CPSC 2150 – Algorithms and Data Structures II

Lab10: Graphs

Total - 50 Marks

Learning Outcomes

- Design and implement an appropriate data structure for a given graph.
- Design and implement the efficient graph-based algorithm in terms of time and space complexity.
- Develop C++ program based on the existing constraints.

Resources

- Chapter 13 of the text book

Description

All algorithms discussed in the lecture for determining the minimum spanning tree have one thing in common: they start building the tree from the beginning and they add new edges to the structure, which eventually becomes such a tree. However, we can go in the opposite direction and build this tree by successively removing edges to break cycles in the graph until no cycle is left.

In this way, the graph turns into the tree. The edges chosen for removal should be the edges of maximum weight among those that can break any cycle in the tree. This algorithm somehow resembles the Kruskal method, but because it works in the opposite direction, we call it **reverseKruskal**.

a. **[35 marks]** Use the above algorithm, and write a function named **reverseKruskal** to find the MST for an undirected graph of distances between different cities (more than 20 cities) which is provided as an input file. The first line of input file represents the number of nodes in the graph. For the rest of lines, each line represents an edge of the graph and includes 3 values; city1, city2 and distance each separated by a space. Your program should print out all the removed edges while it is running. Finally, when the minimum spanning tree has been found, the program should print the edges in the tree and the total distance. (**MST.cpp**)

To test your algorithm with different data you can use your lab 10 exercise to generate the input file, but make sure that your input file represents a connected graph.

b. **[5 marks]** Find the time and space complexity of reverseKruskal function in terms of number of nodes and/or number of edges in the graph (**answers.pdf**).

c. **[10 marks]** Compare this algorithm with Kruskal and Prim algorithms. Be sure to discuss both time and space complexity and data structure in your comparison (**answers.pdf**).

Submit to D2L

*Make a **zip file** named* **StudentNumber-*lab10.zip*** *including all related files* by the end of the lab time**.** For example, if your student number is 10023449, the submitted file must be named as **10023449-lab10.zip**.