Module Interface Specification for Centrality in Graphs

Atiyeh Sayadi

April 7, 2024

1 Revision History

Date	Version	Notes
March 12, 2024	1.0	Initial draft

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 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 here.
In general, the design steps for this project include the following:
1. Designing a module to obtain the graph structure, which is essentially the adjacency matrix. Then, using the constructed graph, metrics such as node degree and the length of the shortest path from one node to another can be obtained. The outputs of these two functions are used, respectively, for Degree Centrality and Closeness Centrality.
2. Creating a module that reads the initial graph file and generates an output using the previously designed graph creation module, allowing a graphical user interface (GUI) to utilize it.
3. Additionally, the graphical user interface output needs to represent the obtained outputs in the form of an image, displaying a graph.

In the design of this software, two modules, namely GUI Provider and Matrix, have been considered. Since these two modules utilize the libraries Tkinter and NumPy respectively, serving as interfaces for these libraries, their interfaces will not be designed in this document.

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 (-∞, ∞)
Graph	$\mathbb{N}^{n \times n}$	

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	Show Graph File Graph
Software Decision Module	Matrix GUI Provider

Table 1: Module Hierarchy

6 MIS of File

6.1 Module

File

6.2 Uses

Graph

6.3 Syntax

6.3.1 Exported Types

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
`read_file`	string	GraphT	The file is empty.

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

file: Set of tuple(ℕ, ℕ)

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

`read_file(s)`

- transition: -
- output: A graph of the Graph module to access the graph properties through it.
- exception: When no file is uploaded or the file is empty, the function should issue a warning about the empty state of the variable "file".

6.4.5 Local Functions

N/A

7 MIS of Graph

7.1 Module

-

7.2 Uses

Matrix

7.3 Syntax

7.3.1 Exported Types

GraphT

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
GraphT	$\mathbb{N}$	GraphT	-
add_edge	$\mathbb{N}$	-	Inputs are out of range.
exist_edge	$\mathbb{N}$, $\mathbb{N}$	$\mathbb{B}$	-
get_degree	$\mathbb{N}$	$\mathbb{N}$	-
get_shortest_path	$\mathbb{N}$, $\mathbb{N}$	$\mathbb{N}$	-
degree_centrality	$\mathbb{N}$	$\mathbb{R}$	-
closeness_centrality	$\mathbb{N}$	$\mathbb{R}$	-

7.4 Semantics

7.4.1 State Variables

graph: $\mathbb{N}^{n \times n}$

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

GraphT is called before any other access programs.

7.4.4 Access Routine Semantics

`GraphT(g)`

- transition:
- output: A n*n zero matrix.
- exception: -

`add_edge(s, e)`

- transition: $graph[s, e] := 1,\ graph[e, s] := 1$
- output: -
- exception: If the indices are out of bounds, accessing the matrix is not possible, so in this case, an error alert is issued by the function.

`exist_edge(s, e)`

- transition: -
- output: $out := (graph[s, e] == 1)$
- exception:-

`get_degree(n)`

- transition: -
- output: $out := \sum_{j=0}^{|graph|-1} graph[n, j]$
- exception: -

`get_shortest_path(s,n)`

- transition: -
- output: $out := Min(|s, a1, a2, ..n|)$
  The shortest path between nodes s and n.
- exception: -

`degree_centrality(n)`

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

`closeness_centrality(n)`

- transition: -
- output: $out := \frac{|graph|-1}{\sum_{j=0}^{|graph|-1} get\_shortest\_path(n,j)}$
- exception: -

7.4.5 Local Functions

N/A

8 MIS Show Graph

8.1 Module

Show Graph

8.2 Uses

File

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
`show_degree_centrality`	GraphT	-	-
`show_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 for degree centrality
- output: -

- exception: N/A

`closeness_centrality(g)`

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

8.4.5 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.