Assignment 2: Graphs

COMP2003J: Data Structures and Algorithms 2
Lecturer: Dr. Ruihai Dong (ruihai.dong@ucd.ie)
Weight: **10% of final grade**
Document Version: 1.0

Introduction

The goal of this assignment is to program some graph implementations.

Download the file `Assignment-2-Source.zip` from Brightspace. The contents of this file include the following important classes and interfaces:

- All the interfaces you require for making a Graph. In particular, the `IGraph` interface includes comments describing all of the methods that a graph implementation should contain (these are in the `graph.core` package).
- An implementation of a Linked List, which you will need within your implementation (this is in the `graph.util` package and is named `DLinkedList`). You should not use built-in Java data structures for this assignment.
- An example of a Graph implementation: `EdgeListGraph`. You should study this file carefully, as the other implementations have some similar characteristics (this is in the `graph.impl` package).
- A program called `EdgeListTest` that shows some examples of code that can test some of the methods in the graph implementation (this is in the default package).

You are required to:

1. **Implement an Adjacency List graph (in a file called `AdjacencyListGraph`)**
2. **Implement an Adjacency Matrix graph (in a file called `AdjacencyMatrixGraph`).**

In each case, you should also create a new **testing** class similar to `EdgeListTest` to check that your implementation is correct. **Note:** The program I have provided does not test all of the methods in the graph implementation. You should add some more tests to check other methods (e.g. removing the vertex HNL should mean that the number of incident edges on LAX to decrease by one).

Assignment Project Quiz Exam Essay Help
WeChat: cestbon 6688
Email: acoder_overseas@163.com

Submission

This is an **individual programming assignment**. Therefore, all code must be written by yourself. Assignment 1 contained some advice about avoiding plagiarism in programming assignments.

- All code should be well-formatted and well-commented to describe what it is trying to do.

- Submit a single zip file to Brightspace, with the following contents:
  - The `AdjacencyListGraph`, `AdjacencyMatrixGraph`, `AdjacencyListTest` and `AdjacencyMatrixTest` classes.

  - If your testing code imports some other graphs from a text file, this text file may be included also. Do not include any extra Java files.

Assignment Project Quiz Exam Essay Help
WeChat: cestbon_688
Email: accoder_overseas@163.com

Assignment 2 Grading Rubric

This document shows the grading guidelines for Assignment 2 (Implementation of Adjacency List and Adjacency Matrix Graphs). Below are the main criteria that will be applied for the major grades (A, B, C, etc.). Other aspects will also be taken into account to decide minor grades (i.e. the difference between B+, B, B-, etc.), including:

- Readability and organisation of code (including use of appropriate functions, variable names, helpful comments, etc.).
- Quality of solution (including code efficiency, minor bugs, etc.).

Passing Grades

D Grade
Good implementation of an Adjacency List Graph or Adjacency Matrix Graph, plus some basic testing.

A "good" implementation is one where all the key methods work correctly in the vast majority of cases (i.e. some occasional bugs will be tolerated) and the code follows the right implementation strategy in most cases. Testing should not be simply to copy the sample tests for the Edge List graph; more tests must be added.

C Grade
Good implementation of an Adjacency List and Adjacency Matrix, plus some basic testing of both; OR
Good implementation of an Adjacency List or an Adjacency Matrix, plus comprehensive testing of the graph in question.
"Comprehensive'" testing should make sure that the different operations of the graph(s) are all tested (e.g. adding and removing vertices and edges, checking that correct vertices are adjacent or not, incident edges are correct, etc.). It should also check that the consequences of these operations are correct (e.g. removing a vertex removes its incident edges also, removing an edge means that its end vertices are no longer adjacent, etc.). The testing code should automatically detect whether a problem has occurred and can inform the user.

B Grade
Excellent implementation of an Adjacency List Graph or an Adjacency Matrix Graph, plus a good implementation of the other graph type, plus comprehensive testing of both graph types.
An "excellent'" implementation is one that always follows the correct implementation strategy, is written in well-organised and well-documented code and is almost entirely free of bugs.

A Grade
Excellent implementation of an Adjacency List Graph and an Adjacency Matrix Graph, plus comprehensive testing of both graph types.

Assignment Project Quiz Exam Essay Help
WeChat: cestbon_6888
Email: acoder_overseas@163.com

Failing Grades

ABS/NM Grade
No submission received/no relevant work attempted.

G Grade
Code does not compile; OR
Little or no evidence of meaningful work attempted.

F Grade
Some evidence of work attempted, but few (if any) methods operate in the correct manner.

E Grade
Adjacency List and/or Adjacency Matrix Graph have been attempted, but there are too many implementation errors for the implementation to be useful in practice; OR
Adjacency List and/or Adjacency Matrix Graph have been attempted, but the methods generally do not follow the correct strategy.

Assignment Project Quiz Exam Essay Help
WeChat: cestbon_688
Email: accoder_overseas@163.com