Middle East Technical University | Department of Computer Engineering

CENG 315

Algorithms

Fall 2024-2025

Take-Home Exam 7

Due date: 29 December 2024, Sunday, 23.59

1 Problem: *Longest* Path to Home

This programming exam is designed to introduce you to the longest path problem, which is the problem of finding a simple path of maximum length between two nodes given a graph. Unlike the shortest path problem, the problem of finding the longest path in a graph (without negative-weight cycles) cannot be solved in polynomial time - except for directed acyclic graphs.

It is the holiday season and your parents are expecting you to come home. You and some of your friends from the same city decided to travel back home together. However, to spend more time together, you decided to travel from the city you study to your hometown using the longest possible path.

You will be given a directed graph of all the roads in the area where the nodes are the towns, edges between nodes are the roads, and weights of the edges are the lengths of the roads. Your task is to calculate the longest path from the given `starting_node` and `destination_node` by implementing the `find_longest_path` function.

```
std::vector<int> find_longest_path(
                const std::vector<std::vector<std::pair<int, int> > >& graph,
                int starting_node,
                int destination_node,
                long& length_of_longest_path);
```

The graph is given to you in `graph` as an adjacency list with weights. You should find the longest path and update the `length_of_longest_path` variable with the length of the longest path you have found, and return the path as a vector of integers where each integer represents the id of the node, ordered by their order in the path starting from `starting_node` and ending in `destination_node`.

You are expected to implement two approaches to this problem in the same function. Firstly, you should check if the given graph is a DAG, if not, you should find the longest path by an exhaustive search - which is a non-polynomial solution. However, if the given graph is a DAG, you should figure out a polynomial-time solution.

Some limitations and hints:

- There will be at max one edge between two nodes in the same direction.
- There will be no negative edges since the roads cannot have negative lengths.
- You cannot visit a node more than once.
- You can return any path with the same longest path length between the starting and destination nodes as long as it is a valid path. Your paths will be checked by their length and validity during grading.
- Complexity of your solutions will be checked by the runtime of your implementations.
- You have different time limits for different inputs. For the non-DAG cases, the inputs have smaller node and edge counts and the time limit per case is 2 seconds. For the DAG cases, you have 0.5 seconds per case but the input sizes are way larger.

Grading:

- Per case:
  - 4 points - the path length is correct
  - 6 points - the given path is a valid path with the given length
- Overall:
  - 40 points - if the non-polynomial exhaustive solution is correct
  - 60 points - if the polynomial solution for DAGs is correct

1.1 Example I/O

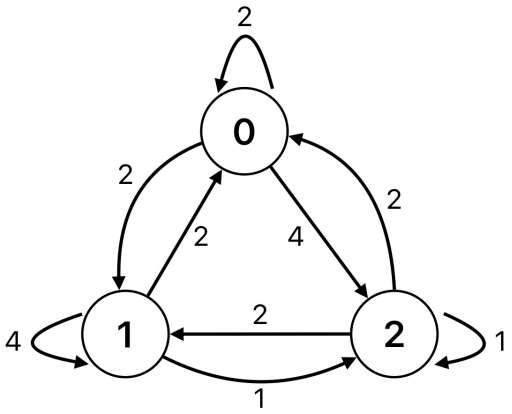

Figure 1: road graph of example I/O

```
input
------
road graph:
0:      { (0, 2)  (1, 2)  (2, 4) }
1:      { (0, 2)  (1, 4)  (2, 1) }
2:      { (0, 2)  (1, 2)  (2, 1) }
starting node: 0
destination node: 2

output
------
longest path length: 4
one possible longest path:
0 - (4) -> 2
```

2 Specifications

- You will implement your solutions in the `the7.cpp` file. Do not change the first line of `the7.cpp`, which is `#include "the7.h"`.
- Do not change the arguments and the return value of the given functions in the file `the7.cpp`, but you are free to add other functions to `the7.cpp`. However, do not include any other library or write include anywhere in your `the7.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 `the7.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 the7.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 "THE7" 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.