GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 8
Due date: June 1, 2022– 23:55

Q1. (70 pts) Define a DynamicGraph interface by extending the Graph interface in the book for the following definition of graph data structure. Write a MyGraph class for the implementation of DynamicGraph interface.

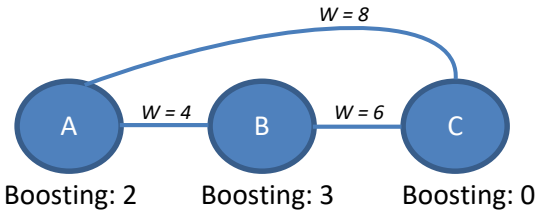
In your implementation, define a Vertex class for representing the vertices in the graph. A vertex must have an index (ID), a label, and a weight. The vertices may have user-defined additional properties (Vertex class should be generic), so you have to handle this requirement. Use adjacency list representation to handle the edges between vertices in the graph data structure.

DynamicGraph interface should have the following methods for manipulating the graph. For your implementation, analyze their execution time complexity in the report.

1. newVertex (string label, double weight): Generate a new vertex by given parameters.
2. addVertex (Vertex new_vertex): Add the given vertex to the graph.
3. addEdge (int vertexID1, int vertexID2, double weight): Add an edge between the given two vertices in the graph.
4. removeEdge (int vertexID1, int vertexID2): Remove the edge between the given two vertices.
5. removeVertex (int vertexID): Remove the vertex from the graph with respect to the given vertex id.
6. removeVertex (string label): Remove the vertices that have the given label from the graph.
7. filterVertices (string key, string filter): Filter the vertices by the given user-defined property and returns a subgraph of the graph.
8. exportMatrix(): Generate the adjacency matrix representation of the graph and returns the matrix.
9. printGraph(): Print the graph in adjacency list format (You should use the format that can be imported by the method in AbstarctGraph in the book).

Q2. (50 pts) Write a method that takes a MyGraph object as a parameter and performs BFS and DFS traversals. The method calculates the total distance of the path for accessing each vertex during the traversal, and it returns the difference between the total distances of two traversal methods. If there are more than one alternative to access a vertex at a specific level during the BFS, the shortest alternative should be considered. The vertices should be considered in distance order during DFS traversal, so, from a vertex v, DFS should continue with a vertex *w* which has the smallest edge from *v*, among all adjacent vertices of *v*.

Q3. **(60 pts)** Write a method that takes a MyGraph object and a vertex as a parameter to perform a modified version of Dijkstra's Algorithm for calculating the shortest paths from the given vertex to all other vertices in the graph. In this modified version, the algorithm considers boosting value of the vertices in addition to the edge weights. The boosting property is a user-defined property that takes double values. The boosting values are subtracted from the total length of paths that they are contained in. For example; the length of the path from *vertex A* to *vertex C* is calculated as $\min\big((4+6),8\big)=8$ in the regular algorithm, but the value of the path should be $\min\big((4+6-3),8\big)=7$ in the modified version. Please note that the boosting properties of the head and tail vertices in a path aren't considered by the algorithm.



GENERAL RULES:

- For any question firstly use the forum on the MS Teams page, and then the contact TA.
- You can submit an assignment one day late and will be evaluated by over sixty percent (%60).

TECHNICAL RULES:

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then, you must at least provide the following in the driver function:
  - Create an array list and add items to the list. Append items to the head, tail, and k^{th} index of the list.
  - Perform at least two different searches by using two items in the list and print the index of the items.
  - Perform another search with an item that isn't in the array list and inform the user that the item doesn't exist in the array list.
  - Delete an existing item from the list and repeat the searches.
  - Try to delete an item that is not on the array list and throw an exception for this situation.

  The driver function should run when the code file is executed.

- Implement clean code standards in your code;
  - Classes, methods, and variables names must be meaningful and related to the functionality.
  - Your functions and classes must be simple, general, reusable, and focus on one topic.
  - Use standard java code name conventions.

REPORT RULES:

- Add all javadoc documentations for classes, methods, variables …etc. All explanations must be meaningful and understandable.
- You should submit your homework code, Javadoc, and report to MS Teams in a "studentid_hw8.tar.gz" file.
- Use the given homework format including selected parts from the table below:

Detailed system requirements	✓
The Project use case diagrams	X
Class diagrams	✓
Other diagrams (extra points)	X
Problem solutions approach	✓
Test cases	✓
Running command and results	✓

GRADING :

- **No OOP design:** **-100**
- **No error handling:** **-50**
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- **Cheating:** **-200**
- Your solution is evaluated over 100 as your performance.

CONTACT :

- Teaching Assistant Burak Koca, b.koca@gtu.edu.tr