Information:

Giray Berk Kuşhan Section 04

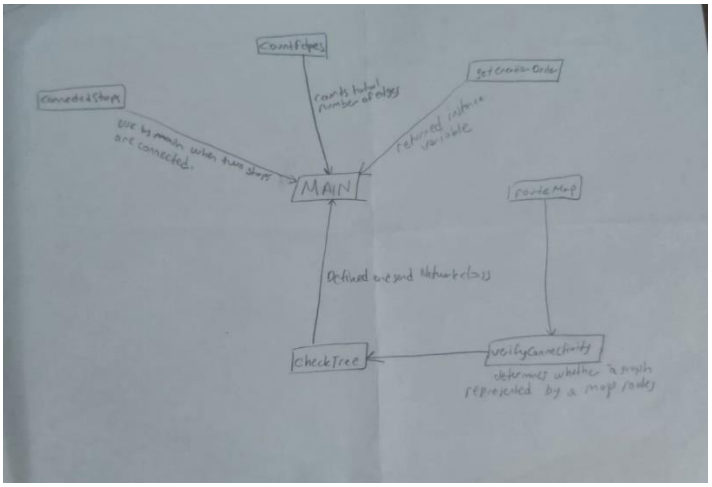
10889878942

Homework 2

Q1:

Problem Statement and Code Design:

In question 1 there is a directed graph, and it represents a ride station in Ankara. Also, it can be showed as a tree structure. There are edges, cycles, and connections. In code part it occurred with Network class, and it includes some of the methods. It checks graph can be showed as a tree or not. Also, it makes control same thing for network. Methods of this questions are that connectStops, getRouteMap, getCreationOrder, checkTree, verifyConnectivity, countEdges.



This is the chart of the question 1.

Implementation and Functionality:

void connectStops (String start, String end): if routeMap has no starting station, add starting point. Then, if routeMap does not have end station, insert the end stop.

<String> getRouteMap (): it returns instance variable in RouteMap.

<String> getCreationOrder (): it returns instance variable in CreationOrder.

checkTree (): it makes control and return false if number of edges is -1. Then, it occurs set to keeping visited stops. After That it get first stop from the routeMap and calls verifyConnectivity. Finally, if all stations could be connected to first station. That means connected. It returns true.

boolean verifyConnectivity (String current, Set<String> visited): Aim of this code is firstly, adding current stop into visited set. Then, gets current stop's neighbors which are in the routeMap. If there are neighbors, iterate. But if there is no neighbor which is not visited, it uses recursive and calls verifyConnectivity method. When all neighbors are visited it returns false.

int countEdges (): start with 0. Then iterate along with neighbor's list in routeMap. Adding size into the edgeCount and return edgeCount as an integer.

Main () gets input as number of stops and rides. Then occurs network object and calls other methods such as connectStops, CreationOrder, getRouteMap step by step.

Testing:

In this part I created 3 test parts for minimum input values, tree structure, non-tree structure separately. This code passed the route map, creation order, and tree check result as I expected in every test. In addition, these test cases occurred in "Tester" class.

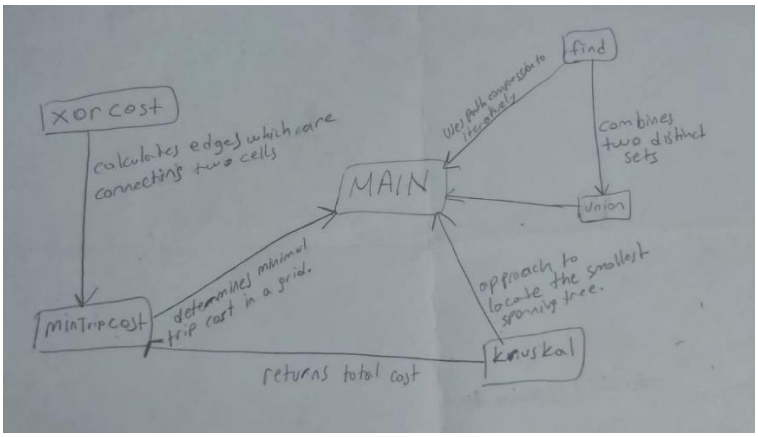
Final Assessments:

Completing this part is not too hard. But firstly, I misunderstand the question and I tried to provide outputs in alphabetically. But third test case is evidence it is not interested. So, it might be hardest part. There was a no specific challenging part. But sorting stations was a little bit hard. I liked this assignment and I have learned couple of things from Data Structure and Algorithms lecture such as representing directed graph and implementing recursive graph traversal algorithms when I researched the assignment.

Q2:

Problem Statement and Code Design:

The question 2 includes find and calculate min cos for navigating between farmlands I grid structure. Also, it aims get used XOR operations to calculate cost for edges which are between the cells and find min cost for every cell. The code is used Minimum Spanning Tree algorithm an applies Kruskal's algorithm to find min cost route. Also, code occurs with Main, minTripCost, xorCost, find, union, Kruskal methods.



This is the chart of the question 2.

Implementation and Functionality:

Main (): It counts test cases and reads number of them. Then it makes same operations for grid size. After that detect values of grid and calculate min cost by calling minTripCost method. Then print result.

int minTripCost (int [] [] grd, int N, int M, int r1, int c1, int r2, int c2): Starts the "prnt" as a parent array. Then occurs an empty list for edges. After that iterate to grid for horizontal and vertical edges. Then, it sets the destination and sources vertices and return results of kruskal method for vertices and edges.

int xorCost (int a, int b): it is used to calculating edge's cost which are connecting two cells by performing XOR. Then return the result as an integer.

int find (int u): It uses path to iterative and make an optimization to their operations.

void union (int u, int v): It combines two disjoint. They have a relationship as one of the is parent of other.

int kruskal (List<Edge> edges, int vertices, int src, int dest): This method initialize cost with 0. Then sort edges according to ascending order. After that it gets current edge and checks whether sources and destination vertices are connected or not. Then break the loop. After that it adds weight to the edges which is current and increase count. Finally, it returns cost as an integer value.

Testing:

In testing part, I created several tests, with a different test case start and end positions, grid sizes and grid configurations are tested. This testing checks control min cost of trip is true or not and verified program's reliability. These test cases occurred in "Tester" class.

Final Assessments:

Completing this part is a little bit hard. Especially third test case was too hard. Also, other problem was understanding the question clearly. So, it might be hardest part is understanding question and solving. But third part might be the most challenging part. I liked this assignment and when I researched the assignment, I learned couple of things from data structure and algorithms lecture such as Kruskal's algorithm, minimum spanning tree, representation of graph using edges.