Module Interface Specification for Centrality in Graphs

Atiyeh Sayadi

March 10, 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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
matrix	$m^{i \times j}$	matrix with i rows and j columns
NodeT	$\{char, \mathbb{N}\}^+$	char, integer or both
EdgeT	tuple of (NodeT, NodeT)	-
Graph	set of EdgeT	-

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. Also, graph is a collection of nodes and edges, where each node represents elements within the network, and the edges represent the connections between them. 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 File Graph
Software Decision Module	Matrix Output

Table 1: Module Hierarchy

6 MIS of File

6.1 Module

File , Matrix, Hardware-Hiding

6.2 Uses

N/A

6.3 Syntax

6.3.1 Exported Types

NodeT= $\{char, \mathbb{N}\}^+$

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
`read_file`	String	$NodeT^{n\times 2}$	-

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

file: A sequence of memory bits

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

`read_file(s)`

- transition: -
- output: $out := \{\forall i : \mathbb{N} | i \in \{1..the\ number\ of\ distinguished\ elements\ in\ s\} \wedge (s1, s2) \in s : matrix\,[i, 0] = s1 \wedge matrix\,[i, 1] = s2\}$
- exception: -

6.4.5 Local Functions

N/A

7 MIS of Graph

7.1 Module

Graph

7.2 Uses

File, Matrix

7.3 Syntax

7.3.1 Exported Types

EdgeT= tuple of (start: NodeT, end: NodeT)
GraphT= set of EdgeT

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
`build_graph`	$NodeT^{n\times 2}$	-	-
`exist_edge`	EdgeT	$\mathbb{B}$	-
`get_degree`	NodeT	$\mathbb{N}$	-
`get_shortest_path`	NodeT, NodeT	$\mathbb{N}$	-
`degree_centrality`	NodeT	$\mathbb{R}$	-
`closeness_centrality`	NodeT	$\mathbb{R}$	-

7.4 Semantics

7.4.1 State Variables

nodes: set of NodeT
edges: GraphT

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

build_graph is called before any other access programs.

7.4.4 Access Routine Semantics

`build_graph(g)`

- transition: $edges := \{(s, e) : EdgeT | s, e \in g : (s, e)\}$, $nodes := \{s : T | s \in g : s\}$
- output: -
- exception: -

`exist_edge(s, e)`

- transition: -
- output: $out := \exists\{(s, e)\ EdgeT | (s, e) \in edges : s, e\}$
- exception:-

`get_degree(n)`

- transition: -
- output: $out := Sum(\{i, n : NodeT | (n, i) \in edges : 1\})$
- exception: -

`get_shortest_path(s,n)`

- transition: -
- output: $out := \{path : sequence\ of\ NodeT |\ path = \{s..n\} : |path|\ is\ minimum\}$
- exception: -

`degree_centrality(n)`

- transition: -
- output: $out := \{n : NodeT |\ n \in nodes : \frac{get\_degree(n)}{|nodes|-1}\}$
- exception: -

`closeness_centrality`

- transition: -
- output: $out := \{n : NodeT |\ n \in nodes : \frac{|nodes|-1}{\sum_{for\ \ each\ \ j\ \in nodes} get\_shortest\_path(n,j)}\}$
- exception: -

7.4.5 Local Functions

N/A

8 MIS GUI

8.1 Module

GUI

8.2 Uses

Graph, Output, Hardware-Hiding

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
`degree_centrality`	GraphT	-	-
`closeness_centrality`	GraphT	-	-

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

win: two-dimensional sequence of colored pixels

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

`degree_centrality(g)`

- transition: Adjusting the pixels of the win for displaying the graph g
- output: -
- exception: N/A

`closeness_centrality(g)`

- transition: Adjusting the pixels of the win for displaying the graph g
- output: -
- exception: N/A

8.4.5 Local Functions

N/A

9 MIS of Output

9.1 Module

Output

9.2 Uses

Hardware-Hiding

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
output	string, ℕ,..	GUI	-

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

10 MIS of Matrix

10.1 Module

Matrix

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
`Matrix`	string, $\mathbb{N}^{n\times n}$, list, ...	$\mathbb{N}^{n\times n}$, $\mathbb{N}$, ..	-

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

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.

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)