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

Deadline: March 31, 2023

Department of Computing and Software
McMaster University

Please read the *Course Outline* for the general policies related to assignments.

Plagiarism is a *serious academic offense* and will be handled accordingly.
All suspicions will be reported to the *Office of Academic Integrity*
(in accordance with the Academic Integrity Policy).

This assignment is an *individual* assignment: do not submit work of others. All parts of your submission *must* be your own work and be based on your own ideas and conclusions. Only *discuss or share* any parts of your submissions with your TA or instructor. You are *responsible for protecting* your work: you are strongly advised to password-protect and lock your electronic devices (e.g., laptop) and to not share your logins with partners or friends! If you *submit* work, then you are certifying that you have completed the work for this assignment by yourself. By submitting work, you agree to automated and manual plagiarism checking of all submitted work.

Late submission policy. Late submissions will receive a late penalty of 20% on the score per day late (with a five hour grace period on the first day, e.g., to deal with technical issues) and submissions five days (or more) past the due date are not accepted. In case of technical issues while submitting, contact the instructor *before* the deadline.

Problem 1. Consider a remote community of N houses $h_1, \ldots h_N$. 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 $h_1, \ldots h_N$. 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). The algorithm is correct since getting the minimum spanning tree with the graph we set up in 1.1 will return the lowest cost to connect all houses with at least 1 fiber connection.

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, ∞
2: for n ∈ N do
3:    path_max :=DIJKSTRAS(G, weight, n)
4:    if path_max < smallest_max then
5:       central_n :=n
6:       smallest_max :=path_max
7:    end if
8: end for
```

Algorithm DIJKSTRAS($\mathcal{G} = (\mathcal{N}, \mathcal{E}), weight, s \in \mathcal{N}$)**:**

```
1: path, cost := [n →?|n ∈ N], [n → ∞|n ∈ N]
2: path[s], cost[s] := s, 0
3: path_max := 0
4: Q := a minimum-priority queue that holds node s with priority 0.
5: while Q ≠ ∅ do
6:    Pop node m with lowest priority with from Q
7:    for all edges (m, n) ∈ E do
8:       if cost[m] + weight((m, n)) < cost[n] then
9:          path[n], cost[n] := m, weight((m, n)) + cost[m]
10:         if cost[n] > path_max then
11:            path_max :=cost[n]
12:         end if
13:         Update n in Q such that n has priority cost[n] in Q
14:      end if
15:   end for
16: end while
17: return path_max
```

I used an adjacency list graph representation.

The algortihm is correct because we run Dijkstra's on every node to find the shortest path to all nodes from that one node. By modifying it a bit, we can keep track of the maximum shortest path from that node to any other node. By doing this for every node, we can keep track of the lowest maximum shortest path for each node, which is the most central distribution center.

The complexity is $\sim |N||\mathcal{E}|log(|\mathcal{N}|)$ as Dijkstra's is $\sim |\mathcal{E}|log(|\mathcal{N}|)$ and I run it once per node.

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 show that T is still the minimum spanning tree of $\mathcal{G}$ given one of its edges (m, n) is reduced by $k \geq 0$, we must show that T is still a spanning tree and is still the minimum spanning tree.

Since changing the weight function doesn't change what nodes the edges connect, nor does it change the nodes, we can see that T is still a spanning tree with $N - 1$ edges.

Now to prove that T is still the minimum spanning tree, we will use contradiction.

Assume T is a minimum spanning tree with the sum of its edges as sum_T. When you reduce one edge (m, n) in T by k, another spanning tree S is now the new minimum spanning tree.

Let's say that $sum_{min} \leq sum_T$ is the sum the edges of T with the one edge reduced by k.

We know that to be a minimum spanning tree, it must also be a spanning tree. Every spanning tree has the property of having N-1 edges. This also means there are 2 cases: (i) the spanning tree S has edge

(m, n) in it and (ii) the spanning tree S doesn't have edge (m, n) in it. In case (i) we know since that edge is reduced by k, the sum of its weights is also reduced by k. In case (ii), the sum of its weights would remain the same.

We also know that for all spanning trees in G, the sum of the weights of its edges sum_s is greater (technically or equal to if there is more than 1 MST) than sum_T. This applies for S. This means in case (ii), $sum_{min} \leq sum_T \leq sum_s$ so S cannot be more minimal than T with reduced weight k. In case (i), we know that $sum_s \geq sum_T$ since T was the minimum spanning tree. Now if S also contains the edge with reduced weight k, we get $sum_s - k \geq sum_T - k$, which still holds, meaning T must still be the minimum spanning tree.

This is a contradiction with our claim, so we know that T must still be the minimum spanning tree.

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.

There is no constraint on how many edges a connecting set must have as long as its at minimum $N - 1$ edges (you need that many to connect the graph). For a spanning tree, there must only be $N - 1$ edges and it must be connected.

We know that for a conneceting set, you can connect the graph with $N - 1$ edges. If we are aiming for the minimal connecting set, then the sum of its edges must be minimal compared to all other connecting sets. Assume we have a minimal connecting set with $N - 1$ edges. If we add any single edge to it (given the weight ≥ 1), then this new connecting set would not be minimal as it has a greater weight than the connecting set without this new edge. By this logic, we know that the minimal connecting set must have $N - 1$ edges.

This means that in ALL cases, a minimal connecting set is the minimal spanning tree.

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).

Assume s is the start node and d is the end node.

Algorithm DIJKSTRAS($\mathcal{G} = (\mathcal{N}, \mathcal{E}), weight, s \in \mathcal{N}, d \in \mathcal{N}$)**:**

$path, cost := [n \rightarrow ? | n \in \mathcal{N}], [n \rightarrow 0 | n \in \mathcal{N}]$
$path[s], cost[s] := s, 1$
$Q :=$ 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**
      $path[n], cost[n] := m, weight((m, n)) * cost[m]$
      Update n in Q such that n has priority $cost[n]$ in Q
    end if
  end for
end while

13: **return** recursively return the path from s to d

This is just Dijkstra's algorithm as seen in class, but instead of calculating distances, we calculate probabilities. The probability of two events happening back to back is equal to multiplying the probability of one event with the probability of the second. The "better" probability, is the higher one. This means line 2 was edited to have the start node at a probability of 1. Then lines 7 and 8 changed to check if its greater than and multiplying the probabilities. This is an $\sim |\mathcal{E}|log(|\mathcal{N}|)$ complexity.

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.