Paul-Ann Francis – U61445542
Ahmed Ghoneim
Guillermo Medina
Ameena Mohammed - U39296299

COP 4530 Final Project Report

Description of Algorithm

Dijkstra's Algorithm is based on a pure greedy approach in order to find the minimal number of steps to reach a destination. The algorithm does so by looking at the weight or cost on each edge present at a vertex then analyzes the shortest path to the next vertex until it reaches the destination. There are two main defined properties for each vertex in this algorithm.

Path property which stores the value of the current minimum path to the vertex and continuously updates it until the shortest path is discovered and stored.

Visited property which finds out whether a vertex has been visited or not and only marks the vertex visited if the shortest path has been discovered. This property is mainly being used so we don't revisit a vertex again.

These properties are very important to understand as initially all the vertices are marked unvisited and the path to all vertices is set to infinity excluding the source vertex. After the source vertex is marked visited. The Algorithm starts to access all the neighbours of the source vertex and perform a process called "relaxation" on each vertex. Relaxation is a process that lowers the cost of reaching a vertex by using another vertex. In every step, an unvisited vertex with the least path must be marked visited and its neighbour's path to be updated. Finally, this process is repeated until all the vertices in the graph are marked visited.

Type of Data Structure used For Graph

Graphs consists of nodes called a vertex and their edges that connect these vertices to other vertices. The edge is given a weight which is simply a number that is used as the distance between two vertices. The graph uses multiple underlying data structures including that of a double-ended queue and a vector; all directly pulled and used from the Standard Library Template. The Vertex, Edge and Graph are all separated into classes where the first two are then used and implemented inside the latter Graph class. The Edge Class contained the variables for the current and adjacent edge as well as the variable for the weight assigned to each of these edges. The Vertex class then inherits the Edge class and adds its own variables and methods. The most important ones being a variable for the string data, one to keep track of the minimum distance, and a vector of type string to store the shortest paths between edges.

The graph class then gets access to these classes through the friend function where it stores the edges as a deque of edges and the vertices as a deque of vertices. It also use a Priority queue to store weighted vertices by distance from the starting node from smallest to largest.

Paul-Ann Francis – U61445542
Ahmed Ghoneim
Guillermo Medina
Ameena Mohammed - U39296299

Time Complexity and Space Efficiency

For Dijkstra's Algorithm and its representation of graphs, it can be structured though a matrix representation or an adjacency list representation. For our purposes, it is better to use an adjacency list as it directly improves our time complexity and space efficiency for our Algorithm. While matrix representations take an O(V^2) run time, Adjacency list takes O(E Log V) and can further be reduced into O((V+E) log V) by using a Fibonacci Heap which that are being used in our Algorithm. Here we assume that E is the number of edges in the graph and V is the number of vertices in the graph. Within the adjacency list, all the vertices can be traversed in O(V+E) time by using Breadth First Search which uses a queue data structure.

Functionality

Dijkstra's Algorithm is mainly used to solve single source shortest path problems that doesn't have a negative edge weight in the graphs. This Algorithm is used to find the minimum distance between two locations along with its path. This is currently being used in the Digital Mapping services in Google Maps. Furthermore, if we Imagine a city to be a graph, the vertices to represent a switching station and the edges to be a transmission line it can be used to find the bandwidth of each transmission line and can be used in Telephone Networking.

Dijkstra's Algorithm can also be used to find the shortest open path in IP routing by using Open Shortest Path First (OSPF). OSPF is a routing protocol that is used to find the optimal path between the source and destination router. The algorithm also finds the shortest cost path of a router to the other routers connected to a specific network.

Dijkstra's Algorithm is also applied when creating social media applications. To suggest familiar people that a user may know, these applications utilize Dijkstra's algorithms to find the shortest path between users measured by the number of connections and interactions made between users. When the social networks are more closely knit, the algorithm along with other features are used to find potential close connections among users. This is why people will often get recommendations to follow or send a friend request to individuals they may know in real life. The use of graphs shows the nearest connections of the people you are already connected to since the chance you may know them is high as well.