UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS212 - Data structures and algorithms

Practical 9 Specifications: Graphs

Release date: 12-05-2025 at 06:00
Due date: 16-05-2025 at 23:59
Total marks: 90

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- You may not import any of the built-in Java data structures. Doing so will result in a zero mark. You may only use native 1-dimensional arrays where applicable.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with Java 8**
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `https://portal.cs.up.ac.za/files/departmental-guide/`.
- Please note that there is a late deadline which is 1 hour after the initial deadline, but you will lose 20% of your mark.

2 Overview

Graphs are important data structures used in Computer Science which is a collection of nodes called vertices connected by edges. These edges can have directions and weights in which case it is called a weighted directed graph. Graphs are usually stored using an array for the vertices and an array for the edges. For this practical you will be tasked using the alternative Adjacency Matrix representation.

3 Tasks

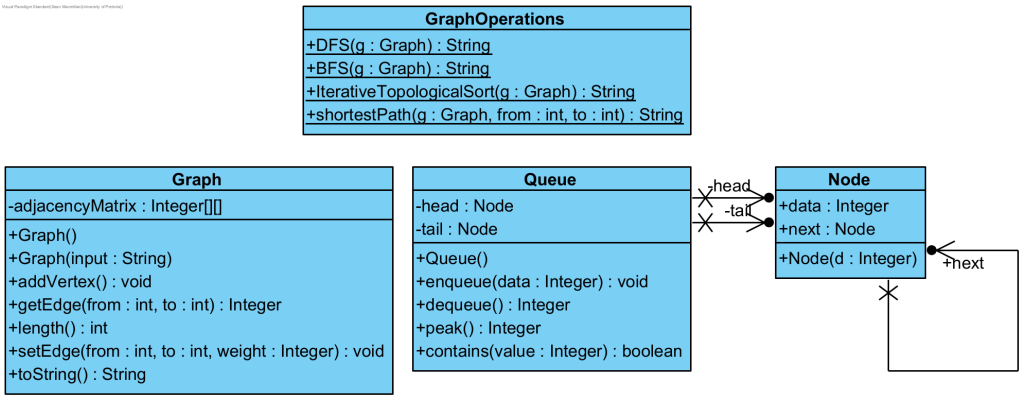

Figure 1: UML

3.1 Graph

- This class is given to you. Do not make any changes to since it will be overwritten on Fitchfork.
- This class is used to store a weighted directed graph using an adjacency matrix.
- The vertices do not have labels associated with them but only indices. Thus a graph with 8 vertices will have vertices 0 up to and including 7.
- If there is no edge between vertices then it will be represented by null. Note: it is possible for an edge to have a weight of 0, and weights can also be negative.
- Read through the comments in the file for detailed descriptions of each function.

3.2 Queue

- This class is given to you. Do not make any changes to since it will be overwritten on Fitchfork.
- This class can be used as a Queue which is needed for some graph algorithms.
- Read through the comments in the file for detailed descriptions of each function.

3.3 GraphOperations

3.3.1 Functions

- DFS
  - This function returns the traversal order of vertices when Depth First Search is performed on the passed-in graph.
  - After each vertex add a space to separate vertices. Note that this means there should be a space at the end of the result.
- BFS
  - This function returns the traversal order of vertices when Breadth First Search is performed on the passed-in graph.
  - After each vertex add a space to separate vertices. Note that this means there should be a space at the end of the result.
- TopologicalSort
  - This function returns the order of vertices from Iterative Topological Sort performed on the passed-in graph.
  - After each vertex add a space to separate vertices. Note that this means there should be a space at the end of the result.
  - The algorithm as was taught has parts where an error can occur due to Cycles in the passed-in graph. If this is the case return the String "Cycle Detected".
- ShortestPath
  - This function returns the shortest path from the start vertex to the end vertex in the provided graph. Use the Label Correcting Algorithm to calculate the path.
  - After each vertex add a space to separate vertices. Note that this means there should be a space at the end of the result.
  - If an invalid vertex is passed-in as a parameter return "Invalid vertex".
  - If no path exists between the start and end vertex then return "No path".

4 Testing

As testing is a vital skill that all software developers need to know and be able to perform daily, approximately 10% of the assignment marks will be assigned to your testing skills. To do this, you will need to submit a testing main (inside the Main.java file) that will be used to test an Instructor-provided solution. You may add any helper functions to the Main.java file to aid your testing. In order to determine the coverage of your testing the jacoco tool. The following set of commands will be used to run jacoco:

```
javac *.java
rm -Rf cov
mkdir ./cov
java -javaagent:jacocoagent.jar=excludes=org.jacoco.*,destfile=./cov/output.exec
    -cp ./ Main
mv *.class ./cov
java -jar ./jacococli.jar report ./cov/output.exec --classfiles ./cov --html
    ./cov/report
```

This will generate output which we will use to determine your testing coverage. The following coverage ratio will be used:

$$\frac{\text{number of lines executed}}{\text{number of source code lines}}$$

and we will scale this ratio according to the size of the class.
The mark you will receive for the testing coverage task is determined using table 1:

Coverage ratio range	% of testing mark
0%-5%	0%
5%-20%	20%
20%-40%	40%
40%-60%	60%
60%-80%	80%
80%-100%	100%

Table 1: Mark assignment for testing

Note the top boundary for the Coverage ratio range is not inclusive except for 100%. Also, note that only the function stipulated in this specification will be considered to determine your mark. Remember that your main will be testing the instructor-provided code and as such it can only be assumed that the functions outlined in this specification are defined and implemented.

5 Upload checklist

The following files should be in the root of your archive

- GraphOperations.java
- Main.java
- Any textfiles needed by your Main

6 Allowed libraries

- No imports allowed.

7 Submission

You need to submit your source files on the FitchFork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 8 standard.

For this practical, you will have 5 upload opportunities and your best mark will be your final mark. Upload your archive to the appropriate slot on the FitchFork website well before the deadline. **No late submissions will be accepted!**