

Exercise: Graph Theory Traversal and Shortest Paths

This document defines the lab for ["Algorithms – Fundamentals \(C#\)" course @ Software University](#).

Please submit your solutions (source code) of all below-described problems in [Judge](#).

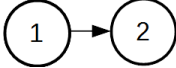
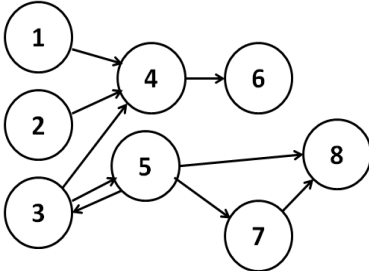
1. Distance Between Vertices

We are given a **directed graph**. We are given also a set of **pairs of vertices**. Find the **shortest distance between each pair** of vertices or -1 if there is no path connecting them.

On the first line, you will get **N**, the number of vertices in the graph. On the second line, you will get **P**, the number of pairs between which to find the shortest distance.

On the next **N** lines will be the edges of the graph and on the next **P** lines, the pairs.

Examples

Input	Picture	Output
2 2 1:2 2: 1-2 2-1		{1, 2} -> 1 {2, 1} -> -1
8 4 1:4 2:4 3:4 5 4:6 5:3 7 8 6: 7:8 8: 1-6 1-5 5-6 5-8		{1, 6} -> 2 {1, 5} -> -1 {5, 6} -> 3 {5, 8} -> 1

9		
8		
11:4		
4:12 1		
1:12 21 7		
7:21		
12:4 19		
19:1 21		
21:14 31		
14:14		
31:		
11-7		
11-21		
21-4		
19-14		
1-4		
1-11		
31-21		
11-14		
		{11, 7} -> 3 {11, 21} -> 3 {21, 4} -> -1 {19, 14} -> 2 {1, 4} -> 2 {1, 11} -> -1 {31, 21} -> -1 {11, 14} -> 4

Hint

For each pair use **BFS** to find all paths from the source to the destination vertex.

2. Areas in Matrix

We are given a matrix of letters of size $N * M$. Two cells are neighbors if they share a common wall. Write a program to find the connected areas of neighbor cells holding the same letter. Display the **total number of areas** and the number of **areas for each alphabetical letter** (ordered by alphabetical order).

On the **first line** is given the **number of rows**.

Examples

Input	Picture	Output
6 8 aaccbaac baaaaccc baabaccc bbdaaccc		Areas: 8 Letter 'a' -> 2 Letter 'b' -> 2 Letter 'c' -> 3 Letter 'd' -> 1

ccdcccc ccdcccc																																															
3 3 aaa aaa aaa	<table><tr><td>a</td><td>a</td><td>a</td></tr><tr><td>a</td><td>a</td><td>a</td></tr><tr><td>a</td><td>a</td><td>a</td></tr></table>	a	a	a	a	a	a	a	a	a	Areas: 1 Letter 'a' -> 1																																				
a	a	a																																													
a	a	a																																													
a	a	a																																													
5 9 asssaadas adsdasdad sdsdadsas sdasdsdsa ssssasddd	<table><tr><td>a</td><td>s</td><td>s</td><td>s</td><td>a</td><td>a</td><td>d</td><td>a</td><td>s</td></tr><tr><td>a</td><td>d</td><td>s</td><td>d</td><td>a</td><td>s</td><td>d</td><td>a</td><td>d</td></tr><tr><td>s</td><td>d</td><td>s</td><td>d</td><td>a</td><td>d</td><td>s</td><td>a</td><td>s</td></tr><tr><td>s</td><td>d</td><td>a</td><td>s</td><td>d</td><td>s</td><td>d</td><td>s</td><td>a</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>a</td><td>s</td><td>d</td><td>d</td><td>d</td></tr></table>	a	s	s	s	a	a	d	a	s	a	d	s	d	a	s	d	a	d	s	d	s	d	a	d	s	a	s	s	d	a	s	d	s	d	s	a	s	s	s	s	a	s	d	d	d	Areas: 21 Letter 'a' -> 6 Letter 'd' -> 7 Letter 's' -> 8
a	s	s	s	a	a	d	a	s																																							
a	d	s	d	a	s	d	a	d																																							
s	d	s	d	a	d	s	a	s																																							
s	d	a	s	d	s	d	s	a																																							
s	s	s	s	a	s	d	d	d																																							

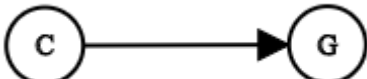
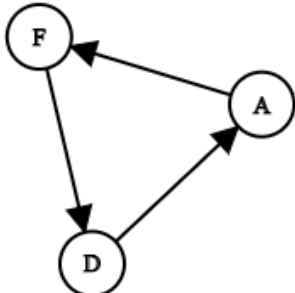
Hint

Initially mark all cells as **unvisited**. Start a **recursive DFS traversal** (or BFS) from each unvisited cell and mark all reached cells as visited. Each DFS traversal will find one of the **connected areas**.

3. Cycles in a Graph

Write a program to check whether a directed graph is **acyclic** or holds any cycles. The input ends with "End" line.

Examples

Input	Picture	Output
C-G End		Acyclic: Yes
A-F F-D D-A End		Acyclic: No

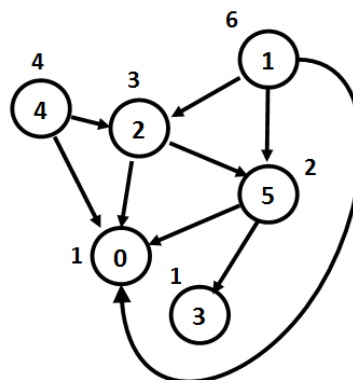
E-Q Q-P P-B End		Acyclic: Yes
K-J J-N N-L N-M M-I End		Acyclic: Yes

4. Salaries

We have a **hierarchy** between the employees in a company.

Employees can have one or several direct managers. People who **manage nobody** are called **regular employees** and their salaries are **1**. People who manage at least one employee are called **managers**. Each manager takes a **salary** which is equal to the **sum of the salaries of their directly managed employees**.

Managers cannot manage directly or indirectly themselves. Some employees might have no manager. See a sample hierarchy in a company along with the salaries computed following the above-described rule:



In the above example, employees 0 and 3 are regular employees and take salary 1. All others are managers and take the sum of the salaries of their directly managed employees. For example, manager 1 takes salary $3 + 2 + 1 = 6$ (sum of the salaries of employees 2, 5 and 0). In the above example employees, 4 and 1 have no manager.

If we have **N** employees, they will be indexed from 0 to $N - 1$. For each employee, you will be given a string with **N** symbols. The symbols are either **'Y'** or **'N'**', showing all employees that are managed by the current employee.

Input

- On the first line, you will be given an integer **N**.
- On the next **N** lines, you will be given strings with **N** symbols (either **'Y'** or **'N'**').

- The input data will always be valid and in the format described. There is no need to check it explicitly.

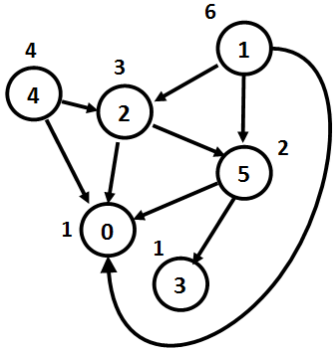
Output

- Print the sum of the salaries of all employees.

Constraints

- N will be an integer in the range [1 ... 50].
- If employee A is the manager of employee B, B will not be a manager of A.

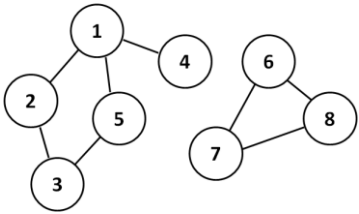
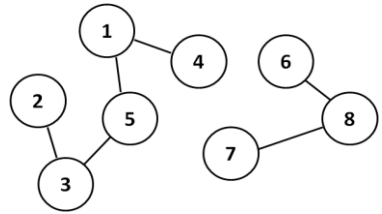
Examples

Input	Output	Comments
1 N	1	Only 1 employee with salary 1.
4 NNYN NNYN NNNN NYYN	5	We have 4 employees. 0, 1, and 3 are managers of 2. 3 is also a manager of 1. Therefore: salary(2) = 1 salary(0) = salary(2) = 1 salary(1) = salary(2) = 1 salary(3) = salary(2) + salary(1) = 2
6 NNNNNN YNYNNY YNNNNY NNNNNN YNYNNN YNNYNN	17	

5. Break Cycles

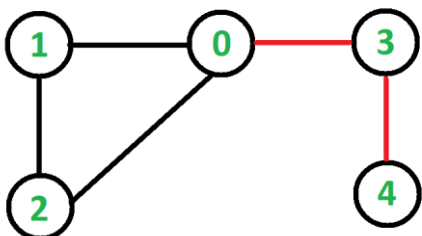
You are given an **undirected multi-graph**. Remove a minimal number of edges to **make the graph acyclic** (to break all cycles). As a result, print the number of edges removed and the removed edges. If several edges can be removed to break a certain cycle, remove the smallest of them in alphabetical order (smallest start vertex in alphabetical order and smallest end vertex in alphabetical order).

Examples

Input	Picture	Output	Picture After Removal
8 1 -> 2 5 4 2 -> 1 3 3 -> 2 5 4 -> 1 5 -> 1 3		Edges to remove: 2 1 - 2 6 - 7	

- The **order** of the output does **not matter** if you print all the important streets.

Examples

Input	Output
5 5 1 - 0 0 - 2 2 - 1 0 - 3 3 - 4	Important streets: 0 3 3 4 
7 8 0 - 1 1 - 2 2 - 0 1 - 3 1 - 4 1 - 6 3 - 5 4 - 5	Important streets: 1 6 