Deuce Palmer
CS 3353-001
3/27/2023

Program 2: Description

Algorithm to create a Maximum Spanning Tree

- I used Kruskal's algorithm and utilized disjoint sets to check for cycles. I utilized the fastest disjoint sets implementation, which corresponds to versions 3 & 4 in the slides. Referring to the slides, the contents of makeset2() are created in my createMaxST() function. The functionality of my findSet() corresponds to findset4() in the slides. Lastly, unionSet() combines the functionality of megretrees3() and union3(). When creating the Maximum Spanning Tree, the code is exited once there are n-1 edges in the MaxST. This increases efficiency as every other subsequent loop call would be redundant as a tree can only have n-1 edges.

Algorithm to find the path between two nodes

- I used a DFS to find the path requested by the user. I did this by beginning at the start node and then saving a snapshot of the stack when the DFS reached the destination. Once the destination is found, I stop the DFS in order to save time and increase efficiency.

How the graph is being represented

- I stored the contents of the graph in a list of edges in order to make Kruskal's algorithm more efficient. With the list of edges, all I had to do was sort the list in descending order of capacity and pull the two nodes stored in the edge pair. As I initially used a list of edges, I also stored the Maximum Spanning Tree as an adjacency list. This proved to be beneficial during the DFS when looking for adjacent nodes, as I would cycle through the list of <node, edge> pairs for the current node being visited. In addition, a copy of the Maximum Spanning Tree was also stored as a list of edges. I chose to do this because it made the print of HW2Prog much easier as the list of edges has 1 entry per edge as there are 2 entries per edge in the adjacency list.

HW2Prog breakdown

- First, I created the Maximum Spanning Tree by calling createMaxST(). Then set all nodes to unvisited and begin the DFS with the starting node in the path. Once the DFS is completed, the capacity for the correctPath will be initialized to the capacity of the first edge. After every edge in the correctPath is examined, the path capacity will equal the capacity of the smallest edge. The contents of the Maximum Spanning Tree will then be printed to standard output (this is where the list of edges comes in handy). Lastly, the function will return the correctPath from s -> t.