CS202, Fall 2024

Homework 4 - Graphs

Due: 23:59, 12/30/2024

Before you start your homework, please read the following instructions carefully:

FAILURE TO FULFILL ANY OF THE FOLLOWING REQUIREMENTS WILL RESULT IN A GRADE SCORE OF 0 (zero) WITHOUT ANY CHANCE OF REDEMPTION.

- See the course page for late submission policies and the Honor Code for Assignments.
- Upload your solutions in a single ZIP archive using the Moodle submission form. Name the file as studentID_name_surname_hw4.zip.
- Your ZIP archive should contain **only** the following files:
    - **studentID_name_surname_hw4.pdf**, the file containing the answers to Questions 1, and 2.
    - In this assignment, you must have separate interface and implementation files (i.e., separate .h and .cpp files) for your class. Your class name MUST BE **MolGraph** and your file names MUST BE **MolGraph.h** and **MolGraph.cpp**. Note that you may write additional class(es) in your solution.
    - Do not forget to put your name, student ID, and section number in all of these files. Add comments on your implementation well. Add a header (see below) to the beginning of each file:

      ```
      /**
       * Title: Graphs
       * Author : Name & Surname
       * ID: 12345678
       * Section : 1
       * Homework : 4
       * Description : description of your code
       */
      ```
    - Do not put any unnecessary files such as the auxiliary files generated from your preferred IDE.
    - Please do not use **Turkish** letters in your file and folder names.
- Your code must be **compilable**.
- Your code must be **complete**.
- You **ARE NOT ALLOWED** to use any data structure or algorithm-related function from the C++ standard template library **(STL)** or any other external libraries.
- Your code must contain the proper **comments** needed to understand its function.
- We will test your code using the **Google test library**. Therefore, the output message for each operation in Question 3 MUST match the format shown in the output of the example code.

- The code (main function) given in Question 3 is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you **MUST NOT** submit these test codes (we will use our own test code). In other words, **DO NOT submit** a file that contains a function called **main**.
- Your code must run on the **dijkstra.cs.bilkent.edu.tr** server. You can test your implementation using the tests from **/home/cs/sobhan.shukueian/Fall2024/Hw4_public directory.**
- For any questions related to the homework, contact your TA: *sobhan.shukueian@bilkent.edu.tr*

DO NOT START YOUR HOMEWORK BEFORE READING THIS SECTION CAREFULLY! THE INSTRUCTIONS MAY DIFFER FROM HOMEWORK TO HOMEWORK!

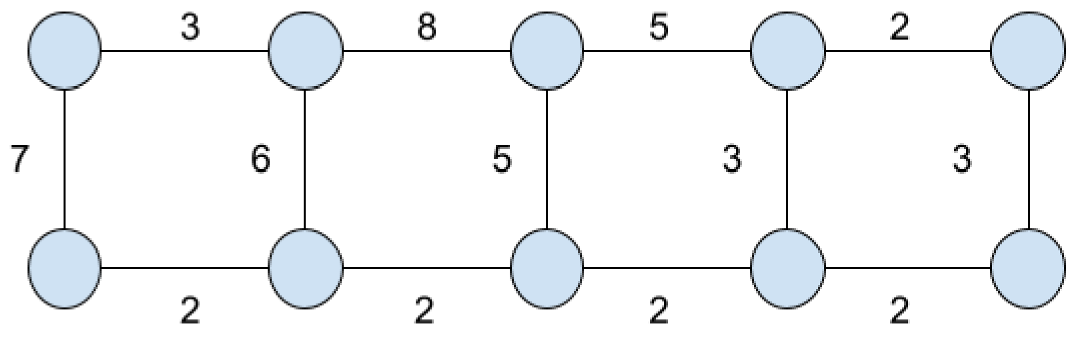
Question 1 (20 points)

Consider a graph with 10 vertices that is a complete graph, and its vertices are numbered from 1 to 10. If the following function determines the weight of each edge, what is the total weight of the edges in the minimal spanning tree of this graph?

$$w_{ij} = w_{ji} = \begin{cases} i+j & i+j \geq 5 \\ i^2+j^2 & i+j < 5 \end{cases}$$

Question 2 (15 points)

A company plans to set up a LAN (Local Area Network) among 10 rooms. The graph below represents the possible direct connections between the rooms, and the weight of each edge shows the cost of wiring between each pair of rooms. If the goal is to complete the wiring with minimal cost, what algorithm is most suitable for solving this problem? Apply it and write the steps.

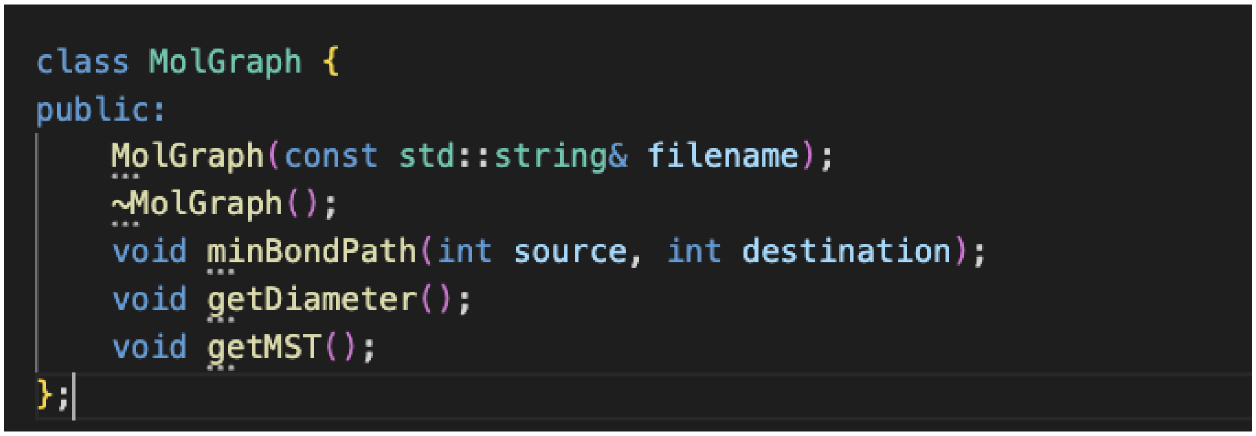


Question 3 (65 points)

A chemistry research institute wants to analyze the connectivity of molecules in a compound. In this task, you will simulate a system that organizes the bonds between atoms in a molecule. Specifically, your implementation will compute three metrics:

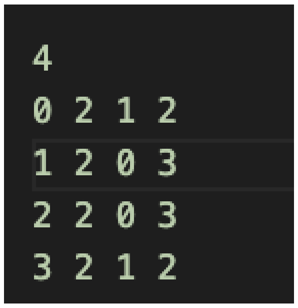
1. The minimum number of bonds must be traversed to move from one atom to another within the molecule.
2. The diameter of the molecule's connectivity graph.
3. The cost and structure of the Minimum Spanning Tree (MST) for the molecule's connectivity graph. The cost of the MST is the sum of weights (representing bond distances) of the edges included in the tree.

Your solution must be implemented in a class called MolGraph. Below is the required public part of the MolGraph class. The interface for the class must be written in a file called **MolGraph.h**, and its implementation must be written in a file called **MolGraph.cpp**. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution.



```
class MolGraph {
public:
    MolGraph(const std::string& filename);
    ~MolGraph();
    void minBondPath(int source, int destination);
    void getDiameter();
    void getMST();
};
```

You will be given a file representing the connectivity between atoms in a molecule. The first line of the input file specifies the number of atoms, and each subsequent line contains information about a specific atom. The format is as follows: <atom id: 1> <degree: 1> <neighbor id: N>, where <neighbor id: N> represents the IDs of N neighboring atoms connected by bonds. For example, consider the following input file for a molecule:



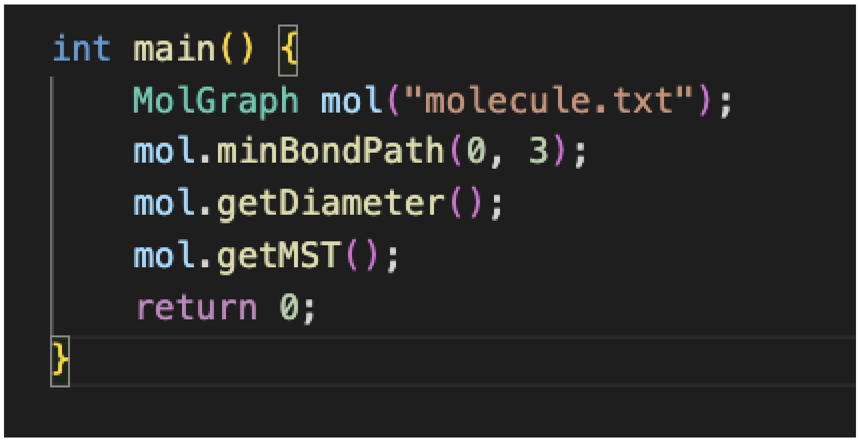
```
4
0 2 1 2
1 2 0 3
2 2 0 3
3 2 1 2
```

In this file:

- The first line indicates there are 4 atoms.
- The second line indicates that atom 0 has bonded to two other atoms (IDs 1 and 2

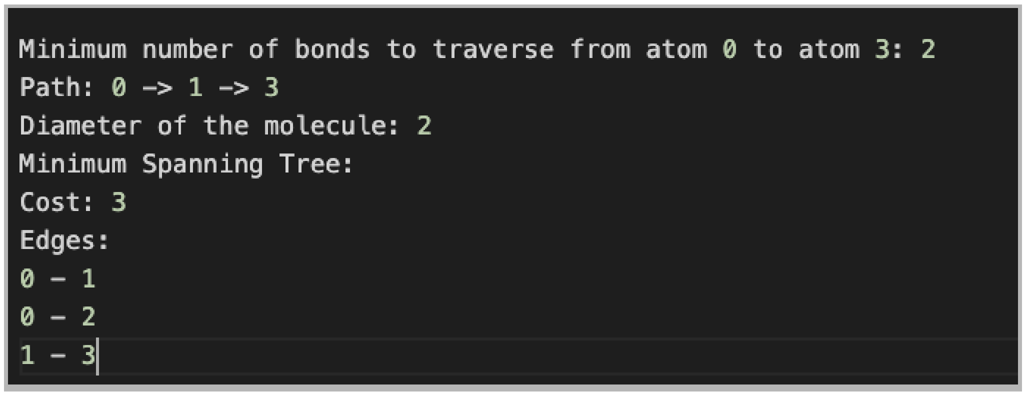
The output for the sample file above (assuming the input file is named molecule.txt) might look like this:

Test Example:



```
int main() {
    MolGraph mol("molecule.txt");
    mol.minBondPath(0, 3);
    mol.getDiameter();
    mol.getMST();
    return 0;
}
```

Test Output:



```
Minimum number of bonds to traverse from atom 0 to atom 3: 2
Path: 0 -> 1 -> 3
Diameter of the molecule: 2
Minimum Spanning Tree:
Cost: 3
Edges:
0 - 1
0 - 2
1 - 3
```

Note*:

- Assume all bond distances are equal to 1.
- Bonds between atoms are bidirectional. For example, if atom 0 is connected to atom 1, then atom 1 is connected to atom 0.
- In cases where multiple shortest paths or MST solutions exist, you only need to output one valid solution. No tie-breaking mechanism is required.
- Assume that the input file is always valid and that the atoms are labeled from 0 to N-1, where N is the number of atoms.
- Assume all atoms are connected, meaning no isolated atoms exist.

- The diameter of the graph is the longest shortest path between any two atoms.

Good luck! 😊