of final locations.

Output

The sole line of the output should contain the smallest time to move all the objects. The error should not exceed 10^{-4}.

input

```
2
0 0 1
0 1 1
1 1
1 0
```

output

```
1.0
```

input

```
2
0 0 1
0 1 1
1 1
2 1
```

output

```
2.0
```

input

```
2
0 0 1
5 0 1
5 12
10 12
```

output

```
13.0
```

input

```
2
0 0 2
5 0 1
5 12
10 12
```

output

```
12.0
```

input

```
4
78 520 5
827 239 5
620 200 7
809 269 7
986 496
754 745
772 375
44 223
```

output

```
68.45242650102003
```

J. Subtree Sums

1 second, 256 megabytes

Given a rooted tree, vertex 1 is the root. Node i has weight w_i. For each node of the tree, find the total weight of all nodes in its subtree.

Input

The first line contains the integer n, the number of tree nodes ($1 \leq n \leq 10^5$). The second line contains n integers w_i ($1 \leq w_i \leq 1000$). The next $n - 1$ lines describe the edges of the tree. Each line contains two numbers, the indices of the vertices that the edge connects. It is guaranteed that the described graph is a tree.

Output

Print n numbers, the sum of weights in the subtree for each node.

input

```
6
2 3 1 1 3 1
1 2
1 6
4 2
6 5
6 3
```

output

```
11 4 1 1 3 5
```