Data Structures (Spring 2021)
Semester Project

Total Marks: 180 **Deadline: 18th June 2021 9:00 PM**

Project groups: This project can be done within a group of preferably three (3) students. There is no restriction on the selection of group members and students can make groups according to their preferences.
Submission: All submissions MUST be uploaded on Google classroom. Solutions sent to the emails will not be graded. To avoid last minute problems (unavailability of Internet, load shedding etc.), you are strongly advised to start working on the project from day one.
You are required to use Visual Studio 2019 for the project. Combine all your work (solution folder) in one .zip file after performing "Clean Solution". Submit zip file on Google classroom within the given deadline. If only .cpp file is submitted it will not be considered for evaluation.
Plagiarism: Zero marks in the project for all members of the group if any significant part of the project is found plagiarized. A code is considered plagiarized if more than 20% code is not your own work.

Network Analysis using Graphs

First things first

Before starting the project (or reading this statement) you should shout out the following three times!

"I will learn something new and exciting in this project"

Overview

There are three datasets given with this project statement. You are required to complete the tasks, given below, for any one of the datasets. At the maximum, you can try any two of the three datasets for more understanding on how graphs help in investigating hidden patterns in the data (you can read it as, "for bonus marks"). You should first try to complete all the tasks on the smallest dataset and then use the same code for the others. **(170 marks)**

You are also required to submit a one-page report about your understanding of the dataset. In the report you are required to explain the patterns in the dataset as much as you can. **(10 marks)**

Note: Your program MUST present a cool menu to run the tasks one at a time or all at once. We will use this menu to test your program. If the menu is not cool/useful, then we will deduct 20 marks from the overall obtained marks. If the output is not cool/readable, we will deduct further 10 marks from the overall obtained marks.

Grading will be done individually after a demo. If a member is unable to explain the working/demo of the project, he/she will be given the minimum (say zero) marks in the project while others will get according to their understanding.

Datasets

1. General Relativity and Quantum Cosmology collaboration network Dataset Description

Arxiv GR-QC (General Relativity and Quantum Cosmology) collaboration network is from the e-print arXiv (https://arxiv.org/archive/gr-qc) and covers scientific collaborations between authors papers submitted to General Relativity and Quantum Cosmology category. If an author i co-authored a paper with author j, the graph contains a undirected edge from i to j. If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes.

The data covers papers in the period from January 1993 to April 2003 (124 months). It begins within a few months of the inception of the arXiv, and thus represents essentially the complete history of its GR-QC section.

2. Astro Physics collaboration network Dataset Description

Arxiv ASTRO-PH (Astro Physics) collaboration network is from the e-print arXiv (https://arxiv.org/archive/astro-ph) and covers scientific collaborations between authors papers submitted to Astro Physics category. If an author i co-authored a paper with author j, the graph contains a undirected edge from i to j. If the paper is co-authored by k authors this generates a completely connected (sub)graph on k nodes.

The data covers papers in the period from January 1993 to April 2003 (124 months). It begins within a few months of the inception of the arXiv, and thus represents essentially the complete history of its ASTRO-PH section.

3. Amazon product co-purchasing network and ground-truth communities Dataset Description

Data was collected by crawling Amazon website. It is based on *Customers Who Bought This Item Also Bought* (explained here: https://davidgaughran.com/also-boughts-amazon-recommendations/) feature of the Amazon website. If a product *i* is frequently co-purchased with product *j*, the graph contains an undirected edge from *i* to *j*. Each product category provided by Amazon defines each ground-truth community.

We regard each connected component in a product category as a separate ground-truth community. We remove the ground-truth communities which have less than 3 nodes.

Tasks

First Set of Tasks – Graph Stats

1. Display the number of nodes (5 marks)
2. Display the number of edges (5 marks)
3. Display the number of source nodes (5 marks)
4. Display the number of sink nodes (5 marks)
5. Display the number of isolated nodes (5 marks)
6. Display the number of bridge edges (15 marks)

7. Display the number of articulation nodes (15 marks)
8. Display the shortest path length distribution (20 marks)
9. Display the diameter of the graph (5 marks)

Second Set of Tasks – Degree Distributions

10. Display the in-degree distribution in the form of a table (10 marks)
11. Display the out-degree distribution in the form of a table (10 marks)

Third Set of Tasks – Connected Components

For the original graph:

12. Display the size of the largest strongly connected component (SCC) (25 marks)
13. Display the size distribution of all SCCs (10 marks)

Considering the graph as an undirected graph:

14. Display the size of the largest weakly connected component (WCC) (25 marks)
15. Display the size distribution of all WCCs (10 marks)

Algorithms

You will need the following algorithms for the third set of tasks.

Breadth-First Search (BFS) Algorithm

The BFS algorithm (Algorithm 1) takes as input a graph *G(V, E)* and a source vertex *s*. The algorithm returns the set of vertices that the source vertex has a path to.

Algorithm 1 BFS algorithm

```
1: procedure BFS(G(V, E),s)
2:     Let reachable be an array
3:     Let Q be a queue
4:     Q.enqueue(s)
5:     reachable.add(s)
6:     while Q is not empty do
7:         v ← Q.dequeue()
8:         for each neighbour w of v in V do
9:             if w is not in reachable then
10:                Q.enqueue(w)
11:                reachable.add(w)
12:            end if
13:        end for
14:    end while
15:    return reachable
16: end procedure
```

Strongly Connected Components (SCC) Algorithm

The SCC algorithm (Algorithm 4) takes a graph *G(V, E)* as input and returns a list of all the SCCs in this graph as output. The SCC of *G* is computed using Eq. 1.21. The function *unique* returns a list of all the distinct elements in the input list. For instance, *unique*([1,2,1,3,2,2]) will return [1,2,3].

Algorithm 2 In algorithm

```
1: procedure IN(G(V, E),s)
2:     Let In be an array
3:     for each vertex v in V do
4:         if s in BFS(G(V, E), v) then
5:             In.add(v)
6:         end if
7:     end for
8:     return In
9: end procedure
```

Algorithm 3 Out algorithm

```
1: procedure OUT(G(V, E),s)
2:     Let Out be an array
3:     Out ← BFS(G(V, E), s)
4:     return Out
5: end procedure
```

Algorithm 4 SCC algorithm

```
1: procedure SCC(G(V, E))
2:     Let SCC be an array
3:     for each vertex v in V do
4:         SCC.add(In(G(V, E), v) ∩ Out(G(V, E), v))
5:     end for
6:     return unique(SCC)
7: end procedure
```

Weakly Connected Components (WCC) Algorithm

The WCC algorithm (Algorithm 5) computes the list of all WCCs given a graph G(V, E) as input.

Algorithm 5 WCC algorithm

```
1: procedure WCC(G(V, E))
2:     Let WCC be an array
3:     Let G'(V', E') be G(V, E) as an undirected graph
4:     for each vertex v in V' do
5:         WCC.add(BFS(G'(V', E'), v)
6:     end for
7:     return unique(WCC)
8: end procedure
```

Definitions

Source Vertex: A vertex with zero in-degree is called a source vertex

Sink Vertex: A vertex with zero outdegree is called a sink vertex

Isolated Vertex: A vertex with in-degree and out-degree both equal to zero is called an isolated vertex.

Degree Distribution: The degree distribution of a graph G(V, E), denoted by P(k), is defined as the probability that a random chosen vertex has degree k. If |V|k denotes the number of vertices with degree k,

$$P(k) = \frac{|V|_k}{|V|}$$

Bridge Edge: A bridge edge is an edge whose removal disconnects the graph.

Articulation Vertex: An articulation vertex is defined as a vertex which when deleted renders the graph a disconnected one.

Diameter: The diameter of a graph is the maximum of the distances between all pairs of vertices in this graph. While computing the diameter of a graph, all infinite distances are disregarded.

Best of Luck ☺