# Module Interface Specification for Centrality in Graphs

Atiyeh Sayadi

March 2, 2024

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| Date 1 | 1.0 | Notes |
| Date 2 | 1.1 | Notes |

# 2 Symbols, Abbreviations and Acronyms

| symbol | description |
|---|---|
| AC | Anticipated Change |
| CC | Closeness Centrality |
| CIG | Centrality in Graphs |
| DAG | Directed Acyclic Graph |
| DC | Degree Centrality |
| M | Module |
| n | Total number of nodes |
| R | Requirement |
| SRS | Software Requirements Specification |
| UC | Unlikely Change |

# Contents

# 3 Introduction

The following document details the Module Interface Specifications for the centrality in graphs project. As stated earlier in various documents such as the SRS, this project aims to measure the centrality of each node in an undirected graph using two methods: degree centrality and closeness centrality.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/AtiyehSayadi/Centrality-In-Graphs/blob/main/docs/ProblemStatementAndGoals/ProblemStatement.pdf.

# 4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by CIG.

| Data Type | Notation | Description |
|---|---|---|
| character | char | a single symbol or digit |
| integer | $\mathbb{Z}$ | a number without a fractional component in (-$\infty$, $\infty$) |
| natural number | $\mathbb{N}$ | a number without a fractional component in [1, $\infty$) |
| real | $\mathbb{R}$ | any number in (-$\infty$, $\infty$) |

The specification of CIG uses some derived data types: matrix. A matrix is a collection of elements arranged in rows and columns within a rectangular array. Each element in the matrix can be any scalar value, such as real numbers, complex numbers, or variables. In addition, CIG uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

# 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

| Level 1 | Level 2 |
|---|---|
| Hardware-Hiding Module | - |
| Behaviour-Hiding Module | GUI<br>File |
| Software Decision Module | Degree<br>Closeness |

Table 1: Module Hierarchy

# 6 MIS of File

## 6.1 Module

File

## 6.2 Uses

N/A

## 6.3 Syntax

### 6.3.1 Exported Constants

N/A

### 6.3.2 Exported Access Programs

`read_file`

| Name | In | Out | Exceptions |
|---|---|---|---|
| `read_file` | - | Matrix | - |

## 6.4 Semantics

### 6.4.1 State Variables

N/A

### 6.4.2 Environment Variables

N/A

### 6.4.3 Assumptions

1: All elements of the input matrix must be numeric.

2: The output matrix is of size n*n.

### 6.4.4 Access Routine Semantics

`read_file`

- transition: -
- output: `new_matrix`
- exception: N/A

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

### 6.4.5 Local Functions

N/A

# 7 MIS of Degree

## 7.1 Module

Degree

## 7.2 Uses

File

## 7.3 Syntax

### 7.3.1 Exported Constants

N/A

### 7.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| `init` | File | - | - |
| `find_degree` | - | Matrix | - |
| `degree_centrality` | - | Matrix | - |

## 7.4 Semantics

### 7.4.1 State Variables

`__matrix`

### 7.4.2 Environment Variables

N/A

### 7.4.3 Assumptions

1: The degree of each node should not exceed or be equal to n- 1.

2: The degree centrality for each node should range between 0 and 1.

3: Output should be available for all nodes.

### 7.4.4 Access Routine Semantics

`init`

- transition: `__matrix`:= File
- output: -
- exception: N/A

`find_degree`

- transition: -
- output: degree_matrix
- exception: N/A

`degree_centrality`

- transition: -
- output: `degree_centrality_matrix`
- exception: N/A

### 7.4.5 Local Functions

N/A

# 8 MIS of Closeness

## 8.1 Module

Closeness

## 8.2 Uses

File

## 8.3 Syntax

### 8.3.1 Exported Constants

N/A

### 8.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| `init` | File | - | - |
| `shortest_path` | - | Matrix | - |
| `closeness_centrality` | | Matrix | - |

## 8.4 Semantics

### 8.4.1 State Variables

`__matrix`

### 8.4.2 Environment Variables

N/A

### 8.4.3 Assumptions

1: The sum of shortest paths to other nodes for each node should not be less than n- 1.

2: The closeness centrality for each node should range between 0 and 1.

3: Output should be available for all nodes.

### 8.4.4 Access Routine Semantics

`init`

- transition: `__matrix`:= File
- output: -
- exception: N/A

`shortest_path`

- transition: -
- output: shortest_path_matrix
- exception: N/A

`closeness_centrality`

- transition: -
- output: `closeness_centrality_matrix`
- exception: N/A

### 8.4.5 Local Functions

N/A

# 9 MIS of GUI

## 9.1 Module

GUI

## 9.2 Uses

File, Degree, Closeness

## 9.3 Syntax

### 9.3.1 Exported Constants

N/A

### 9.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| `init` | tkinter .Tk | - | - |
| `degree_centrality` | - | image | - |
| `closeness_centrality` | - | image | - |
| `show_grapth_matrix` | - | image | - |

## 9.4 Semantics

### 9.4.1 State Variables

N/A

### 9.4.2 Environment Variables

N/A

### 9.4.3 Assumptions

N/A

### 9.4.4 Access Routine Semantics

`init`

- transition: inteface:= tk.Tk
  inteface.title:= Show Graphs
  inteface.geometry:= 300x150
  `degree_button`:= Degree Centrality
  `closeness_button`:= Closeness Centrality
  `file_button`:= The Main Graph

- output: -
- exception: N/A

`degree_centrality`

- transition: -
- output: image
- exception: N/A

`closeness_centrality`

- transition:-
- output: image
- exception: N/A

`show_grapth_matrix`

- transition: -
- output: image
- exception: N/A

### 9.4.5 Local Functions

N/A

# References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

# 10 Appendix

[Extra information if required —SS]

# 11 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)