AMAR BESSEDIK

CSC560 – ANALYSIS OF ALGORITHMS.

KRUSKAL'S PROJECT

DESCRIPTION

This project solves the Minimum Spanning Tree (MST) problem using Kruskal's Algorithm with path compression. To achieve our goal, data for weighted graphs with (n) vertices and (m) edges are given in text files. An MST, if it exists, is a subgraph containing (n – 1) edges such that when their weights are added together, they give the minimum possible cost needed to visit all vertices exactly once without creating any cycles. An MST is not possible to compute if the graph is not connected as the algorithm runs out of edges to consider before satisfying the above condition.

The implementation of Kruskal's algorithm uses a disjoint-set data structure implemented using an array of integers called "sets". In this data structure at start, the cell at index "i" is the single representative (or label) of the element "i". Thus, we make (n) disjoint subsets where one of them contains one single vertex. As the subsets merge together, if the index "i" is not the representative of its subset, we go through the subsets until we find its representative. Once this operation is achieved, we proceed to path. The compression of the path that led to our target can be achieved by introducing a caching mechanism into the find operation where all elements in the path to the representative will, after that, point to the target label. This way it speeds up the subsequent find operations and speeds up the overall running time of Kruskal's algorithm. At the end, if an MST is computed, all edges that make it will be displayed along with their total weight.

Space analysis:

- O(n) space is needed to construct the disjoint data structure.
- O(n) space is needed to construct the graph.
- O(n) space is needed to construct the MST.

Time analysis:

- Heapsort(E): O(m Log m)
- Merge (label1, label2): O(m Log_2 n).
- Test for cycles, O(1).
- Find(x), O(n) + O(m Log_2 n) = O(m Log_2 n).

Run Summary (included)

Bug Report:

The programs work properly and there are no know bugs.

Technical Information

- Make a **Graph object** by reading the corresponding text file of a graph which is in the local directory where execution takes place.
- Make a **Kruskal instance** using the graph object previously created.
- Kruskal instantiate an **MST object**.
- Extract the graph's **parameters**: # of vertices & edges.
- Pass the above parameters to the **Kruskal function**.
- The Kruskal function **outputs execution results** depending on whether there is an MST or not.

Appendix1 (source code included)

Appendix 2 (Run & drawings included)