Assignment 5 - Hady Ibrahim (400377576)
SFWRENG 2CO3: Data Structures and Algorithms–Winter 2023

Deadline: March 31, 2023

Department of Computing and Software
McMaster University

Problem 1. Consider a remote community of N houses $h_1, \dots h_N$. We want to provide each house with internet access at minimal cost.

We can make a local network between these houses by connecting them via long-range Wi-Fi using directional antennas. The cost to connect two houses h_i, h_j is given by $C(h_i, h_j)$ and will depend on their distance and the terrain in between (e.g., long distances and hills require strong radios and repeaters).

In addition, we can connect one or more houses directly to the internet via a new fiber connection. For house h_i, the cost of this fiber connection is $F(h_i)$. If a house has a direct internet connection, then it can share this connection with all other houses that are reachable via a path of long-range Wi-Fi connections.

The community wants to find how to connect all members of this community with internet at a minimal cost.

P1.1. Model the above problem as a graph problem: what are the nodes and edges in your graph, do the edges have weights, and what problem are you trying to answer on your graph?

The nodes in the graph represent the houses in the remote community, denoted by $h_1, \dots h_N$. The edges in the graph represent the potential connections between the houses via long-range Wi-Fi using directional antennas. These edges will be undirected as the $C(h_i, h_j)$ relationship is symmetric (the cost to wire goes both ways, so if I'm wired to you, you're wired to me). The cost to connect two houses h_i, h_j is given by $C(h_i, h_j)$, which represents the weight of the edge between nodes h_i and h_j in the graph.

We will also add a node F which denotes the fiber source. This node will have an edge to every house h_i with a weight of $F(h_i)$. Note: Not all houses have the ability to connect to fiber or to another house. If it doesn't exist the edge wouldn't exist in the graph.

The problem we are trying to answer on this graph is to find a minimum-cost spanning tree that connects all the houses in the community to the internet. The minimum-cost spanning tree will ensure that all the houses are connected to the internet while minimizing the cost of the connections between them and also assuring there is at least 1 connection to the fiber source.

P1.2. Provide an efficient algorithm to find a way to connect all members of this community with internet at minimal cost. Explain why your algorithm is correct, what the complexity of your algorithm is, and which graph representation you use.

The graph passed into the algorithm is the graph described in 1.1.

```
Algorithm ConnectCommunity($\mathcal{G} = (\mathcal{N}, \mathcal{E}), weight$):
1:  $E, M := \emptyset$, r where r is a node from M
2:  $Q$ :=min priority queue initialized with neighbours n of r alongside their edges $(n, (r, n) \in \mathcal{E})$
3:  while $M \neq N$ do
4:      Pop min weighted $(n, (m, n))$ from $Q$ $m \in M$ and $n \notin M$
5:      $E, M := E \cup (m, n), M \cup n$
6:      for $(n, w) \in \mathcal{E}$ do
7:          if $w \in Q$ then
8:              if $weight((n, w)) < weight(Q(w)$ edge) replace $Q(w)$ edge with (n,w)
9:          else
10:             add $(n, w)$ to $Q$
11:         end if
12:     end for
13: end while
14: return $E$
```

The graph representation is adjacency list representation. As seen in class this is just Prim's algorithm which is $|\mathcal{E}|log(|\mathcal{N}|)$ as proven in class ($log(|\mathcal{N}|)$ for getting min edge from min heap and you do this for each edge)

Problem 2. A company has several distribution centers. To simplify logistics, the company wants to figure out which distribution center is the "most central": the maximum time it takes to transport freight from this distribution center to any other distribution center is *minimal*. Assume we know, for every pair of distribution centers X and Y, the time it takes to transport freight from X to Y.

P2.1. Model the above problem as a graph problem: what are the nodes and edges in your graph, do the edges have weights, and what problem are you trying to answer on your graph?

To model the above problem as a graph problem, we can consider each distribution center as a node in the graph, and the time it takes to transport freight from one distribution center to another as the weight of the edge between the corresponding nodes.

Nodes: Each distribution center is represented as a node.

Edges: The edges in the graph represent the time it takes to transport freight from one distribution center to another in a straight line. The weights of the edges are the actual times. These are undirected since, I should be able to traverse the same path both ways in the same time.

Problem: The problem we are trying to answer on the graph is to find the "most central" distribution center, which is the distribution center with the minimal maximum time it takes to transport freight to any other distribution center. Therefore this is a shortest path algorithm problem for every single node, then finding the node with the minimum of its maximum shortest path.

P2.2. Provide an efficient algorithm to find the most central distribution center. Explain why your algorithm is correct, what the complexity of your algorithm is, and which graph representation you use.

```
Algorithm CentralDistribution($\mathcal{G} = (\mathcal{N}, \mathcal{E}), weight$):
1:  $central_n, smallest_{max} := null, \infty$
```

```
for $n \in \mathcal{N}$ do
    $path_{max}$ :=DIJKSTRAS($\mathcal{G}$, $weight$, $n$)
    if $path_{max} < smallest_{max}$ then
        $central_n$ :=$n$
        $smallest_{max}$ :=$path_{max}$
    end if
end for
```

```
Algorithm DIJKSTRAS($\mathcal{G} = (\mathcal{N}, \mathcal{E}), weight, s \in \mathcal{N}$):
    $path, cost := [n \rightarrow ? | n \in \mathcal{N}], [n \rightarrow \infty | n \in \mathcal{N}]$
    $path[s], cost[s] := s, 0$
    $path_{max} := 0$
    $Q$ := a minimum-priority queue that holds node $s$ with priority 0.
    while $Q \neq \emptyset$ do
        Pop node $m$ with lowest priority with from Q
        for all edges $(m, n) \in \mathcal{E}$ do
            if $cost[m] + weight((m, n)) < cost[n]$ then
                $path[n], cost[n] := m, weight((m, n)) + cost[m]$
                if $cost[n] > path_{max}$ then
                    $path_{max}$ :=$cost[n]$
                end if
                Update $n$ in $Q$ such that $n$ has priority $cost[n]$ in $Q$
            end if
        end for
    end while
    return $path_{max}$
```

Problem 3. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be an undirected weighted graph with weight function *weight*.

P3.1. Consider a minimum spanning tree T of $\mathcal{G}$ such that edge $(m, n) \in \mathcal{E}$ is part of T. Prove that T is still a minimum spanning tree if we reduce the weight *weight*(m, n) by $k \geq 0$ (hence, the new weight of edge (m, n) is *weight*$(m, n) - k$).

To prove that T is still a minimum spanning tree if we reduce the weight of edge (m, n) by $k \geq 0$, we need to show that:

T is still a spanning tree of G, meaning it connects all the nodes of G and does not have any cycles. T is still a minimum spanning tree of G with the new weight function. First, let's prove that T is still a spanning tree of G after the weight reduction. Since we are only reducing the weight of edge (m, n), all other edges in the original minimum spanning tree T are still included in the new spanning tree. Hence, T still connects all the nodes of G.

To show that T does not have any cycles, we can use a proof by contradiction. Suppose that after the weight reduction, T has a cycle C that includes the edge (m, n) with reduced weight. Then, we can remove any edge in C that is not in the path between m and n in T, and we obtain a new spanning tree with a smaller total weight than T, contradicting the assumption that T is a minimum spanning tree of G. Therefore, T is still a spanning tree of G with no cycles.

Next, we need to show that T is still a minimum spanning tree of G with the new weight function. To do this, let T' be any other spanning tree of G with a total weight less than T after the weight reduction. Then, the sum of weights of all edges in T' is:

sum(T') = sum(T) - k + weight(m, n) - k

Since weight(m, n) is reduced by k, the total sum of weights of T' is less than that of T by 2k. Therefore, T is still a minimum spanning tree of G with the new weight function.

Hence, we have proved that T is still a minimum spanning tree of G if we reduce the weight of edge (m, n) by $k \geq 0$.

BASICALLY given any other spanning tree is the sum of all its edges. The most it can improve it its sum - k. But since T is minimal before the - k, then it still wins since its min(sum) - k

P3.2. We say that a set of edges $\mathcal{E}' \subseteq \mathcal{E}$ is a *connecting set* if there is a path between all pairs of nodes $(m, n) \in \mathcal{N} \times \mathcal{N}$ using only the edges in $\mathcal{E}'$. We say that a connecting set is minimal if the sum $\sum_{e \in \mathcal{E}'} weight(e)$ is minimal among all connecting sets. Argue in which cases a minimal connecting set is a minimal spanning tree.

A minimal connecting set is a minimal spanning tree if and only if the graph G is connected.

Proof:

First, suppose that G is connected. Then any minimal connecting set E' must include exactly |N| - 1 edges, where |N| is the number of nodes in G, in order to form a spanning tree. Since a minimal connecting set E' is a tree that connects all nodes in G, it has no cycles, and hence it is acyclic. Furthermore, since E' is a tree, it is connected, meaning that there is a unique path between any pair of nodes in E'. Finally, since E' is a minimal connecting set, it must be the case that there is no other set of edges that connects all nodes in G with a smaller total weight. Therefore, E' is a minimal spanning tree.

Now suppose that G is not connected, and let G1, G2, ..., Gk be the connected components of G. Then any minimal connecting set E' must include exactly k - 1 edges, in order to connect all of the connected components. However, it is possible to construct a set of edges that connects all of the connected components with a smaller total weight than any spanning tree that includes edges within the connected components. Specifically, we can choose any edge (m, n) that connects two different connected components, and remove all edges within the connected components, keeping only the edges that connect to m and n. The resulting set of edges connects all of the connected components, and has smaller total weight than any spanning tree that includes edges within the connected components. Therefore, a minimal connecting set is not necessarily a minimal spanning tree when G is not connected.

Problem 4. Consider a communication network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ in which the nodes represents network devices (e.g., computers, wireless access points, switches, and routers) and the edges represent a communication channel (e.g., a network cable or a wireless connection). In this graph, every edge $(m, n) \in \mathcal{E}$ has a weight $0 \leq weight(m, n) \leq 1$ that indicates the probability of *failure-free delivery* when a message is sent from m to n (the reliability of the communication channel).

You may assume that the probability that a message sent by m to n is delivered without failures is independent of the rest of the network. Provide an efficient algorithm that computes the most reliable communication-path between two nodes (such that the probability of failure-free delivery is maximal and the probability of failures is minimal).

Assignment Details

Write a report in which you solve each of the above problems. Your submission:

1. must be a PDF file;
2. must have clearly labeled solutions to each of the stated problems;
3. must be clearly presented;
4. must *not* be hand-written: prepare your report in LaTeX or in a word processor such as Microsoft Word (that can print or exported to PDF).

Submissions that do not follow the above requirements will get a grade of zero.

Grading

Each problem counts equally toward the final grade of this assignment.