Joshua Steward

Lowell Batacan

Kruskal's Algorithm

Kruskal's algorithm is a classic example of a greedy algorithm in which the at every step, the current lowest cost is chosen as the next element. It's also an example of one of these greedy algorithms in which the obvious solution works perfectly. As long as the next lowest cost edge at every stage does not form a cycle, it grows into the Minimum Cost Spanning Tree.

Our algorithm works by first sorting the edge list. By defining a default equality operation function in the edge class that we stored our edges in, this was done efficiently and in a "pythonic" manner. It reads in the file and the edges, storing the edges in a list of this object. Then, in the actual minimum cost spanning tree algorithm, it creates a separate list for the kruskal algorithm edges. Looping through all the edges, we check if the next edge creates a cycle. This can be done because the list is already sorted. If it doesn't create a cycle, it is appended to the end of the minimum cost spanning tree's list of edges. If it does create a cycle, it doesn't do anything. To detect a cycle, we explored several different approaches. The approach that worked most effectively, especially when using an adjacency matrix, would be to loop through every column and row searching for edges, and storing the original edge to be checked against. If next edge is equal to this original edge, it is a cycle. If not, we check for the next edge, and if found, the cycle checking function is called recursively with the column then as the row. A flagged variable detects if it is cyclic.

Kruskal's algorithm can be proven for correctness using induction; i.e., that it finds a minimum cost spanning tree. Our tree is progressively added to, and so it is important to show that we are along the right track as long as the tree is not cyclic at any stage in the algorithm. The second part would need to prove that at the termination, the tree gives a minimum cost spanning tree. In the base case, without anything in the tree, this is already proven to have a minimum cost spanning tree. In the induction step, just before adding a new edge, if it is not cyclic, then we can divide the edges in the tree into connected components. These components either have both ends in the entire edge set or one edge, and the current edge is the least cost edge because the list is already sorted. From this, it can be concluded that with the new edge added in to the tree, it is a promising union. Because of this, when the algorithm reaches termination, it is a minimum cost spanning tree.

An n-grid may only have number of edges ((n^2)-n)*2. This can be proven because there would be n^2 nodes. There are horizontal edges, which must have (n - 1) * n in order to complete the graph, and vertical edges, which must have (n - 1) * n to fulfill this. In the case of n = 4, there can be at most 24 edges.