

PROGRAMMING HOMEWORK 2 REPORT

**CMPE 343/224-SECTION:04**



May 9, 2023

Begüm Şara Ünal

# Problem Statement and Code Design

## Question 1

The provided code determines whether a given directed graph representing a ride network can be transformed into a tree structure. A series of taxi journeys connect beginning and finishing stations to make up the ride network. The program's inputs are the number of stations in the ride network, the total number of taxi rides, and a list of all taxi rides. The program's outputs include a list of stations and their related neighbors and a message stating if the current ride network can be transformed into a tree structure.

The code consists of a single class named "uberNetwork" that contains a static inner class named "uberCheck.” The "Q1" class serves as the entry point to the program and contains the "main" method that reads the input, creates an instance of the "uberCheck” class, and calls its methods to process the input and output the results.

The "uberCheck” class encapsulates the logic for processing the ride network. It contains three fields: "stops,” "destination,” and "stationOrder". The "stops" field represents the number of stations in the ride network, the "destination" field represents the graph that connects the stations, and the "stationOrder" field represents the order in which the stations were added to the graph.

The "uberCheck.” class contains four methods: "connection", "getDestination", "getStopOrder", and "checkTree". The "connection" method is called for each taxi ride and updates the "destination" and "stationOrder" fields accordingly. The "getDestination" and "getStopOrder" methods return the "destination" and "stationOrder" fields, respectively. The "checkTree" method checks whether the ride network can be converted into a tree structure by performing a breadth-first search on the graph and checking if all the nodes are visited and there are no cycles.

## Question 2

This assignment involves calculating the minimum cost of navigating between farmlands in a town using the MST algorithm.

To accomplish this task, we need to read the number of test cases from the user, ask for the number of rows and columns in the grid for each test case, and read the starting and ending cell coordinates for the given farmlands. Additionally, we need to read the values of the farmlands in the grid, calculate the cost of an edge connecting any two cells with values a and b using the formula E(e)=a⊕b, and finally use the MST algorithm to find the minimum cost of a good trip between two cells.

# Implementation and Functionality

## Question 1

uberNetwork class: reads the input using the "Scanner" class and creates an instance of the "uberChecking" class with the number of stations as its argument. It then reads the list of taxi rides, calls the "connection" method for each ride to update the "destination" and "stationOrder" fields, and calls the "getDestination" and "getStationOrder" methods to obtain the graph and the order of the stations, respectively. It then processes the graph and outputs the list of stations and their corresponding neighbors using a for-each loop that iterates over the "stationOrder" list and obtains the neighbors of each station from the "destination" field. It also calls the "checkTree" method to check if the ride network can be converted into a tree structure and outputs the corresponding message.

## Question 2

The program is composed of different sub-modules, each with its task.

readInput(): Takes no input and returns the number of test cases, a list of tuples containing the grid size and starting and ending cell coordinates for each test case, and a list of 2D arrays containing the values of the farmlands in the grid.

calculateEdgeCost(): Calculates the cost of an edge e connecting any two cells with values a and b using the formula E(e)=a⊕b. It takes two integers (a and b) as input parameters and returns the cost of the edge.

createGraph(): The module creates a graph of the given grid using the values of the farmlands as vertices and the cost of the edges between them as weights. It takes a 2D array of integers (the values of the farmlands in the grid) as an input parameter and returns a dictionary representing the graph.

calculateMSTCost(): The module uses the MST (Minimum Spanning Tree) algorithm to find the minimum cost of a good trip between two cells. It takes a dictionary representing the graph, the starting and ending cell coordinates, and the grid size as input parameters. It returns the minimum cost of a good trip.

main(): The module is responsible for the overall flow of the program. It calls the readInput() module to read the input from the user and then uses the other sub-modules to calculate the minimum cost of the excellent trip for each test case.

# Testing

Within this segment, I present a tester class that pinpoints the vital test points within our program. This class generates further test data beyond the given sample input/output, which helps to clarify precisely what aspects of the solution are being tested with each set. Additionally, I've included descriptions of any program bugs discovered during testing. These explanations illustrate our testing process, summarize our findings, and assert that we've covered all potential program behaviors.

# Final Assessments

* What were the trouble points in completing this assignment?
  + The trouble point in the first question is the output parts. The output should be printed in order; however, I changed my code many times to solve this issue.
  + While working on this task, my most significant difficulties were comprehending the problem statement and figuring out how to utilize the MST algorithm to determine the most cost-effective route for a successful journey.
* Which parts were the most challenging for you?
  + The most demanding aspect of the project was developing the createGraph() module, which involved accurately constructing a graph of the provided grid utilizing the farmland values as vertices and the edge costs as weights.
* What did you like about the assignment? What did you learn from it?
  + I like to reinforce the topics. I learned how to use a minimum spanning tree with this assignment.