Middle East Technical University | Department of Computer Engineering

CENG 315

Algorithms

Fall 2024-2025

Take-Home Exam 6

Due date: 15 December 2024, Sunday, 23.59

1 Problem Definition

After a security breach occurred and some data was stolen from an email provider, you found a network of friends using this provider for communication. The network shows the amount of time friends sent emails to each other or themselves.

You want to analyze the network to understand the relationships of these friends. You define a metric called *Friendship Score (FS)* that uses the shortest paths between nodes as follows:

$$FS(n1, n2) = \frac{SP_1(n1)}{SP_2(n1, n2)} * \frac{SP_1(n2)}{SP_2(n2, n1)}$$

where:

- the length of a path is the total weight of the edges on that path,
- the distance of a node to itself is $infinity$ unless there is a self-edge or a cyclic path back to the node itself,
- $SP_1(n)$ is the length of the shortest path from node n to any node (including itself),
- $SP_2(n1, n2)$ is the length of the shortest path between nodes $n1$ and $n2$,
- $FS(n1, n2)$ is 0 if only one of the friends has emailed the other one (i.e. either $SP_2(n1, n2)$ or $SP_2(n2, n1)$ is $infinity$),
- $FS(n1, n2)$ is -1 if none of the friends has emailed the other one.

Given the friendship network as a directed weighted graph, you are expected to calculate the friendship score between each node pair. The graph is structured in a way where a directed weighted edge between Friend1 and Friend2 shows the number of emails Friend1 sent to Friend2 as the weight of that edge.

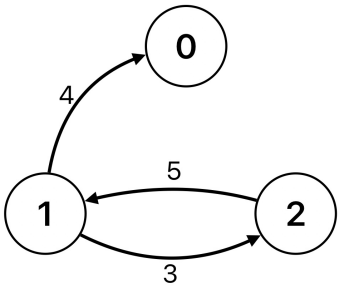
```
std::vector< std::vector<float> > get_friendship_scores(
            const std::vector< std::vector< std::pair<int, int> > >& network);
```

In this exam, you are expected to implement the function `get_friendship_scores`. The friendship network is given to you in `network` as an adjacency list with weights. You should calculate the $FriendshipScore$ for each pair in the network and return the `friendship_scores` accordingly.

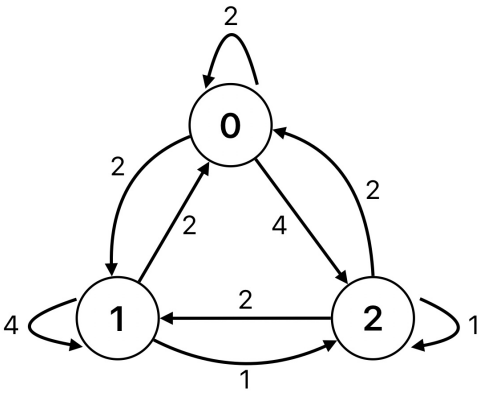
1.1 Example I/O

```
INPUT
_____
network: // node1: (node2, weight)
0:      {}
1:      { (0, 4)  (2, 3) }
2:      { (1, 5) }

OUTPUT
______
friendship scores:
FS(0,0) = -1
FS(1,0) = 0
FS(1,1) = 0.140625
FS(2,0) = 0
FS(2,1) = 1
FS(2,2) = 0.390625
```

```
INPUT
_____
network: // node1: (node2, weight)
0:      { (0, 2)  (1, 2)  (2, 4) }
1:      { (0, 2)  (1, 4)  (2, 1) }
2:      { (0, 2)  (1, 2)  (2, 1) }

OUTPUT
______
friendship scores:
FS(0,0) = 1
FS(1,0) = 0.5
FS(1,1) = 0.111111
FS(2,0) = 0.333333
FS(2,1) = 0.5
FS(2,2) = 1
```



(a) friendship network of first I/O



(b) friendship network of second I/O

2 Limits and Specifications

In this exam, the complexity of your implementations will be checked by the run of your implementations staying within the VPL limits. Hence the system and variable limitations are as follows:

- at least 4, at most 500 nodes in the `network` graph,
- 16 seconds as the maximum execution time,
- a maximum memory limit of 1 GB,
- and a stack size of 4 MB for function calls (ie. recursive solutions)

Additionally:

- You will implement your solutions in the `the6.cpp` file. Do not change the first line of `the6.cpp`, which is `#include "the6.h"`.
- Do not change the arguments and the return value of the given functions in the file `the6.cpp`, but you are free to add other functions to `the6.cpp`. However, do not include any other library or write include anywhere in your `the6.cpp` file (not even in comments).
- You are given a `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.
- 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 graded with a limited set of inputs.
- If you want to test your work and see your outputs on your locale you can use the following commands:

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

3 Regulations

- **Implementation and Submission:** The template files are available in the Virtual Programming Lab (VPL) activity called "THE6" on ODTUCLASS. At this point, you have two options:
  - You can download the template files, complete the implementation, and test it with the given sample I/O on your local machine. Then submit the same file through this activity.
  - You can directly use the editor of the VPL environment by using the auto-evaluation feature of this activity interactively. Saving the code is equivalent to submitting a file.
- **Programming Language:** You must code your program in C++. Your submission will be tested on the VPL environment in ODTUCLASS, hence you are expected to make sure your code runs successfully there.
- **Cheating: This assignment is designed to be worked on individually.** Additionally, the use of any LLMs (chatgpt, copilot, the other one that you are thinking about...) and copying code directly from the internet for implementations is strictly forbidden. Your work will be evaluated for cheating, and disciplinary action may be taken if necessary.
- **Evaluation:** Your program will be evaluated automatically using "black-box" testing, so make sure to obey the specifications. No erroneous input will be used. Therefore, you don't have to worry about invalid cases.