COMP202
Data Structures and Algorithms
Homework 4

Instructor: Alptekin Küpçü
Due Date: Jan 10 2025, 23:59
Submission Through: Github Classroom

This programming assignment will test your knowledge and your implementation abilities of what you have learned in the Graph parts of the class.

This homework must be completed individually. Discussion about algorithms and data structures are allowed but group work is not. **Coming up with the same algorithm and talking about the ways of implementation leads to very similar code which is treated as plagiarism!** If you are unsure, you should not discuss. Any academic dishonesty, which includes taking someone else's code or having another person doing the assignment for you will not be tolerated. **By submitting your assignment, you agree to abide by the Koç University codes of conduct.**

Description

This assignment requires you to implement:

- Graph
    - o Undirected Weighted Graph
- Graph search algorithms
    - o DFS
    - o BFS
    - o Dijkstra

The files provided to you have comments that will help you with implementation. In addition, keep the slides handy as they include the pseudo-codes for the algorithms. The file descriptions are below. Bold ones are the ones you need to modify and they are placed under the *code* folder, with the rest under *given*. The homework consists of three parts:

Graph Implementation

This part of the homework requires you to implement a given graph ADT for two different types of graphs. There is no explicit Edge class or an explicit Vertex class. This is done to give you total freedom. If you need extra classes such as these, feel free to put them as nested classes but do not provide extra files.

- *iGraph.java*: The interface that defines the graph ADT. Make sure you pay attention to all the comments!
- ***BaseGraph.java***: The abstract base class for the graphs that you should implement. You can put the common functions here. You may choose to use either adjacency list or adjacency matrix for graph representation.
- ***UndirectedWeightedGraph.java***: Implement an undirected-weighted graph in this file.
- *GraphTesting.java*: The file that includes the autograder implementation for this part. Can be run by itself.

Graph Algorithms

This part of the homework requires you to implement depth first search (DFS), breadth first search (BFS), Dijkstra's single-source all-destinations shortest path (or actually smallest cost) algorithm and cycle finding algorithms (for both directed and undirected graphs).

- ***GraphAlgorithms.java***: Implement the algorithms. You can (and probably should) add more methods and fields.
- *AlgTesting.java*: The file that includes the autograder implementation this part. Can be run by itself.

Grading

Your assignment will be graded through an autograder. Make sure to implement the code as instructed, use the same variable and method names. A version of the autograder is released to you. Our version will more or less be similar, potentially including more tests.
Run the main program in the *Grade.java* to get the autograder output and your grade.

Submission

Submissions will be through GitHub classroom.

Submission and General Instructions

- You will clone the template repository and work on that.
- Do not change the project structure that we provided you initially. You will work on the files under the **code** folder.
- Do not create new files. You are allowed to create new classes and helper functions. You can add these new classes or functions to the existing files.
- Do not touch the autograder files, otherwise your final grade might be inconsistent.
- Once you are done with the homework, do not forget to push your latest changes. Late submissions will be ignored.
- Make sure to remove all the unused imports and that you have the original package names. What compiles on your machine may not compile on ours due to simple import errors. Code that does not compile will receive a grade of 0.