BLG336E - Analysis of Algorithms II

Homework 1

Due Date: 28.03.2023, 23:59 PM

1 Introduction

A group of **n** kids are playing a game of catching a ball in a park. Their location in the park are mapped into a grid space with (x_i, y_i) coordinates. Also, each kid has a different strength level that is represented as p_i for i-th kid. The rules of the game are as following.

- Kids do not change their locations in the grid space.
- The i-th kid can throw the ball to the j-th kid **directly** if and only if:
  - the squared distance between i-th kid and j-th kid is smaller then the strength of i-th kid,
  - j-th kid should be able to send the ball back to the i-th kid.

In this homework, it is expected from you to implement a graph (Section 2.1) considering the rules above. After that, you should traverse the graph to solve the problems in Section 2.2 and 2.3 by using breadth first search and depth first search algorithms.

2 Code (70 Points)

2.1 Graph Implementation

Implement the code which creates **(n x n)** adjacency matrix as a graph, $\mathbf{G = (V, E)}$, according to the rules given in the Introduction section. Each kid should be represented by a vertex. An edge represents a possible direct pass in between two kids.

2.2 Breadth First Search

Let's imagine a scenario that kids want to deliver the ball starting from the i-th kid (source kid) to the j-th (target kid). Consider a **pass** is defined as a **direct** throw in between two kids.

What is the amount of the **minimum passes** to deliver the ball from the i-th **source** kid to the j-th **target** kid?

NOTES:

- You have to use the graph structure that you implemented before.
- You have to traverse the graph with BFS algorithm.
- When you traverse the graph and decide which node is the next node from the current node, you have to choose node with the lower index number.

2.3 Depth First Search

Let's imagine that a source kid started the game by passing the ball to another kid. Assuming that each kid can pass the ball maximum one time, is it possible for the ball to come back to the source kid in a cycle? If so, print out the pass route starting from the source kid.

NOTES:

- You have to use the graph structure that you implemented before.
- You have to traverse the graph with DFS algorithm.
- When you traverse the graph and decide which node is the next node from the current node, you have to choose node with the lower index number.

3 Compile & Run

Your program should compile using the following command:

g++ -std=c++11 -Wall -Werror main.cpp -o main

Also, your program should run using the following command:

./main <input.txt>

Contents of the input.txt will be as following:

```
<int: No of kids(n)> <int: Source kid(i)> <int: Target kid(j)>
<int: x of kid 0> <int: y of kid 0> <int: p of kid 0>
<int: x of kid 1> <int: y of kid 1> <int: p of kid 1>
...
<int: x of kid n-1> <int: y of kid n-1> <int: p of kid n-1>
```

4 Report (30 Points)

Please prepare a report and discuss the following items:

1. Explain your code and your solution. (10 points)
   (a) Write your pseudo-code.
   (b) Show the time complexity of your algorithm on the pseudo-code.
2. Why should you maintain a list of discovered nodes? How does this affects the outcome of the algorithms? (5 points)
3. How does increasing the number of the kids affects the memory complexity of the algorithms? (15 points)
   (a) Show the space complexity of your algorithm on the pseudo-code.
   (b) Show run-time of all cases in a graph with number of nodes in x-axis and run-time in y-axis.

5 Grading Details

- You have 14 test scenarios totally, 8 of them public test scenarios and 6 of them hidden test scenarios. You can check your solutions using 8 public test case scenarios.
- Each test case has 5 points (0.7 points for graph implementation, 2.2 points for BFS, 2.1 points for DFS).
- You may need to use Calico to test your solutions by using **public_test_cases.t**.
- After deadline, the hidden test scenarios also will be shared with you. Therefore, you may estimate your approximate grades by using Calico.
- You required to write clean and readable code. A penalty of up to 10 points will be conducted in that case.

Important Notes:

- Please be aware of the deadline.
- Interactions among individuals are prohibited.
- It is prohibited to share or copy any code from your classmates or from the Internet.You have to submit your own, individual project.
- Please submit your homework through **only** Ninova.
- You must submit all your source code in a single cpp file that includes your source code and a softcopy report. You can define multiple classes in a single cpp file.
- Your source code has to be named as **main.cpp**.
- All your code must be written in C++, and should be compiled and run on ITU's Linux Server (you can access it through SSH) using g++. Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.
- When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary.
- In the report, you should show and explain how the algorithm works to find solutions of the given questions with meaningful explanations. Do not just tell the story please.
- You should be aware that the Ninova's clock may not be synchronized with your computer, watch, or cell phone. If you have submitted to Ninova once and want to make any changes, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail.
- Send an email to **sayinays@itu.edu.tr** or **cengiz16@itu.edu.tr** for your questions.