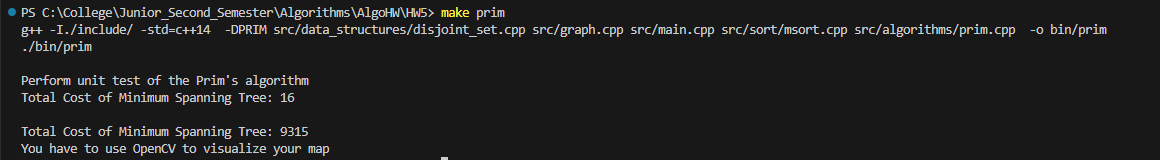
Blake Williams

Algorithms HW5

Student ID:010974718

5/4/2024

This project focuses on implementing 3 different Algorithms on the same project that has been worked on all semester. The first algorithm to be implemented was the Prims algorithm. The goal of the algorithm was to construct a minimum spanning tree. To do this first a variable called mst was declared which stores the edges that are a part of the constructed MST. Next was a variable called pq which stands for priority queue which stores the smallest weight edge given the highest priority, it uses EdgeKeyComparion to achieve this. Next is the visited variable which is a Boolean vector which keeps track of if vertices are visited or not. Finally the parent variable which stores the parent of each vertex into the MST tree. The way the algorithm works in the implemented code is it begins at vertex 0, marks the start as visited, then adds all connected edges to the priority queue. Once the priority queue is created a while loop is created which runs until the priority queue is empty. Inside the while loop first it extracts the top edge since it has the smallest weight, next it checks to see if the edge is already visited, if it is then it is skipped. If not visited, then it is marked as visited, adds the current edge to the mst, updates the parent of ‘v’, and pushes all edges from vertex ‘v’ to the queue, as long as they have not been visited yet. The cycle completes until the queue is empty then it returns the mst containing the edges. Next to be implemented was the Kruskal algorithm. To do this 2 variables are created, the first called edges which is a vector that stores all the edges from the Graph via the .exportEdges() function. The next variable is another vector that stores all the edges that are in the minimum spanning tree. The first step is to Sort the edges in the ‘edges’ function by lowest to highest order. Next a Disjoin set is declared where each vertex starts in its own set, this is needed for detecting cycles. Finally the algorithm runs through each of the sorted edge and for each individual edge checks if the two vertices of the edge belong to the same set. If they do then adding an edge would create a cycle, thus the edge is skipped. If they don’t, and are of different sets, then the edge does not form a cycle and is added to the MST. Finally the Dijkstra’s algorithm was implemented. First 4 variables were created being path, pq(priority queue), dist(shortest known distance), parent(predecessor of each vertex). The algorithm first initializes the priority queue, and while the priority queue is not empty it processes vertex u from the edge with the lowest weight. If the cumulative weight is greater than the shortest distance dist[u] to vertex u, then this is an outdated path and the loop continue to the next iteration. For each of the ajacent vertexes ‘v’ and ‘u’ if the algorithm finds if a shorter path to ‘v’ is found by going through ‘u’. If true then it updates dist[v], and records ‘u’ as the parent. Once desetnation is reached the algorithm traces back to the start by following parents. This path is reverse of what we want so the reverse() function is used to invert the path to what we need.

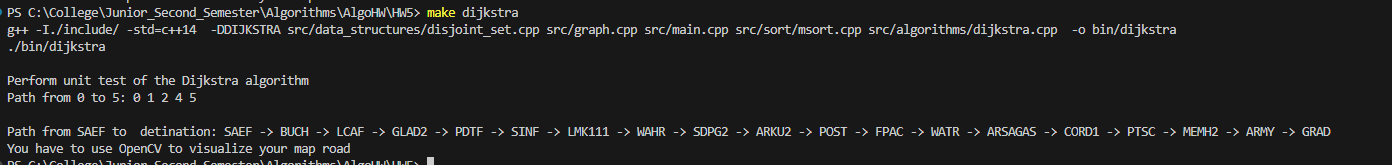


Above: Prim Algo

A black screen with white text

Description automatically generated

Above: Kruskal



Above:Dijkstra

View williams\_typescript.txt for the full compilation examples