CENG218 – Analysis and Design of Algorithms
Homework 3

Exercise 1: Shortest Paths

The CSV file *probabilities.csv* includes the information for 11 players appearing in a match. In the file, a value for a player on the main diagonal is the probability of the player scoring a goal. Each of the values in a row other than the value on the main diagonal is the probability that a pass is successful from this player to the related player. For example, a value in row 1, column 2 is the probability of a pass being successful from player 1 to player 2. A zero value for a player on the main diagonal means that the player cannot shoot and there is no probability for scoring a goal. However, other zero values mean that there is no edge from a player to another player. Hence, in this exercise the ball is now at the given player, and the aim is to find the path that maximizes the probability of the given player to score a goal.

A path length is the sum of the values on the edges of the path. In order to minimize the product rather than the sum, we take logarithms. When the edge values are replaced by the logarithms, the shortest path is the smallest product (Because log(a*b)=loga + logb). However, we want to maximize the joint probability rather than to minimize it. Thus, we multiply the edge values by -1. With the preprocessing of the edge values and using a shortest path algorithm, we maximize the product rather than minimize the sum.

Two positional command-line arguments will be given to the program:

1. Path of the probabilities file (e.g. data/probabilities.csv)
2. A positive integer s for indicating the index of the player.

You can assume both path and s are valid (Such file exists and $0 \leq s \leq 10$).

This program is expected to do the followings:

1. Creates a directed graph.
2. Reads the probabilities.csv file line by line and adds the probabilities to the graph.
3. Displays the matrix given in the file.
4. Preprocesses the edge values except for the self-edges in the graph by taking the logarithm 10 of an edge value and multiplying it by -1.
5. Executes the Dijkstra's shortest paths algorithm with the given player s as the source vertex using the preprocessed values.
6. Since all the players in a match cannot shoot, to find the path maximizing the goal-scoring probability for the given player, the program performs postprocessing for the shooters. It multiplies the calculated value for a shooter by the probability of that shooter scoring a goal. If the given player is a shooter, the probability of scoring a goal has already been given and there is no need for multiplication.
7. Displays the path that maximizes the probability of the given player to score a goal.

Exercise 2: Greedy Algorithms

This exercise uses times rather than probabilities. The CSV file *times.csv* contains the information of the times for 11 players present in a match. The values of the main diagonal indicate the shooting times for the players. Each of the other values in a row is the time taken for the player to pass the ball to the respective player. As in Exercise 1, a zero value on the main diagonal indicates that the player cannot shoot, and other zero values demonstrate the ball is not passed and there is no edge from a player to another player. In this exercise, the ball is now at the given player and the player tries to score a goal as quickly as possible. Hence, find the path that gets the ball to the goal in the shortest time.

The idea is that if the player holding the ball can shoot, the player shoots. Otherwise, the player gives a pass to whomever the player can pass in the shortest time. But, the player does not choose the players who have already held the ball. If there are more than one player with the shortest time, the player chooses the one with the smaller index.

If there is no one to pass the ball, it means it is not possible to shoot anyway and the input is invalid. You can think this situation does not happen.

The program displays the matrix given in the file and the path that gets the ball to the goal in the shortest time.

Important Notes

- Your outputs must be exactly the same with the example run files and you can create your own files for testing.
- Assignments must be submitted by teams of up to two students. Once you form a team and start working on the homework, you cannot change your team. Otherwise, two different teams may end up with similar submissions, which will be considered cheating.
- One of the team members must submit a single file named **ceng218_hw3_studentno1_studentno2.zip**. If both of the members submit a file, then we will select a random one. If you do the homework individually, submit a file named ceng218_hw3_studentno.zip. The compressed file must only contain two files: **exercise1.cc** and **exercise2.cc**. These files must be independent from each other. If they reference each other, simply copy the common code to both files to make them independent.
- Much of the homework will be evaluated **automatically**! Thus, the functionality of your code (in terms of inputs and outputs) is strictly defined. There is only one correct output for each scenario (i.e. input set).
- Make sure you do not rely on absolute paths in your code! (For homework, always use relative paths.)