

Frankfurt University of Applied Science

Object Oriented Programming with

Java – Advanced Course

**Basic analysis of graph-based**

**communication network models**

**Project Documentation**

Khuong Lu Minh - 1276079

Thinh Huynh Quang Phuoc - 1276202

Nguyen Nguyen Tri - 1276121

Cong Dang Chi - 1275957

Robin Müller-Bady

December 19, 2019

( this paper intentionally left blank )

Table of Contents

[1. Introduction 4](#_Toc31578924)

[**1.1.** **Purpose** 4](#_Toc31578925)

[**1.2.** **The scope of the Report** 4](#_Toc31578926)

[**1.3.** **Audiences** 4](#_Toc31578927)

[2. Group Members 4](#_Toc31578928)

[3. Project Documentation 4](#_Toc31578929)

[**3.1.** **Requirement** 4](#_Toc31578930)

[**3.2. Technical description** 5](#_Toc31578931)

[**3.2.1. Input \*.graphml** 5](#_Toc31578932)

[**3.2.2. Properties of edges and nodes** 5](#_Toc31578933)

[**3.2.3. Connectivity** 5](#_Toc31578934)

[**3.2.4. Diameter** 5](#_Toc31578935)

[**3.2.5. Shortest path** 5](#_Toc31578936)

[**3.2.6. Betweenness centrality measure** 5](#_Toc31578937)

[**3.2.7. Output to \*.graphml** 5](#_Toc31578938)

[**3.2.8. Error handling** 5](#_Toc31578939)

[**3.2.9. Thread** 6](#_Toc31578940)

[**3.3. UML diagram** 6](#_Toc31578941)

[4. Milestones 7](#_Toc31578942)

[**4.1. Milestone 1** 7](#_Toc31578943)

[**4.1.1. Work schedule** 7](#_Toc31578944)

[**4.1.2. Constraint** 7](#_Toc31578945)

[**4.2. Milestone 2** 7](#_Toc31578946)

[**4.2.1. Work schedule** 7](#_Toc31578947)

[**4.2.2. Constraint** 8](#_Toc31578948)

[**4.3. Milestone 3** 8](#_Toc31578949)

[**4.3.1. Work schedule** 8](#_Toc31578950)

[**4.3.2. Constraint** 8](#_Toc31578951)

# 1. Introduction

* 1. **Purpose**

The aim of this report is to satisfy the requirements for the Advanced OOP with JAVA – Winter 19/20. As its purpose, it includes an overall description of the project. In our project, we try to find the optimized solution for the given problems and implement in Java.

* 1. **The scope of the Report**

In total, what our project aims for is to solve the problem in the shortest time as possible. Our team works quite well on coding domain; but still there are some constraints in reporting and working together for a project as a team. With the length of 9 weeks, we will afford to complete it before the deadline.

* 1. **Audiences**

The intended audience of this document is the course instructor, who will base on this to determine if the project should be accepted as proposed, accepted with modifications, or not accepted, and to the members of the project team that will implement and verify as also take responsibility of the product.

# 2. Group Members

|  |  |
| --- | --- |
| **Name** | **Main responsibilities** |
| Khuong Lu Minh | -Main coding.  -Algorithm.  -Code orientation for team. |
| Nguyen Nguyen Tri | -Documentation.  -UML design. |
| Thinh Huynh Quang Phuoc | -Documentation.  -Object-oriented design. |
| Cong Dang Chi | -Algorithm  -Code |

# 3. Project Documentation

* 1. **Requirement**

1. Start the program from the CLI or GUI and provide an input file for processing.
2. The input file format is \*.graphml (an XML-based format). The files contain the node ID, described by node key v id, the edge ID, described by the edge key e id, and the edge weighting, described by the edge key e weight.
3. Information of the \*.graphml file is transferred into appropriate data structures.
4. Output the following graph properties of the provided network graph on the system output stream.
5. Number of nodes.
6. Number of edges.
7. The vertex names or IDs.
8. The edge names or IDs.
9. Connectivity.
10. Diameter.
11. Output the following node and edge properties of the provided network graph on the system output stream:
12. The shortest path between two vertices according to the Dijkstra algorithm.
13. Calculate the betweenness centrality measure for a selected node.
14. Calculate all above graph and node/edge properties and output them into a new \*.graphml file.

## **3.2. Technical description**

### **3.2.1. Input \*.graphml**

The program uses 4 words at a time as a sign to recognize and store edges, nodes,… .For example, if program encounters “v\_id” or “e\_id”, the program will store the corresponding edge or vertex ID in a list and count +=1.

### **3.2.2. Properties of edges and nodes**

The program prints out list of edges, nodes, source and target nodes of that edge.

### **3.2.3. Connectivity**

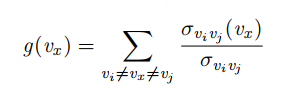
The program counts the number of nodes in the array of visited nodes. If that number is equal to the number of nodes of a graph, then the graph is connected. Otherwise, it is not connected, and the diameter is infinity.

### **3.2.4. Diameter**

For each pair of nodes, the program stores the value of total vertices as also the total weight of that graph. Then compare the results and choose the greatest value. Finally, the program prints out the weight of that path.

### **3.2.5. Shortest path**

We use the Dijkstra algorithm to calculate the shortest path.

****

### **3.2.6. Betweenness centrality measure**

To calculate the betweenness centrality, we use the given formula , in which *σvivj* is the total number of shortest paths from node *vi* to *vj* and *σvivj* (*vx*) is the number of shortest paths that passes through node *vx*.

### **3.2.7. Output to \*.graphml**

The program will:

* Print out the total number of vertices.
* For each vertex, print out the shortest path to other vertices as also its weight base on Dijkstra algorithm, then calculate the betweenness.
* Print out the total number of edges.
* For each edge, print out the source and target node, node ID and its weight.
* Verify the connection of the graph.
* Print out the weight of the longest shortest path.

### **3.2.8. Error handling**

The programs can:

* Check if the file existed or not.
* Check the extension of the file is corrected or not.
* Check if the graph is connected or not. If yes, the program will print it out.
* If thread is interrupted, return error message.
* If users input wrong defined type of input / negative number / adding too many or not enough values, return the error message

### **3.2.9. Thread**

Our program uses thread in calculating betweenness centrality by dividing into two threads, and each thread calculates half of the total vertices.

**3.2.10. Interface and abstract class**

Create 3 methods get\_ID(), toString() and printMe().

## **3.3. UML diagram**

Figure 1. UML diagram for Communication Network Analysis program

# 

# 4. Milestones

## **4.1. Milestone 1**

### **4.1.1. Work schedule**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | ii | iii | iv.a | iv.b | iv.c | iv.d | iv.e | iv.f | v.a | v.b | vi | UML diagram | Milestones 1 | User Handbook |
| Khuong Lu Minh |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Thinh Huynh Quang Phuoc |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nguyen Nguyen Tri |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cong Dang Chi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Figure 2. Gantt Chart of work schedule for milestone 1

|  |  |
| --- | --- |
| Completed |  |
| Working |  |
| Initial |  |
| Not plan |  |

### **4.1.2. Constraint**

We found some difficulties in:

- Using multi-threading in our code and synchronize them.

- Outputting to a \*.graphml file.

- Using logging and handler appropriately.

## **4.2. Milestone 2**

### **4.2.1. Work schedule**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | ii | iii | iv.a | iv.b | iv.c | iv.d | iv.e | iv.f | v.a | v.b | vi | UML diagram | Milestones 2 | User Handbook |
| Khuong Lu Minh |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Thinh Huynh Quang Phuoc |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nguyen Nguyen Tri |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cong Dang Chi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Figure 3. Gantt Chart of work schedule for milestone 2

|  |  |
| --- | --- |
| Completed |  |
| Working |  |
| Initial |  |
| Not plan |  |

### **4.2.2. Constraint**

We found some difficulties in:

- Outputting to a \*.graphml file.

## **4.3. Milestone 3**

### **4.3.1. Work schedule**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | ii | iii | iv.a | iv.b | iv.c | iv.d | iv.e | iv.f | v.a | v.b | vi | UML diagram | Milestones 3 | User Handbook |
| Khuong Lu Minh |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Thinh Huynh Quang Phuoc |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nguyen Nguyen Tri |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Cong Dang Chi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Figure 4. Gantt Chart of work schedule for milestone 3

|  |  |
| --- | --- |
| Completed |  |
| Working |  |
| Initial |  |
| Not plan |  |

### **4.3.2. Constraint**

No constraint has been discovered.

**Appendix A: Acronyms**

| Acronym | Translation |
| --- | --- |
| GUI | Graphic User Interface |
| CLI | Command Line Interface |

Table 1 - Acronyms

**Appendix B: References**

| Document Name | Document location or URL | Date |
| --- | --- | --- |
| Project Description | Description of Java project | 20/01/2020 |
| Betweenness centrality | <https://en.wikipedia.org/wiki/Betweenness_centrality> | 20/01/2020 |

Table 2 - Referenced Documents