Barely Covered

STL, Learn Proper Documentation, Autograding, Commenting, Brute Force

Due: October 2nd

Description

Consider a graph (V, E) where V is a set of nodes and E is a set of edges which connect 2 nodes and ordering on the elements in V, then the cover of two connected vertices v_i and v_j is defined as the distance in the ordering between v_i and v_j . The maximum cover of the ordering is then defined as the maximum of the individual covers. For example, consider the following graph: This can be

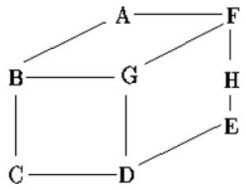

Figure 1: A Sample Graph

ordered in many ways, two of which are illustrated below (with lines connecting adjacent vertices): For these orderings, the maximum covers of the nodes (in order) are 6, 6, 1, 4, 1, 1, 6, 6 giving a

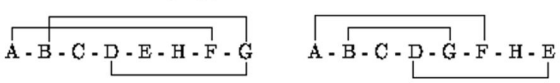

Figure 2: Two possible orderings

maximum cover of 6, and 5, 3, 1, 4, 3, 5, 1, 4 giving a maximum cover of 5. Write a program that will find the ordering of a graph that minimizes the maximum cover.

Input

Input consists of a series of graphs. The first line of input contains a single integer g indicating the number of graphs. The first line of each graph contains of an integer n indicating the number $(n > 0)$ of vertices in the graph. The next n lines are of the format $ldl_1l_2 \ldots l_d$ indicating that

vertex l is connected to the vertices $l_1, l_2 \ldots and l_d$ Although for analysis you should assume an infinite number of nodes, each node name (a single upper case character in the range 'A' to 'Z' .

Output

Output will consist of one line for each graph, listing the ordering of the nodes followed the maximum cover for that ordering. All items must be separated from their neighbors by exactly one space. If more than one ordering produces the same maximum cover, then choose the one that would appear first in an alphabetic listing.

Sample Input

```
1
8
A 2 B F
B 3 A G C
C 2 B D
D 3 C G E
E 2 D H
F 3 A G H
G 3 B F D
H 2 F E
```

Corresponding Sample Output

```
A B C F G D H E 3
```

Required Data Structure

```
class Graph
{
public:
Graph (int size); // creates an empty graph with size vertices
void fillGraph(); // fills in the graph from cin
void printGraph(); // prints the graph (for debugging only)
int maxCover( vector<char>  order); // returns the maxCover for the
                                        // ordering order
int cover( char vertex, vector<char> order); // returns the cover size for vertex
private:
Matrix<char> adj;
};
// from Data Structures and Algorithm Analysis in C++ (Third Edition), by Mark Allen Weiss
template <class Object>
class Matrix
```

```
{
  public:
    Matrix( int rows, int cols ) : array( rows )
    {
        for( int i = 0; i < rows; i++ )
            array[ i ].resize( cols );
    }

    const vector<Object> & operator[]( int row ) const
      { return array[ row ]; }
    vector<Object> & operator[]( int row )
      { return array[ row ]; }

    int numrows( ) const
        { return array.size( ); }
    int numcols( ) const
        { return numrows( ) ? array[ 0 ].size( ) : 0; }
  private:
    vector< vector<Object> > array;
};
```

How the program will be graded

You should electronically submit two documents, your memo and your source code. There will one folder . You can submit more than 1 document to canvas . If you upload the same document twice the first will replace the second.

Memo

What	pts	Oct 2^{nd}
Name	1	
Time Analysis O() of every function[1] (in terms of the number of letters and the number of previous guesses only)	8	
Space Analysis O() of every function	8	
Test Plan[2] with at least 4 original nontrivial[3] tests	12	

[1]The main() is a function.

[2]A test plan is a table with 4 columns and 1 row per test. The columns are named Reason for the test, actual input data, expected output data, and actual output. You do NOT have to have a working program to write a test plan. Each reason should be unique.

[3]A non trivial test contains only legal data (data the conforms to the input specification) with graphs contain at least 1 vertex .

Source Code Document

What	pts	Oct 2^{nd}
Name	1	
Description[4]	3	
Style	10	
pre/post conditions[5]	7	
Functionality using the STL	50	
fillGraph	10	
printGraph	5	
maxCover	25	
rest	10	

[4]The description should be written to some one who knows NOTHING about the program. It should discuss what the program does (in your own words). After reading the description the user should be able to create legal input and predict the output. This includes a description of the format of the input and the output.

[5]Preconditions are indicated with pre: and are the assumptions the function makes to work properly. Often involve legal ranges of all parameters. Postconditions are indicated with post: and say want one can assume after the function runs. Includes what the function does. All functions need post conditions.