PROJECT REPORT
SE115 FALL 2024-2025

Name: Semih Öztürk

Student ID: 20220601089

Purpose of the Project:

It aims to simulate a system that computes the shortest path between cities over a given map using Dijkstra's algorithm.

Classes in the project and their functions:

Main Class:

This is the main part of the program. It receives the file path from the user, reads the file and communicates with other classes to calculate the shortest path. The result is both displayed on the terminal and written to a file.

Some Critical Points:

Scanner scanner = new Scanner(new File(filePath));

This line allows us to read the map from the file path given by the user.

System.out.println(result);

The user will be shown the result through the terminal. If we don't add this, the user will not see the result!

City Class

This class represents each city. It stores only the name of the city and allows other classes to access this information.

Critical Points:

private String name;

Holds a name for each city.

public String getName() {

return name;

}

This method returns the name of the city. Other classes access the city information with this method.

I did not have any difficulty because it was at Basic Level.

CountryMap Class:

Stores information about all cities and roads. It represents roads with an adjacency matrix. This matrix holds the distances between two cities.

Critical Points:

private int[][] adjacencyMatrix;

This matrix stores the lengths of roads between cities. If there is no road between two cities, the corresponding cell is assigned Integer.MAX_VALUE.

public int[][] getAdjacencyMatrix() {

return adjacencyMatrix;

}

This method returns information about paths so that other classes can access this information.

WayFinder Class

This is the most important class! It finds the shortest path between the start and end cities specified by the user. For this I used a method like Dijkstra's algorithm.

Critical Points:

int[] distances = new int[cityCount];

boolean[] visited = new boolean[cityCount];

The distances array stores the minimum distance to each city. the visited array keeps track of which cities have been visited.

while (currentIndex != -1) {

path = cities[currentIndex].getName() + " " + path;

currentIndex = previous[currentIndex];

}

This loop is the shortest path. So it determines which cities we pass through from start to finish.

Challenges I faced while doing the project and their solutions:

I really struggled at a few points while doing this project, but these difficulties were actually learning opportunities for me.

Problem

At the beginning, I had a hard time doing file input-output operations. Especially reading the data from the file correctly and converting it into the structure the program needs seemed complicated at first. In my first attempts, I kept getting FileNotFoundException errors because I was giving the wrong file path.

Solution

To solve this problem, I first tried to learn how to get the file path through the terminal. I also planned how to read the data line by line by using the Scanner class more carefully. Finally, I fixed the bugs by testing each step little by little.

Problem

It was a bit challenging to write a graph-based algorithm like Dijkstra's algorithm with arrays at SE115 level. In particular, it was complicated to link cities to indices and to build the distance matrix correctly. I kept getting incorrect results because of wrong indexes.

Solution

At this point, I first simulated the algorithm step by step on paper, trying to figure out what to do for each city. After I set up a system that matches city names with indexes in the code, things became much easier. Testing every change I made by printing it out made it easier for me to find bugs.

Problem

It was a bit confusing to show the result in the terminal on the one hand and print it to a file on the other. Especially in the first attempts, either nothing showed up in the terminal or the output to the file was missing.

Solution

To solve this problem, I tested the code step by step, first focusing only on the terminal output, and after everything worked, I added the write to file part. I also cleaned up any unnecessary or confusing code when writing to file, so that I had a simpler and more functional method.

Sources I Used While Doing the Project

While developing the project, I used many different sources, which both guided me on technical issues and helped me understand the project better.

Lecture Slides (SE115)

The basics I learned in SE115 (arrays, loops, classes and functions) formed the basis of all the structures I used in the project. I especially benefited from these notes when working with arrays and using loops effectively.

Internet Research

I researched online resources to understand how I could implement certain parts of the project. In particular, I used the following sites:

GeeksForGeeks

I used it to understand the logic of Dijkstra's algorithm and to see how to implement it step by step.

I also found useful information on building data structures with arrays.

Stack Overflow

I used it to find solutions to bugs I encountered and to make certain pieces of code more efficient.

I used the solutions here for getting file paths from the terminal and error-free file operations.

W3Schools Java Tutorials

I used it to quickly remember the basic functions and structures of Java.

Sample Projects and Code Fragments

Analyzing similar projects gave me an idea about which structures I could use. I analyzed how I could simplify the methods used in these projects in my own project.

My own trial and error process

It didn't always work to transfer everything I read from the sources directly into the code. Many times I encountered bugs and developed my own ways of trying to solve them.

Guidance from Teachers and Friends

The tips given by our lecturer and the discussions we had with friends helped me understand the project better. Listening to someone else's point of view helped me to find different solutions to problems, especially when I had difficulties.