Department of Computer Science
Trent University

COIS 3020H: Data Structures and Algorithms II

Assignment 1: The Internet

(115 marks)

(due Sunday, February 4, 2024 at 23:59 ET)

Instructions

1. Work in teams of two or three. Solo assignments are not accepted.
2. Work together on the same tasks rather than apart on different tasks.
3. A 10% penalty is applied for each day late up to five days. Thereafter, no submissions are accepted.

Introduction

The Internet can be viewed from both physical and logical perspectives. From a physical point of view, the Internet is a collection of web servers connected together with fibre optic cables. From a logical point of view, the Internet is a collection of webpages connected together with hyperlinks. For this assignment, both perspectives are modeled as graphs in the following ways.

Server Graph

The collection of servers, on one hand, is represented by an undirected, unweighted server graph $S = (V, E)$ where:

1. each vertex $s \in V$ represents a web server and
2. each undirected edge $(s, t) \in E$ represents the physical connection between server s and server t.

Each server has a unique name and a list of the webpages that it hosts. Also, there is no hard limit on the number of servers.

Web Graph

The collection of webpages, on the other hand, is represented by a directed, unweighted web graph $W = (V, E)$ where:

1. each vertex $u \in V$ is a webpage and
2. each directed edge $(u, v) \in E$ is a hyperlink from webpage u to webpage v. To simplify our discussion, there is at most one hyperlink from u to v.

Each webpage has a unique name as well as the name of the server that hosts it.

Task 1: Server Graph (43 marks)

Using the partial definitions below, implement and test the server graph as an expandable adjacency matrix.

```
public class ServerGraph
{
    // 3 marks
    private class WebServer
    {
        public string Name;
        public List<WebPage> P;
        ...
    }

    private WebServer[] V;
    private bool[,] E;
    private int NumServers;
    ...

    // 2 marks
    // Create an empty server graph
    public      ServerGraph( ) ...

    // 2 marks
    // Return the index of the server with the given name; otherwise return -1
    private int FindServer(string name) ...

    // 3 marks
    // Double the capacity of the server graph with the respect to web servers
    private void DoubleCapacity( ) ...

    // 3 marks
    // Add a server with the given name and connect it to the other server
    // Return true if successful; otherwise return false
    public bool AddServer(string name, string other) ...

    // 3 marks
    // Add a webpage to the server with the given name
    // Return true if successful; otherwise return false
    public bool AddWebPage(WebPage w, string name) ...

    // 4 marks
    // Remove the server with the given name by assigning its connections
    // and webpages to the other server
    // Return true if successful; otherwise return false
    public bool RemoveServer(string name, string other) ...

    // 3 marks (Bonus)
    // Remove the webpage from the server with the given name
    // Return true if successful; otherwise return false
    public bool RemoveWebPage(string webpage, string name) ...

    // 3 marks
    // Add a connection from one server to another
    // Return true if successful; otherwise return false
    // Note that each server is connected to at least one other server
    public bool AddConnection(string from, string to) ...
    // 10 marks
    // Return all servers that would disconnect the server graph into
    // two or more disjoint graphs if ever one of them would go down
    // Hint: Use a variation of the depth-first search
    public string[] CriticalServers( ) ...

    // 6 marks
    // Return the shortest path from one server to another
    // Hint: Use a variation of the breadth-first search
    public int ShortestPath(string from, string to)

    // 4 marks
    // Print the name and connections of each server as well as
    // the names of the webpages it hosts
    public void PrintGraph( ) ...
    ...
}
```

Task 2: Web Graph (36 marks)

Using the partial definitions below, implement and test the web graph as an expandable adjacency list.

```
// 5 marks
public class WebPage
{
    public string Name      {get; set;}
    public string Server    {get; set;}
    public List<WebPage> E  {get; set;}
    ...

    public WebPage(string name, string host) ...
    public int FindLink(string name) ...
    ...
}

public class WebGraph
{
    private List<WebPage> P;
    ...

    // 2 marks
    // Create an empty WebGraph
    public WebGraph( ) ...

    // 2 marks
    // Return the index of the webpage with the given name; otherwise return -1
    private int FindPage(string name) ...

    // 4 marks
    // Add a webpage with the given name and store it on the host server
    // Return true if successful; otherwise return false
    public bool AddPage(string name, string host, ServerGraph S) ...

    // 8 marks
    // Remove the webpage with the given name, including the hyperlinks
    // from and to the webpage
    // Return true if successful; otherwise return false
    public bool RemovePage (string name, ServerGraph S) ...
```

```
        // 3 marks
        // Add a hyperlink from one webpage to another
        // Return true if successful; otherwise return false
        public bool AddLink (string from, string to) ...

        // 3 marks
        // Remove a hyperlink from one webpage to another
        // Return true if successful; otherwise return false
        public bool RemoveLink (string from, string to) ...

        // 6 marks
        // Return the average length of the shortest paths from the webpage with
        // given name to each of its hyperlinks
        // Hint: Use the method ShortestPath in the class ServerGraph
        public float AvgShortestPaths(string name, ServerGraph S) ...

        // 3 marks
        // Print the name and hyperlinks of each webpage
        public void PrintGraph( ) ...
        ...
    }
```

Task 3: The Main (16 marks)

The server and web graphs define the physical and logical models of the Internet and work in tandem to reflect the **same** reality. In the main program, an instance of each graph is instantiated and the following steps are carried out to exercise your program. Ensure that both successful and unsuccessful cases are tested.

1. Instantiate a server graph and a web graph.
2. Add a number of servers.
3. Add additional connections between servers.
4. Add a number of webpages to various servers.
5. Add and remove hyperlinks between the webpages.
6. Remove both webpages and servers.
7. Determine the critical servers of the remaining Internet.
8. Calculate the average shortest distance to the hyperlinks of a given webpage.

Testing and Documentation (20 marks)

In a separate document, illustrate your test cases and compare them with the results of your program.

Submission

Please submit (zip) all source and executable code along with your test cases via Blackboard. Only one submission per team is required. Make sure to include the names of all team members on your submission. However, if a team member does not participate in the completion of the assignment then that team member can be omitted.