

[CENG 315 ALL Sections] Algorithms

Dashboard / My courses / 571 - Computer Engineering / CENG 315 ALL Sections / December 3 - December 10 / THE6

Navigation

- ✓ Dashboard
 - ❖ Site home
 - > Site pages
- ✓ My courses
 - ✓ 571 - Computer Engineering
 - > CENG 223 All Sections
 - > CENG 223 Section 2
 - ✓ CENG 315 ALL Sections
 - > Participants
 - Competencies
 - Grades
 - > General
 - > October 2 - October 8
 - > October 9 - October 15
 - > October 15 - October 22
 - > October 23 - October 29
 - > October 30 - November 5
 - > November 6 - November 12
 - > November 13 - November 19
 - > November 20 - November 26
 - > November 27 - November 27
 - December 3
 - > December 3 - December 10
 - ✓ THE6
 - Description
 - Submission view
 - > December 11 - December 17
 - > December 18 - December 24
 - > December 25 - December 31
 - > January 1 - January 7
 - > January 8 - January 14
 - > CENG 315 Section 1

Description

Submission view

THE6

Available from: Saturday, December 16, 2023, 12:00 PM

Due date: Sunday, December 17, 2023, 11:59 PM

Requested files: the6.cpp, test.cpp, the6.h ([Download](#))

Type of work: Individual work

EDIT: If you are planning to work on your local, you can find the sample io [here](#).

Input files are formatted as follows:

```
<node_count> <edge_count> followed by the <vertex1> <vertex2> <weight> lines for the bond_energies graph  
<node_count> <edge_count> followed by the <vertex1> <vertex2> <weight (always1)> lines for the molecule_structure graph
```

Output files are formatted as follows:

```
<total_lowest_energy>  
<longest_chain>
```

Only numerical values are compared as fixed outputs. Since the structures may have possible correct answers, they are evaluated by the solver in the VPL.

You, an AI hater, recently learned that an AI model discovered new molecule structures equivalent to 800 years' worth of knowledge[1][2], and you are extremely annoyed by this news. While on an anger rant, you claim that you can write a basic program to find new optimal structures by yourself. Your chemist friends dare you to do so. They also offer to check the validity of your program, if you also write them another program that helps with their project.

You come up with an idea to find new molecule structures: If we have a graph where each vertex is an atom and each edge is the energy of the bond between the two atoms, maybe a new possible molecule structure can be found by selecting the edges with lowest energy bonds, without creating a cycle. Your chemist friends tell you this is most definitely a wrong approach, but you insist on trying.

Additionally, your chemist friends need a program to find the longest chain in a given molecule, so you agree to also write that program for them.

Problem

This exam consists of two parts, graded independently.

> PART1

You need to complete the function `find_structure()` which returns the total bond energy of the found molecule:

```
int find_structure(std::vector< std::vector<std::pair<int,int> > > &bond_energies,  
std::vector< std::vector<std::pair<int,int> > > &lowest_energy_structure);
```

- **bond_energies:** a bidirectional graph represented by an adjacency list, where the vertices (**v**) are atoms, edges are bonds, and weights (**w**) of edges are the energies of the bonds. In other words, an entry in the adjacency list for vertex v_0 is $\langle v_1, w \rangle$, which represents an edge between v_0 - v_1 with a weight of w .
- **lowest_energy_structure:** the found molecule structure represented as an adjacency list, with the same format as **bond_energies**. You are expected to find the structure and assign it to this argument.
- There is at most one bond with two atoms with only one weight value, i.e. there is at most one undirected edge between any two vertices in the graph, meaning at most one weight value for each pair. Unidirectionality is shown in the adjacency list by adding two mirror entries for each edge for ease of implementation.
- Vertices are represented as integers starting from 0, and the **maximum number of vertices in the graph is 1000**.
- Weights are represented as integers starting from 1, and the **maximum weight value is 100**.
- **return value is the total bond energy of the lowest_energy_structure.**

> PART2

You need to complete the function `find_longest_chain()` which returns the number of the atoms in the longest chain of the given molecule.

```
int find_longest_chain(std::vector< std::vector<std::pair<int,int> > > &molecule_structure,  
std::vector<int> chain);
```

- **molecule_structure:** a graph with the same representation and limits as the **bond_energies** and **lowest_energy_structure** arguments of PART 1. **Additionally, it is guaranteed that the structure is a connected, acyclic, undirected graph, with all edges having the same weight (i.e. weights are not important).** Unidirectionality is shown in the adjacency list by adding two mirror entries for each edge for ease of implementation.
- **chain:** a vector of integers, where each integer maps to a vertex ID in the found longest chain. The vector should follow the order of the chain. The validity of the chain will be checked by the tester to see if consecutive vertices in the vector have edges between them or not.
- **return value is the total number of atoms in the longest chain of the molecule. The longest chain is defined as the count of vertices on the path between the two farthest vertices in the graph, including the start and end vertices.**

Constraints and Hints:

- When deciding between edges with the same weights connecting v to two different vertices $u_{i,1}$ and $u_{i,2}$, always favor the $u_{i,j}$ with a smaller index.
- Similarly, if you need to decide between two edges $(u_{i,1}, u_{i,2})$ and $(u_{j,1}, u_{j,2})$ where $u_{i,1} = u_{j,1}$, you should again favor the edge with the smaller $u_{i,2}$.

Similarly, if you need to choose between two edges (u_1, u_2, w_1) and (u_2, u_3, w_2) where $w_1 < w_2$, you should again take the edge with the smaller weight, meaning (u_1, u_2, w_1) .

- There are **no self-loops**, meaning there is no edge such that (u_i, u_j, w) where $i=j$.

Evaluation:

- After your exam, black-box evaluation will be carried out. You will get full points if you return the correct order or strongly connected components. The grade you see in the VPL contains 50% of your final grade. We will evaluate your grades with different inputs after the end of the exam.

Grade distribution is as follows:

- 30% part1 return value **lowest_total_energy**
- 30% part1 **lowest_energy_structure**
- 10% part2 return value **longest_chain_size**
- 30% part2 **chain**

Example IO:

1)
> PART 1

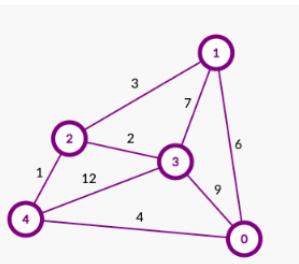


fig 1: bond_energies graph

bond_energies:

- ```
0 { (1,6) (3,9) (4,4) }
1 { (0,6) (2,3) (3,7) }
2 { (1,3) (3,2) (4,1) }
3 { (0,9) (1,7) (2,2) (4,12) }
4 { (0,4) (2,1) (3,12) }
```

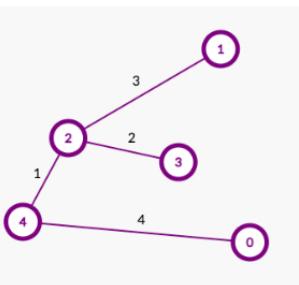


fig 2: lowest\_energy\_structure

#### lowest\_energy\_structure found:

- ```
0 { (4,4) }
1 { (2,3) }
2 { (1,3) (4,1) (3,2) }
3 { (2,2) }
4 { (2,1) (0,4) }
```

return value **lowest_total_energy**: 10

> PART 2

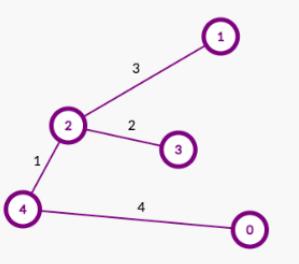


fig3: molecule_structure

note: for the first i/o, input of part2 is the same as the output of part1.
this is not guaranteed to be the case for all outputs. please check other i/o.

molecule_structure:

- ```
0 { (4,4) }
1 { (2,3) }
2 { (1,3) (3,2) (4,1) }
3 { (2,2) }
4 { (2,1) (0,4) }
```

```

chain:
(3)-(2)-(4)-(0) and alternatively, (1)-(2)-(4)-(0)

return value longest_chain_size: 4

```

**Specifications:**

- There are 2 tasks to be solved in **36 hours** in this take-home exam.
- You will implement your solutions in **the6.cpp** file.
- You are free to add other functions to **the6.cpp**
- **Do not change** the first line of the6.cpp, which is #include "the6.h"
- Some libraries are included in "the6.h" for your convenience, you can use them freely.
- **Do not change** the arguments and the return value of the functions **find\_structure()** and **find\_longest\_chain()** in the file the6.cpp
- **Do not include** any other library or write include anywhere in your the6.cpp file (not even in comments).
- You are given **test.cpp** file to test your work on **ODTUClass** or your **locale**. You can, and you are, encouraged to modify this file to add different test cases.
- If you want to test your work and see your outputs you can compile your work on your locale as:

```

>g++ test.cpp the6.cpp -Wall -std=c++11 -o test
> ./test

```

- You can test your the6.cpp on the virtual lab environment. If you click **run**, your function will be compiled and **executed with test.cpp**. If you click **evaluate**, you will get **feedback** for your current work, and your work will be **temporarily graded** for a limited number of inputs.
- The grade you see in lab is not your final grade, **your code will be reevaluated with different inputs** after the exam.

The system has the following limits:

- a maximum execution time of 8 seconds (your program needs to return in less than a second per test case on average - each test case has 2 parts)
- a 1 GB maximum memory limit,
- an execution file size of 4M.
- Solutions with longer running times will not be graded.
- If you are sure that your solution works in the expected complexity, but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.

\*not relevant to the question:

- [1] <https://deepmind.google/discover/blog/millions-of-new-materials-discovered-with-deep-learning/>  
[2] <https://www.nature.com/articles/s41586-023-06735-9>

## Requested files

### the6.cpp

```

1 #include "the6.h"
2
3 // do not add extra libraries here
4
5 int find_structure(std::vector< std::vector< std::pair<int,int> > >& bond_energies, std::vector< std::vector< std::pair<int,int> > >& lowest_energy_structure)
6 int lowest_total_energy = 0;
7
8 return lowest_total_energy;
9 }
10 int find_longest_chain(std::vector< std::vector< std::pair<int,int> > >& molecule_structure, std::vector<int>& chain){
11 int longest_chain_size = 0;
12 return longest_chain_size;
13 }
14

```

### test.cpp

```

1 #include <iostream>
2 #include <fstream>
3 #include "the6.h"
4
5
6 void print_adj_list(std::vector< std::vector< std::pair<int,int> > >& adj_list) {
7 int N = adj_list.size();
8 if (N == 0) {
9 std::cout << "list is empty!" << std::endl;
10 return;
11 }
12
13 for (int v=0;v<N;v++) {
14 std::cout << v << "\t";
15 for (auto p : adj_list[v]) {
16 std::cout << "(" << p.first << "," << p.second << ")";
17 }
18 std::cout << ")"\n";
19 }
20 return;
21 }
22
23 // you can use this if you want to print the adj list as a matrix
24 void print_adj_list_as_matrix(std::vector< std::vector< std::pair<int,int> > >& adj_list) {
25 int N = adj_list.size();
26 if (N == 0) {
27 std::cout << "list is empty!" << std::endl;
28 return;
29 }
30 int** matrix;
31 matrix = new int*[N];
32 for(int temp=0; temp < N; temp++) matrix[temp] = new int[N];
33 for (int i=0; i<N; i++){
34 for (int j=0; j<N; j++){
35 matrix[i][j] = -1; // no edge
36 }
37 }

```

```

38
39 for (int i=0; i<N; i++){
40 for (std::pair<int,int> x: adj_list[i]) {
41 matrix[i][x.first] = x.second;
42 }
43 }
44
45 for (int i=0; i<N; i++){
46 for (int j=0; j<N; j++){
47 if (matrix[i][j] == -1) std::cout << "- ";
48 else std::cout << matrix[i][j] << " ";
49 }
50 std::cout << std::endl;
51 }
52
53 for(int i=0; i<N; i++) delete[] matrix[i];
54 delete[] matrix;
55 return;
56 }
57
58 void read_from_file(std::vector< std::vector< std::pair<int,int> > > & bond_energies, std::vector< std::vector< std::pair<int,int> > > & molecule_structure){
59 char addr[] = "inp01.txt"; // 01-05 are available
60 std::ifstream infile (addr);
61 if (!infile.is_open()){
62 std::cout << "File \"<< addr
63 << "\" can not be opened. Make sure that this file exists." << std::endl;
64 return;
65 }
66
67 int V_p1, E_p1, V_p2, E_p2;
68
69 infile >> V_p1 >> E_p1;
70 bond_energies.resize(V_p1);
71 for (int l=0; l<E_p1; l++) {
72 int v1, v2, w;
73 infile >> v1 >> v2 >> w;
74 bond_energies[v1].push_back(std::make_pair(v2,w));
75 bond_energies[v2].push_back(std::make_pair(v1,w));
76 }
77
78 infile >> V_p2 >> E_p2;
79 molecule_structure.resize(V_p2);
80 for (int l=0; l<E_p2; l++) {
81 int v1, v2, w;
82 infile >> v1 >> v2 >> w;
83 molecule_structure[v1].push_back(std::make_pair(v2,w));
84 molecule_structure[v2].push_back(std::make_pair(v1,w));
85 }
86
87
88 infile.close();
89 }
90
91 int main(){
92 std::vector< std::vector< std::pair<int,int> > > bond_energies;
93 std::vector< std::vector< std::pair<int,int> > > molecule_structure;
94 std::vector< std::vector< std::pair<int,int> > > lowest_energy_structure;
95 std::vector<int> chain;
96 int longest_chain_size, lowest_total_energy;
97
98 read_from_file(bond_energies, molecule_structure);
99 lowest_energy_structure.resize(bond_energies.size());
100
101 lowest_total_energy = find_structure(bond_energies, lowest_energy_structure);
102 std::cout << "PART 1: " << std::endl << "Bond energy graph:" << std::endl;
103 print_adj_list(bond_energies);
104 std::cout << "Graph of the lowest energy structure found:" << std::endl;
105 print_adj_list(lowest_energy_structure);
106 std::cout << "Total energy of the lowest energy structure: " << lowest_total_energy << std::endl;
107
108 longest_chain_size = find_longest_chain(molecule_structure,chain);
109 std::cout << "PART 2: " << std::endl << "Molecule structure graph:" << std::endl;
110 print_adj_list(molecule_structure);
111 std::cout << "Atom count in longest chain: " << longest_chain_size << std::endl;
112 std::cout << "Longest chain: " << std::endl;
113 std::cout << "(" << chain[0] << ")";
114 for (int i=1; i<chain.size(); i++) {
115 std::cout << " - (" << chain[i] << ")";
116 }
117
118 return 0;
119 }

```

### the6.h

```

1 #ifndef THE6_THE6_H
2 #define THE6_THE6_H
3 #include <vector>
4 #include <utility>
5 #include <queue>
6 #include <stack>
7 #include <climits>
8 #include <algorithm>
9
10 //updating this file will not change the execution in the VPL
11
12 int find_structure(std::vector< std::vector< std::pair<int,int> > > &bond_energies, std::vector< std::vector< std::pair<int,int> > > &lowest_energy_structure)
13
14 int find_longest_chain(std::vector< std::vector< std::pair<int,int> > > &molecule_structure, std::vector<int>& chain);
15
16
17 #endif //THE6_THE6_H
<

```

VPL

You are logged in as mustafa baris emektar ([Log out](#))

CENG 315 ALL Sections

Get the mobile app

