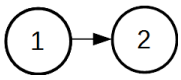
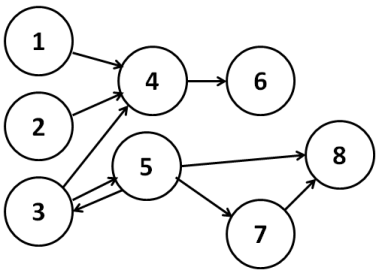
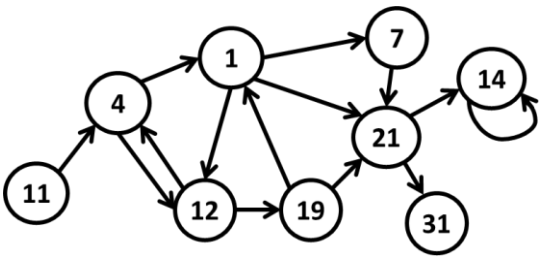
Homework: Graph Algorithms

This document defines the **homework assignments** for the "Algortihms" course @ Software University. Please submit a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems.

Problem 1. Distance between Vertices

We are given a **directed graph** consisting of N vertices and M edges. 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. There are no specified requirements for the input and output, so you may hardcode the input and output values.

Examples:

Input	Picture	Output
Graph: 1 -> 2 2 -> Distances to find: 1-2 2-1		{1, 2} -> 1 {2, 1} -> -1
Graph: 1 -> 4 2 -> 4 3 -> 4, 5 4 -> 6 5 -> 3, 7, 8 6 -> 7 -> 8 8 -> Distances to find: 1-6 1-5 5-6 5-8		{1, 6} -> 2 {1, 5} -> -1 {5, 6} -> 3 {5, 8} -> 1
Graph: 11 -> 4 4 -> 12, 1 1 -> 12, 21, 7 7 -> 21 12 -> 4, 19 19 -> 1, 21 21 -> 14, 31 14 -> 14 31 -> Distances to find: 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.

Problem 2. Areas in Matrix

We are given a matrix of letters of size N * M. Two cells are neighbor 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.

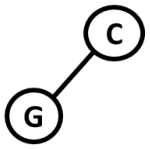
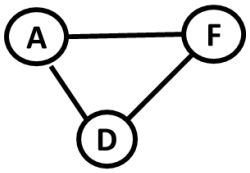
Examples:

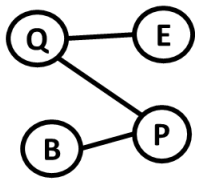
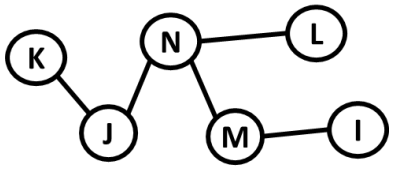
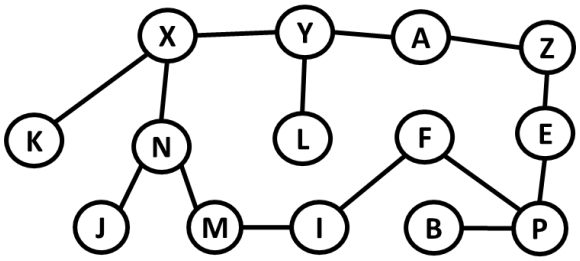
Input	Picture	Output
Number of rows: 6 aacccaac baaaaccc baabaccc bbdaaccc ccdccccc ccdccccc	a a c c c a a c b a a a a c c c b a a b a c c c b b d a a c c c c c d c c c c c c c d c c c c c	Areas: 8 Letter 'a' -> 2 Letter 'b' -> 2 Letter 'c' -> 3 Letter 'd' -> 1
Number of rows: 3 aaa aaa aaa	a a a a a a a a a	Areas: 1 Letter 'a' -> 1
Number of rows: 5 asssaadas adsdasdad sdsdadsas sdasdsdsa ssssasddd	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 's' -> 8 Letter 'd' -> 7

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**.

Problem 3. Cycles in a Graph

Write a program to check whether an undirected graph is **acyclic** or holds any cycles.

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

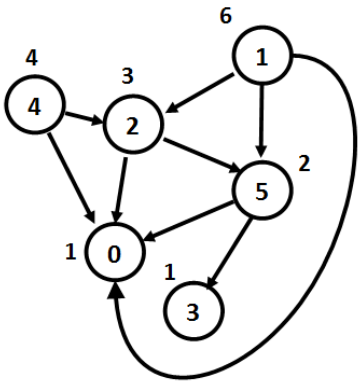
E – Q Q – P P – B		Acyclic: Yes
K – J J – N N – L N – M M – I		Acyclic: Yes
K – X X – Y X – N N – J M – N A – Z B – P I – F A – Y Y – L M – I F – P Z – E P – E		Acyclic: No

Hint: Modify the Topological Sorting algorithm (source removal or DFS-based).

Problem 4. Salaries

You can test your solution to the problem in the Judge system here.

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 (transitively) themselves. Some employees might have no manager (like the big boss). See a sample hierarchy in a company along with the salaries computed following the above described rule:



In the above example the 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'll be given a string with N symbols. The symbol at a given index **i**, either **'Y' or 'N'**, shows whether the current employee is a direct manager of employee **i**.

Hint: find the node with no parent and start a **DFS traversal** from it to calculate the salaries on the tree recursively.

Input

- The input data should be read from the console.
- On the first line you'll be given an integer N.
- On the next N lines you'll 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.

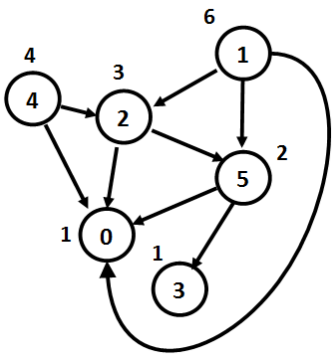
Output

- The output should be printed on the console. It should consist of one line.
- On the only output line print the sum of the salaries of all employees.

Constraints

- N will be an integer in the range [1 … 50].
- For each i-th line, the i-th symbol will be 'N'.
- If employee A is the manager of employee B, B will not be a manager of A.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

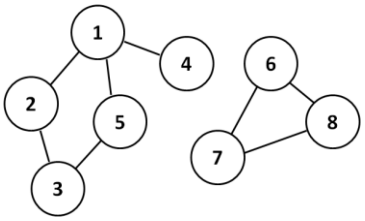
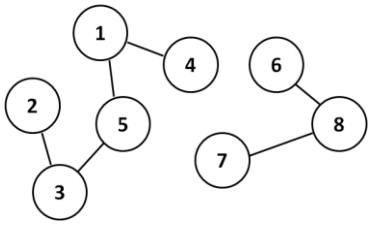
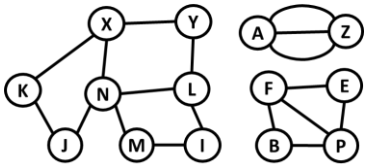
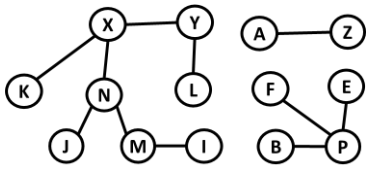
Input	Output	Comments
1 N	1	Only 1 employee with salary 1.
4 NNYN NNYN NNNN NYYN	5	We have 4 employees. 0, 1, and 3 are managers of 2. 3 is also a manager of 1. Therefore: salary(2) = 1 salary(0) = salary(2) = 1 salary(1) = salary(2) = 1 salary(3) = salary(2) + salary(1) = 2
6 NNNNNN YNYNNY YNNNNY NNNNNN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

Problem 5. * Break Cycles

You are given **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

1 -> 2, 5, 4 2 -> 1, 3 3 -> 2, 5 4 -> 1 5 -> 1, 3 6 -> 7, 8 7 -> 6, 8 8 -> 6, 7		Edges to remove: 2 1 - 2 6 - 7	
K -> X, J J -> K, N N -> J, X, L, M X -> K, N, Y M -> N, I Y -> X, L L -> N, I, Y I -> M, L A -> Z, Z, Z Z -> A, A, A F -> E, B, P E -> F, P P -> B, F, E B -> F, P		Edgeds to remove: 7 A - Z A - Z B - F E - F I - L J - K L - N	

Hints (Click on the arrow to show)

- Enumerate edges {**s**, **e**} in alphabetical order. For each edge {**s**, **e**} check whether it closes a cycle. If yes - remove it.
    - To check whether an edge {**s**, **e**} closes a cycle, temporarily remove the edge {s, e} and then try to find a path from **s** to **e** using DFS or BFS.