Assignment 2: Exhaustive Search

SangYong Jin, Danny Navarro, Shelley Pham

Given a network with n > 3 nodes and a weight matrix W[0..n−1, 0..n−1] of positive integers, of a weighted, connected undirected graph modeling a network, decide whether the network is one of the topologies, if any: ring, star, fully connected mesh.
Note: represent infinity by the value 100.

INPUT: a positive integer n and a list of n^2 positive values
OUTPUT: message "ring" or "star" or "complete" or "neither"

Contents

Pseudocode

Ring

`for (i=0 to n-1) do:`
`  for (j=0 to n-1) do:`
`  if (W[i][j] != 0 AND W[i][j] != 100) do:`
`  count++`
`  endfor`
 	
`  //if the vertex does not have 2 edges, then it's not`
`  //a ring`
`  if (count != 2) do:`
`  cond = false`
`  break`
 	
`  count = 0`
`endfor`
	`- ((n-1-0)/1 + 1) = n`
	`- ((n-1-0)/1 + 1) = n`
	`- 3 + max(1,0) = 4`
	
	`- 1 + max(2,0) = 3`
	
	`- 1`

Proof

S.C. = n * [(n * 4) + 3 + 1] = 4 n^2 + 4n

Proof for O(n^2)

$$\lim_{n\to\infty} \frac{4n^2 + 4n}{n^2} = \lim_{n\to\infty} \frac{4n + 4}{n} = \lim_{n\to\infty} \frac{4n}{n} + \lim_{n\to\infty} \frac{4}{n} = \lim_{n\to\infty} \frac{4n}{n} + 0 = \lim_{n\to\infty} \frac{4n}{n}$$

l'Hospital Rule = $\lim_{n\to\infty} 4$ = 4 >= 0 and finite;

Back to Contents

Star

`for (i=0 to n-1) do:`
`  for (j=0 to n-1) do:`
`  if (W[i][j] != 0 AND W[i][j] != 100) do:`
`  count++`
`  endfor`
 	
`  //if there exist a center and there exists a vertex`
`  //with more than 1 edge`
`  if (center == 1 && count > 1) do:`
`  cond = false`
`  break`
 	
`  //if there's a vertex with n-1 edges, then the`
`  //topology could be a star`
`  if (count == n-1) do:`
`  center = 1`
 	
`  count = 0`
`endfor`
 	
`//if we reached the end of the matrix and there's no`
`//center, then the topology is not a star`
`if (center == 0)`
`  cond = false`
	`-((n-1-0)/1 + 1) = n`
	`- n`
	`- 3 + max(1,0) = 4`
	
	`- 3 + max(2,0) = 5`
	
	`- 2 + max(1,0)= 3`
	
	`- 1`
	
	`- 1 + max(1,0)`

Back to Contents

Proof

S.C. = [n * (4n + 5 + 3 + 1))] + [1+max(1,0)] = $4n^2 + 9n + 2$

Proof of O(n^2):

$$\lim_{n\to\infty} \frac{4n^2+9n+2}{n^2} = \lim_{n\to\infty} 4 + \lim_{n\to\infty} \frac{9}{n} + \lim_{n\to\infty} \frac{2}{n^2} = 4 + 0 + 0 = \quad 4 >= 0 \text{ and finite;}$$

Back to Contents

Mesh

`//iterates through lower half triangle of the matrix`
`for (i=1 to n-1) do:`
`  for (j=0 to i-1) do:`
`  //if there's a vertex that's not connected to`
`  //another vertex, it's not a mesh`
`  if (W[i][j] != 100) do:`
`  cond = false`
`  break`
`  endfor`
 	
`  //break out of loop if 'it's not a mesh`
`  if (!cond) do: break`
 	
`endfor`
	- $\sum_{i=1}^{n-1}$
	- `((i-1-0)/1 + 1) * 3 = 3i`
	
	- `1 + max(2,0)`
	
	- `2`

Proof

S.C. = ($\sum_{i=1}^{n-1}$ 3i) + 2 = [(n-1)/2] * [3 + 3(n-1)] + 2 = [(n-1)/2] * [3n] + 2 = $\frac{3n^2 - 3n}{2}$ + 2 = (3/2)n^2 + (3/2)n + 2

Proof of O(n^2)

$\lim_{n\to\infty} \frac{(3/2)n^2 + (3/2)n + 2}{n^2}$ = $\lim_{n\to\infty} \frac{(3/2)n^2}{n^2}$ + $\lim_{n\to\infty} \frac{(3/2)n}{n^2}$ + $\lim_{n\to\infty} \frac{2}{n^2}$ = $\lim_{n\to\infty}$ (3/2) + $\lim_{n\to\infty} \frac{(3/2)}{n}$ + $\lim_{n\to\infty} \frac{2}{n^2}$

= (3/2) + 0 + 0 = (3/2) >= 0 and finite;

Back to Contents

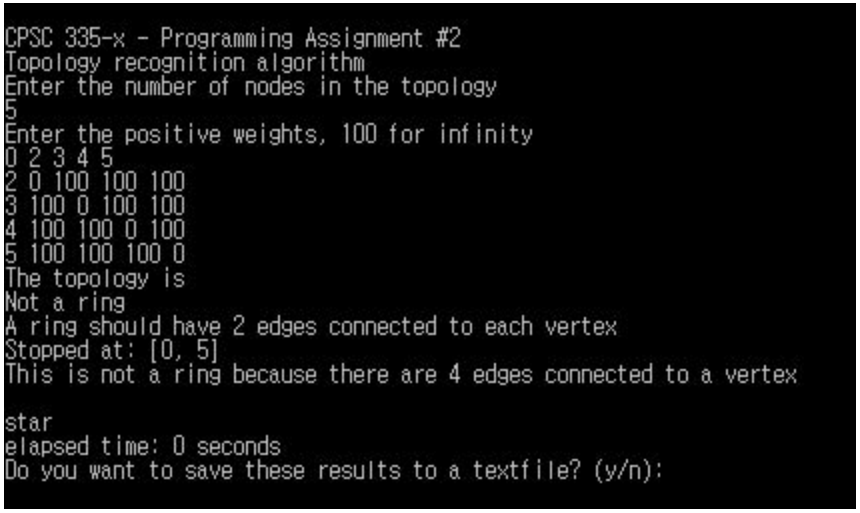
Output

Ring

```
CPSC 335-x - Programming Assignment #2
Topology recognition algorithm
Enter the number of nodes in the topology
5
Enter the positive weights, 100 for infinity
0 2 100 100 5
2 0 3 100 100
100 3 0 1 100
100 100 1 0 4
5 100 100 4 0
The topology is
ring
elapsed time: 0 seconds
Do you want to save these results to a textfile? (y/n):
```

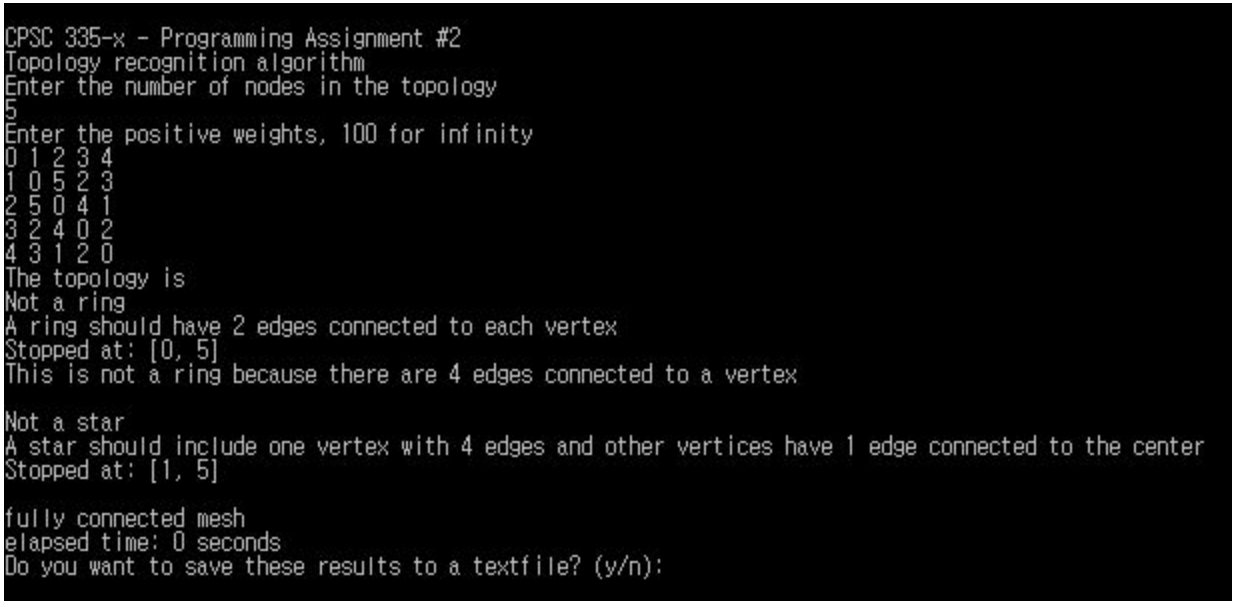
Back to Contents

Star



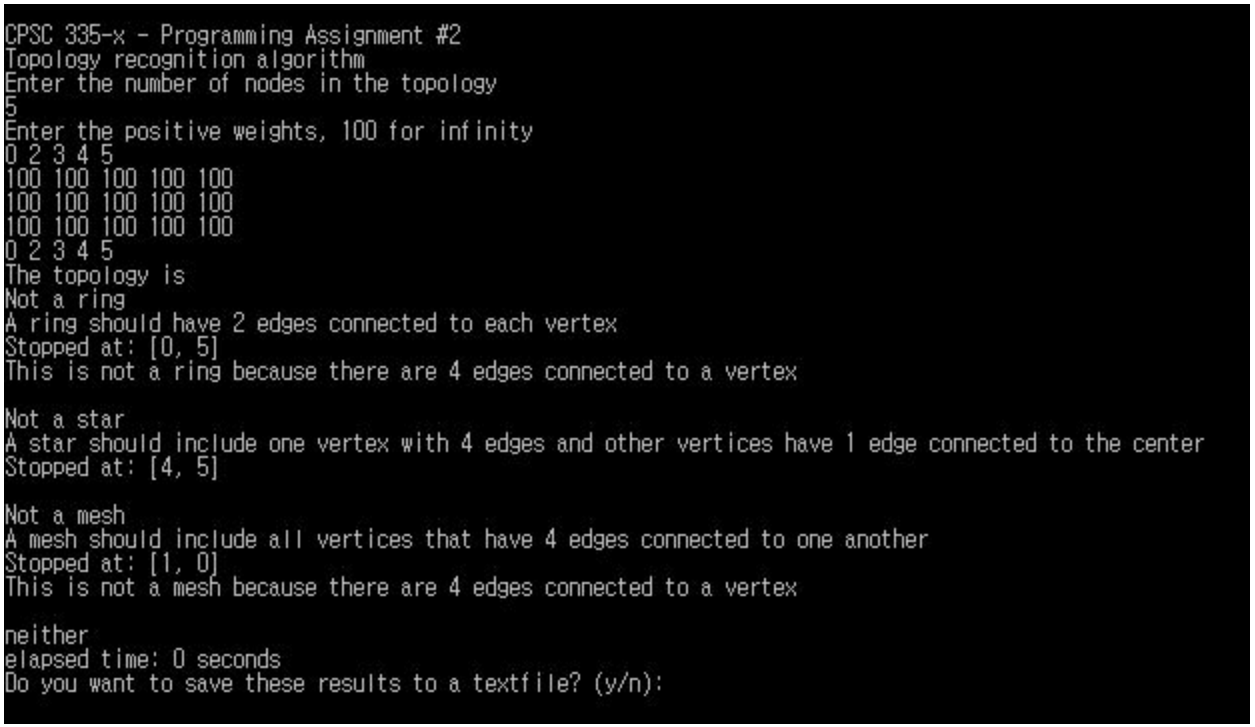
Back to Contents

Mesh



Back to Contents

Neither



Back to Contents

Code

```
// Assignment 2: Topology recognition problem
// Sangyong Jin, Danny Navarro, Shelley Pham

#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <chrono>
#include <fstream>
#include <string>

using namespace std;

void print_to_file(int n, int W[][100], string topology, double seconds);

int main() {
      char repeat = 'y';
      double seconds = 0;
      string topology;

      int n, i, j, count, center;
      int W[100][100];
      bool cond;

      // display the header
      cout << endl << "CPSC 335-x - Programming Assignment #2" << endl;
      cout << "Topology recognition algorithm" << endl;
      cout << "Enter the number of nodes in the topology" << endl;
      // read the number of nodes
```

Back to Contents

```
cin >> n;
// read the weight matrix
cout << "Enter the positive weights, 100 for infinity" << endl;
for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
            cin >> W[i][j];

cout << "The topology is" << endl;

// Start the chronograph to time the execution of the algorithm
auto start = chrono::high_resolution_clock::now();
count = 0;
cond = true;
// loop to check whether the topology is a ring
for (i = 0; i < n; i++) {
      // YOU NEED TO IMPLEMENT THIS LOOP
      for (j = 0; j < n; j++) {
            //count number of edges connected to the vertex
            if (W[i][j] != 0 && W[i][j] != 100)
                  count++;
      }
      //if there's not exactly 2 edges connected to the vertex, it's not a ring
      if (count != 2) { cond = false; break; }

      count = 0; //reset counter
}

// End the chronograph to time the loop
auto end = chrono::high_resolution_clock::now();

if (cond) {
```

Back to Contents

```
            topology = "ring";
            cout << topology << endl;
            int microseconds = chrono::duration_cast<chrono::microseconds>(end - start).count();
            seconds = microseconds / 1E6;
            cout << "elapsed time: " << seconds << " seconds" << endl;
            print_to_file(n, W, topology, seconds);
            system("pause");
            return EXIT_SUCCESS;
      }
      //for personal checking
      else {
            cout << "Not a ring" << endl;
            cout << "A ring should have 2 edges connected to each vertex" << endl;
            cout << "Stopped at: [" << i << ", " << j << "]" << endl;
            if (count > 2)
                  cout << "This is not a ring because there are " << count << " edges connected
to a vertex" << endl << endl;
      }


      // Start the chronograph to time the execution of the algorithm
      start = chrono::high_resolution_clock::now();

      count = 0;
      cond = true;
      center = 0;
      // loop to check whether the topology is a star
      for (i = 0; i < n; i++) {
            // YOU NEED TO IMPLEMENT THIS LOOP
            for (j = 0; j < n; j++) {
                  //count the number of edges
```

Back to Contents

```
            if (W[i][j] != 0 && W[i][j] != 100)
                  count++;
      }

      //if there's already a center and there's more than 1 edge, it's not a star
      if (center == 1 && count > 1) { cond = false; break; }
      //if there's a vertex with n-1 edges, it could be a star
      if (count == n - 1) { center = 1; }

      count = 0; //reset counter
}
//if we reached the end of the matrix and has no center, then it's not a star
if (center == 0) { cond = false; }

// End the chronograph to time the loop
end = chrono::high_resolution_clock::now();

if (cond && (center == 1)) {
      int microseconds = chrono::duration_cast<chrono::microseconds>(end - start).count();
      seconds = microseconds / 1E6;
      topology = "star";
      cout << topology << endl;
      cout << "elapsed time: " << seconds << " seconds" << endl;
      print_to_file(n, W, topology, seconds);
      system("pause");
      return EXIT_SUCCESS;
}
//for personal checking
else {
      cout << "Not a star" << endl;
```

Back to Contents

```
            cout << "A star should include one vertex with " << n - 1 << " edges and other
vertices have 1 edge connected to the center" << endl;
            cout << "Stopped at: [" << i << ", " << j << "]" << endl;
            if (count > n - 1)
                  cout << "This is not a star because there are " << count << " edges connected
to a vertex" << endl;
            if (center == 0)
                  cout << "This is not a star because there's no center" << endl;
            cout << endl;
      }

      // Start the chronograph to time the execution of the algorithm
      start = chrono::high_resolution_clock::now();

      cond = true;
      // loop to check whether the topology is a fully connected mesh
      for (i = 1; i < n; i++) {
            // YOU NEED TO IMPLEMENT THIS LOOP
            for (j = 0; j < i; j++) {
                  //check whether there exist a vertex that's not connected to another one
                  if (W[i][j] == 100) { cond = false; break; }
            }
            //if there exist a vertex that's not connected to another existing vertex, it's not a
mesh
            if (!cond) { break; }
      }

      // End the chronograph to time the loop
      end = chrono::high_resolution_clock::now();

      if (cond) {
```

Back to Contents

```
            topology = "fully connected mesh";
            cout << topology << endl;

            int microseconds = chrono::duration_cast<chrono::microseconds>(end - start).count();
            seconds = microseconds / 1E6;
            cout << "elapsed time: " << seconds << " seconds" << endl;
            print_to_file(n, W, topology, seconds);
      }
      else {
            //for personal checking
            cout << "Not a mesh" << endl;
            cout << "A mesh should include all vertices that have " << n - 1 << " edges connected
to one another" << endl;
            cout << "Stopped at: [" << i << ", " << j << "]" << endl;
            cout << "This is not a mesh because there are " << count << " edges connected to a
vertex" << endl << endl;

            topology = "neither";
            cout << topology << endl;
            int microseconds = chrono::duration_cast<chrono::microseconds>(end - start).count();
            seconds = microseconds / 1E6;
            cout << "elapsed time: " << seconds << " seconds" << endl;
            print_to_file(n, W, topology, seconds);
      }

      system("pause");

      return EXIT_SUCCESS;
}

void print_to_file(int n, int W[][100], string topology, double seconds) {
```

Back to Contents

```
	//asks if user wants to output result to textfile
	char ans;
	cout << "Do you want to save these results to a textfile? (y/n): ";
	cin >> ans;
	if (ans == 'y' || ans == 'Y') {
		ofstream resultfile("topology.txt");
		if (resultfile.is_open()) {
			for (int i = 0; i < n; i++) {
				for (int j = 0; j < n; j++) {
					resultfile << W[i][j] << " ";
				}
				resultfile << endl;
			}
			resultfile << "\nThe topology is a " << topology << endl;
			resultfile << "Elapsed time: " << seconds << " seconds" << endl;
		}
	}
}
```

Back to Contents