Master of Molecular Science and Software Engineering

Assignment #4: Graph Theory and Parallel Computations.

Chem 274 B - Introduction to Software Engineering

DUE Date: Saturday, December 2, 2023 by 11:30 PM U.S. Pacific Standard Time

Individual Assignment

Assignment #4.

The main goal of this assignment is to provide you with an opportunity to practice some of the concepts covered in the first six modules of this course. Key learning objectives (LOs) from these modules are:

LO1: Students deepen their understanding of data structures and graph theory and C++ implementations.

LO2: Students continue to learn and apply from best software engineering practices,

LO3: Students deepen their algorithmic and performance analysis skills

LO4: [optional] Students design and implement parallel algorithms.

There is only 1 problem worth 100 points in this assignment, and a bonus problem worth 20 points. Each problem has a set of questions and programming tasks for which you will need to submit C++ programs.

For your answers to the questions;

- You will write a paragraph or more making sure that you provide justifications for all your responses,
- You will submit a PDF file with your answers. Make sure to identify the questions that you are answering (e.g. “Problem 1. 2.1” or Problem 2 1.1, etc),
- You can use your favorite text editor (e.g., Overleaf, Google Doc, MS Word, a plain text editor, Latex, etc.) but **submit only the corresponding PDF file with your answers and name your PDF file: Answers-Assignment 4.pdf**.

For the C++ programs;

- You need to organize your programs using the best software engineering practices used in this course.
- Remember to document your programs well (e.g. using meaningful names for variables and functions, file names, include README files and relevant documentation inside your programs).
- Make sure your computer programs are readable (i.e., exercise writing computer programs with code readability in mind),
- Use makefiles to create executables, clean up directories, and testing.
- Use meaningful tests for verification and validation
- Identify and check for failure modes (e.g., failure to allocate arrays, input errors, etc.)

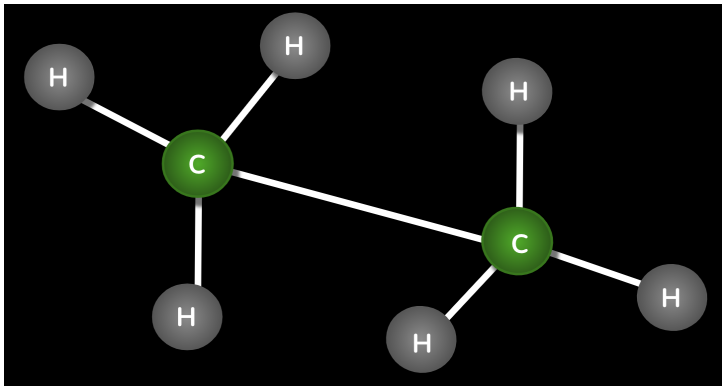

Figure 1. Ethane molecule C_2H_6. Each Carbon atom (green) has four chemical bonds and every Hydrogen atom has one chemical bond. The corresponding Ethane molecule graph has 8 vertices and seven edges.

1. **Problem I. Molecular Structures using Graphs** (*100 pts*)
   Graph theory, a branch of mathematics, is sometimes used when modeling chemical structures and molecular dynamics. In Chem 274A you have studied the representation of molecules using graphs (Problem set 3). There you used the Python library NetworkX to implement some key functionality in the assignment.
   In Chem 274B's Assignment 4, we will focus on the algorithmic analysis, and C++ Design implementation using very simple data structures that you will implement in C++. Therefore, do not use any C++ external libraries to implement the graph functionality.

   1.1. Graph Theory. Figure 1 illustrates an Ethane molecule, we can represent this by a graph that has 8 vertices and 7 edges. Notice that in this particular molecule, the atoms can be either Carbon or explicit Hydrogen atoms. To facilitate the graph manipulation, we number the vertices of the molecule as illustrated in Figure 2.

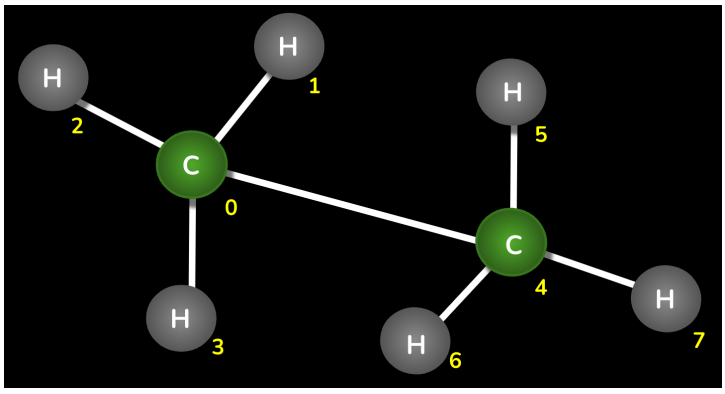

Figure 2. Graph representation of an Ethane molecule C_2H_6. All vertices have been labeled (in yellow) with a unique number, which serves as the vertex ID. See Appendix A for more information.

Problem: Design two C++ classes that can be used to represent molecular structures for computational modeling. A molecule has *m* different types of atoms, an atom has a *k* number of chemical bonds (e.g. a carbon atom has 4 chemical bonds) and atoms are represented by vertices in the graph. Recall that atoms are bound by edges of the graph (e.g., 0-2, 0-3, 0-1, 0-4, 4-6, 4-5, and 4-7 in Figure 2). Make sure to follow the best software engineering practices while designing and implementing your Molecule class and write meaningful tests.

1.1.1. **[LO1 and LO2]** Your first C++ class uses an adjacency matrix representation for a graph G = (V, E), where V = atoms and E = bonds,
1.1.2. **[LO1 and LO2]** Your second C++ class uses an adjacency-list representation for a graph G = (V, E), where V = atoms and E = bonds,
1.1.3. **[LO1 and LO2]** Write tests to verify that your C++ class implementations work.
1.1.4. **[LO3]:** Compare your 2 C++ classes implementation using spatial (memory) and temporal (time) performance analysis. The examples in this assignment are relatively small, so you will have to rely on asymptotic behavior of the underlying C++ classes as the number of atoms and bonds grow ($n \rightarrow \infty$)

1.2. **Breadth First Search (BFS)**. **Problem:**
1.2.1. **[LO1 and LO2]** Write a C++ implementation of the Bread First Search. Use your Molecule data structure as your base graph data structure. Make sure to follow the best software engineering practices while designing and implementing your Molecule class and write meaningful tests.
1.2.2. **[LO3]:** In your C++ implementation of BFS, which of your C++ classes did you use? Justify your choice well.

1.3. **Distance Matrix.** A distance matrix is a symmetric matrix which contains the shortest distances between every pair of vertices in the graph. The distance between a vertex to itself is 0. The Distance Matrix corresponding to Figure 1, is shown in Figure 3.
1.3.1. **[LO1 and LO2]:** Write a C++ implementation that computes the distance matrix for an input molecule. Make sure to follow the best software engineering practices while designing and implementing your Molecule class and write meaningful tests.
1.3.2. **[LO3]:** What is the algorithmic complexity of computing a distance matrix, provide an explanation for both the temporal (time to solution) and spatial (memory usages).

1.4. **Algorithmic Analysis. Question:** What is the algorithmic complexity of computing the distance matrix. Notice that to compute the distance matrix you need to use the BFS algorithm on every vertex of the graph and n is the number of vertices. Provide an analysis for both the temporal (time to solution) and spatial (memory usages).

Relevance: One application of the the distance matrix is in the computation of the *Weiner number, which* is the sum of all the distances between a pair of non-hydrogen atoms in a molecule (vertices in the graph representation of the molecule), because of its symmetry, one only needs to sum the matrix entries in either the upper diagonal (yellow) or lower

diagonal (blue) of the distance matrix. The Weiner number for the Ethane molecule in Figure 2 is 1.

	0	1	2	3	4	5	6	7
0	0	1	1	1	1	2	2	2
1	1	0	2	2	2	3	3	3
2	1	2	0	2	2	3	3	3
3	1	2	2	0	2	3	3	3
4	1	2	2	2	0	1	1	1
5	2	3	3	3	1	0	2	2
6	2	3	3	3	1	2	0	2
7	2	3	3	3	1	2	1	0

Figure 3. Distance matrix for the Ethane molecule display in Figure 2. Every row and column correspond to a vertex in the graph (i.e., an atom). For instance, for atom 0, its pair distances are recorded in the first column and first row of the distance matrix, 0→0 = 0, 0→1 = 1, 0→2 = 1, 0→3 = 1, 0→4 = 1, 0→5 = 2, 0→6 = 2, and 0→7 = 2.

---

2. **Problem II. *Parallel Matrix-Matrix multiplications*** (*20 pts*).
Matrix multiplications are at the heart of many computational applications (e.g. solution of linear and nonlinear systems of equations, machine learning algorithms, etc). In its most general form, the matrix-matrix product is written as:

$$C_{n\times m} = A_{n\times k} \times B_{k\times m} \qquad \textit{[Eq. 1]}$$

Special cases of Eq.1, is the matrix vector operation, where the matrix B has only one column (i.e. $m = 1$).

2.1. **[LO1 and LO2]: Problem:** Write a C++ implementation of the product of two real matrices. Make sure to check that the dimensions of the input matrices agree to compute the products. Make sure to follow the best software engineering practices while designing and implementing meaningful tests..

2.2. **[LO1, LO2 and LO4]: Problem:** Write a C++ Parallel implementation of a matrix-matrix product using Open-MP. Like before make sure to check that the dimensions of the input matrices agree to compute the products. Make sure to follow the best software engineering practices while designing and implementing meaningful tests.

2.3. **[LO3 and LO4]: Question:** How well do your parallel implementation scale? In other words, are you getting any speed-ups? Justify your answer by performing some relevant tests varying the size of the matrices, the number of threads (OpenMP version)?

APPENDIX A. Here are some molecules that you can use as input for your tests. Their Graph representations are listed here as input text files. Recall that a graph is defined as G = (V, E), where V is the set of vertices and E is the set of edges of the graph. The atoms in the molecule are represented by the vertices (or nodes) in the graph and the edges are their bonds.

For your tests, you will read a molecular structure from an input text file, with the following lines describing the atoms (vertices) and their corresponding chemical bonds (edges):

line 1: n, where n is the total number of atoms in the molecule
line 2: ID1 <atom 1 type> <atom 1 num bonds> **// Atom 1's description: ID, Type and # of bonds**
line 3: ID2 <atom 2 type> <atom 2 num bonds> **// Atom 2's description: ID, Type, and # of bonds**
: :
line n+1: IDn <atom n type> <atom n num bonds> **// Atom's description: ID, Type, and # of bonds**
line n+2: m, where m is the number of undirected edges in the molecules, the number of edges assigned to a vertex (atom) cannot exceed the atom's number of bonds.
line n+3: atom IDx, atom IDy **// Edge 1's description**
: :
line n+m+3: edge m: atom IDx, atom IDy **// Edge m's description**

A1. Example 1: Ethane molecule (as depicted in Figure 2) and the corresponding input text file will contain the following lines (filename **ethane.txt**):

```
8
0 'C' 4
1 'H' 1
2 'H' 1
3 'H' 1
4 'C' 4
5 'H' 1
6 'H' 1
7 'H' 1
7
0 1
0 2
0 3
0 4
4 5
4 6
4 7
```

In the above file, the first line has the number 8 because there are 8 atoms in the molecule and you need to read the next 8 lines containing the atom descriptions. The second line describes the first atom, which has an ID = 0, is a Carbon atom, and has four chemical bonds. Lines 3-9, describe the other 7 atoms. Line 10: has the number of chemical bonds (edges), 7 in the Ethane molecule. The next 7 lines describe the undirected chemical bonds 0-1, 0-2, 0-3, 0-4, 4-5, 4-6 and 4-7.

A2. Example 2: Octane molecule. This molecule is depicted in Figure 4.

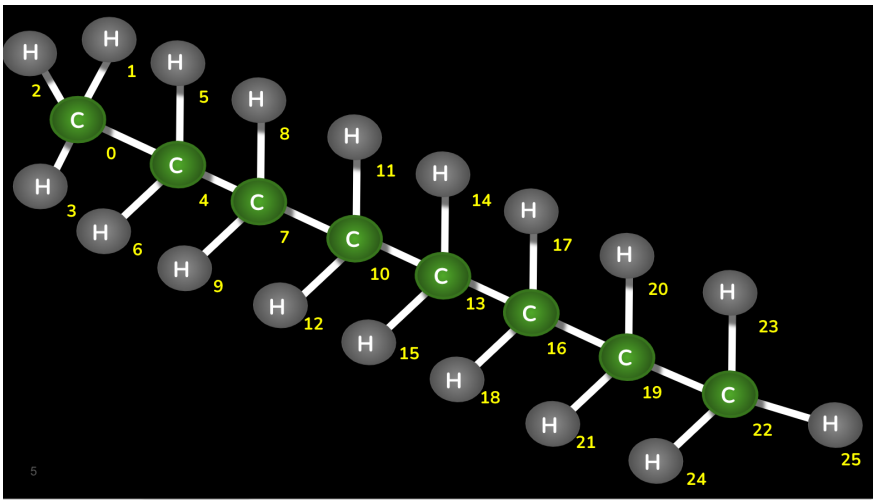

Figure 4. Graph representation of an Octane molecule C_8H_{18}. It has 8 Carbon atoms and 18 Hydrogen atoms, and 25 edges representing its bonds.

The Octane text file has the following lines (filename: **octane.txt**)

26
0 'C' 4
1 'H' 1
2 'H' 1
3 'H' 1
4 'C' 4
5 'H' 1
6 'H' 1
7 'C' 4
8 'H' 1
9 'H' 1
10 'C' 4
11 'H' 1
12 'H' 1
13 'C' 4
14 'H' 1
15 'H' 1
16 'C' 4
17 'H' 1
18 'H' 1
19 'C' 4
20 'H' 1

21 ‘H’ 1
22 ‘C’ 4
23 ‘H’ 1
24 ‘H’ 1
25 ‘H’ 1
25
0 1
0 2
0 3
0 4
4 5
4 6
4 7
7 8
7 9
7 10
10 11
10 12
10 13
13 14
13 15
13 16
16 17
16 18
16 19
19 20
19 21
19 22
22 23
22 24
22 25