Ivy Truong

5/1/20

CSCI 3412

Homework 6

Question 2)

a) To represent $269.63 in the least number of bills and the least number of coins, we would first determine the biggest bill to use to represent the amount, then going down the list from there. If an amount has a digit that is in the hundreds place or higher, then we would start using the $100. If there is a digit in the tens place and the digit is greater than or equal to 5, then start using the $50. If the digit in the tens place is less than 5 but greater than 1, then use the $20. If the digit in the tens place is a 1, use the $10. If the ones place digit is greater than or equal to 5, then use the $5. If the ones place digit is greater than or equal to 1, then use the $1. If the tenths digit is greater than or equal to 5, use the 50 cents. If the tenths and hundredths place in the amount is greater than or equal to .25, then use the 25 cents. If the tenths place is greater than or equal to 1, then use the 10 cents. If the hundredths place is greater than or equal to 5, then use the 5 cents. If the hundredths place is less than 5, use the 1 cent.

When a bill or coin is obtained, it is subtracted from the original amount, so we know how much is left. To get $269.63, the process is highlighted in steps below:

* Get $100 bill, leftover amount is $169.63
* Get $100 bill, leftover amount is $69.63
* Get $50 bill, leftover amount is $19.63
* Get $10 bill, leftover amount is $9.63
* Get $5 bill, leftover amount is $4.63
* Get $1 bill, leftover amount is $3.63
* Get $1 bill, leftover amount is $2.63
* Get $1 bill, leftover amount is $1.63
* Get $1 bill, leftover amount is $0.63
* Get 50 cents, leftover amount is $0.13
* Get 10 cents, leftover amount is $0.03
* Get 1 cent, leftover amount is $0.02
* Get 1 cent, leftover amount is $0.01
* Get 1 cent, leftover amount is $0.00
* Total bills and coins: 2 $100 bills, 1 $50 bill, 1 $10 bill, 1 $5 bill, 4 $1 bills, 1 quarter, 1 dime, 3 pennies

b) A change in the conditions that make this greedy algorithm fail is if the order of how the bills and coins were presented were not in descending order. For example, if the order of the bills and coins was $50, $20, $0.01, $100, $0.25, and someone wanted to find the least number of bills to make $100, with the greedy algorithm, it would only see the $50 as the most optimal choice at the time, so the amount of $100 is represented as 4 $50 bills rather than $100.

Question 3)

a) Dijkstra’s algorithm is an algorithm that solves for the shortest path of an undirected or directed graph with no negative edges. The main goal is to solve for the shortest path to all connecting nodes on the graph without any cycles. In this algorithm, each edge between two nodes are determined by a process called relaxation. In relaxation, the algorithm will determine if adding this new edge will have the lowest cost between the start node and the node to visit through this edge. For example, consider a path that goes from 1🡪3 with a total distance of 7 and a path from 1🡪2 with a total distance of 3. If the next shortest edge that is connected to the next visited node is 2🡪3 and its cost is 2, the process of relaxation will check if the distance from 1🡪2🡪3 is shorter than the current shortest path to node 3 from node 1 (which is 7). The relaxation process will see that the total distance from node 1 to node 3 using the path 1🡪2🡪3 is shorter than the current shortest path 1🡪3, so the algorithm will relax the edge 2🡪3, which will include this edge as part of the new shortest path from node 1 to node 3. The next hop starting from the start node in order to reach the neighboring node is updated. For example, since 1🡪2🡪3 is the shortest path, the path to 3 has a cost of 2 with the next hop being node 2. The process of the algorithm goes as follows:

First, visit an unvisited node in the graph. Second, determine which neighboring node has the shortest path to the visited node. When the edge is determined, use the process of relaxation to see if the chosen edge should be part of the shortest path or not. If a path to the neighboring node is not in the shortest path (meaning has not been visited yet), relax the edge and add it to the shortest path. If a path to the neighboring node already exists in the shortest path, compare the new path to the current shortest path and replace the current shortest path if the new path is shorter. Before visiting another node, the algorithm will continue relaxing neighboring nodes until all edges to neighbors of the current visited node have been relaxed. After all the neighbors of the current visited node have been relaxed, the algorithm will visit the next unvisited node in the queue (which is based on the order of how the edges were relaxed for the last visited node(s)) and repeat the process of relaxation for the edges connecting its neighbors. This process will repeat until all nodes have been visited.

One of the main disadvantages of this algorithm is that it does not support edges with a negative weight. This is because Dijkstra’s algorithm only considers what is the shortest path, so it does consider if a path is increasingly negative. For example, if the shortest path from node 1 to node 3 is 5 and another path going from 1🡪2🡪3 is -40, Dijkstra’s algorithm will only see that -40 is less than 5 because in reality negative values are smaller than positive values. Dijkstra’s algorithm does not consider the magnitudes, so if the negative and positive signs could be removed, we could see that taking a path of 40 is more costly than taking a path of 5.

b) Prim’s algorithm is an algorithm that solves for the minimal spanning tree given an undirected graph with possibly negative edges. A minimal spanning tree is a subgraph of the original graph that connects all the nodes together with no cycles. The total number of edges that are present in an MST is the total number of vertices minus 1 (|V| - 1 = E). If the number of edges is greater than or equal to the number of vertices in the graph, it would introduce cycles and the subgraph would no longer be a minimal spanning tree. In this algorithm, the edges are decided to be part of the MST by removing the heaviest edge/edge with the highest magnitude that makes a cycle in the graph. To follow this property, the algorithm would only choose the smallest edge(s) from the graph and include it into the MST. The process of the algorithm goes as follows:

At the beginning of the algorithm, a node on the graph is chosen randomly as the first node in the minimal spanning tree. To determine which neighboring node to visit next, the algorithm will compare all the edges that are connected to the current visited node. The algorithm will then find the smallest edge out of the list and add the neighboring node as a visited node.

After calculating the shortest edge for the first node, it gets a little bit more complicated. To find the next smallest edge, the algorithm will compare all the edges that connect to the visited nodes. For example, if nodes 1 and 2 are in the MST, the algorithm will check all the edges that connect with node 1 or node 2 and find the smallest out of this list. If the smallest edge to a neighboring node has been visited, the algorithm will move on to the next smallest edge. This is done so that there are no cycles being formed in the MST. If the neighboring node has not been visited, then the edge will be added to the MST and the neighboring node becomes a visited node. This process will repeat until all nodes have been visited, meaning there is an edge that connects to each node to the graph with no cycles.

The main advantage of this algorithm in comparison to Dijkstra’s algorithm is that it can handle negative edges. This is because Prim’s algorithm only compares the edge weights to other neighboring edges to find the smallest edge weight; it calculates the subset path of the whole path. In Dijkstra’s algorithm, it calculates the whole path constantly from the start node to all other nodes, which can result in a large negative magnitude as the shortest path and is not reasonable.

The main difference between Prim’s algorithm and Dijkstra’s algorithm is that Dijkstra’s algorithm determines the shortest path in respect to the start node/first node visited in the graph. In Prim’s algorithm, it determines a minimal spanning tree by finding the shortest path between neighboring nodes. Prim’s algorithm doesn’t necessarily have the shortest path from a single node, but it does determine an optimal path to neighboring nodes.