CS 3510 Section B: Design & Analysis of Algorithms September 19th 2024

Problem Set 4: Topological Sort, MSTs, Shortest Paths

YOUR NAME HERE Due: September 27th 2024

- Please type your solutions using LaTeX or any other software. Handwritten solutions will not be accepted.
- Your algorithms must be in plain English & mathematical expressions, and the pseudo-code is optional. Pseudo-code, without sufficient explanation, will receive no credit.
- Unless otherwise stated, all logarithms are to base two.
- If we ask for a specific running time, a correct solution achieving it will receive full credit even if a faster solution exists.

Problem 1: House Placement (25 points)

As a suburban planner, you need to place k houses in Ladhaland numbered 1 to k in a $k \times k$ grid. However, you are given two lists of house pairs labeled `rowConstraints` and `colConstraints` of length m and n respectively. For a pair (a, b) in `rowConstraints`, there is a requirement that house a must be in a row strictly above the row that house b is in. For a pair (a, b) in `colConstraints`, there is a requirement that house a must be in a column strictly to the left of the column that house b is.

Design an algorithm that returns the row and column coordinates for each of the k houses given the number of houses k, `rowConstraints`, and `colConstraints`. If it is impossible to fulfill all the requirements in `rowConstraints` and `colConstraints`, return `"Impossible"`. Provide a correctness and runtime analysis.

Solution:

- *Algorithm:*
  Create two directed graphs: one for row constraints (rowGraph) and one for column constraints (colGraph). For each pair (a,b) in rowConstraints, add a directed edge from a to b in rowGraph. For each pair (a,b) in colConstraints, add a directed edge from a to b in colGraph. Perform a topological sort on each of the two graphs using the algorithm given in the lecture, if a cycle is detected, return "impossible". In order of the two graphs after the topology search starting from (0,0) for whichever is larger (m or n), for colGraph add 1 to x-coordinate (y could be any other constant), and for rowGraph add 1 to y-coordinate (x could be any other constant). Return the row and column coordinates for each house.
- *Correctness: The graphs correctly represent the constraints given by the problem. And if a topological sort is possible, it means there is no cycle, and the constraints can be satisfied. The topological order gives a valid sequence to place houses. The topological order directly translates to row and column indices, ensuring all constraints are met.*
- *Time Complexity Analysis:*
  Building the graphs takes O(m+n) time, and the two topology sort takes O(k+m) and O(k+n) time respectively, assigning coordinates should take O(m+n) time, thus the whole thing is O(k+m+n) time.

Problem 2: Populate Banana Centers (25 points)

There are n banana centers in Decatur numbered 1 to n. Your goal is to provide a source of bananas for all the banana centers by building banana farms and conveyer belts to transport bananas.

You have two options for each banana center i: we can either build a farm inside the center with cost f_i, or we can build a conveyer belt from another center. You are given an array `conveyers` where `conveyers`$[i] = [u, v, c]$ which denotes that the cost of building a conveyer belt from centers u to v is c. Conveyer belts are bidirectional and there may be multiple valid connections between the same two centers with different costs.

Design an algorithm to return the minimum cost of populating all banana centers with farms. Provide a runtime and correctness analysis.

Solution:

- *Algorithm:*
  Create a graph and represent each banana center as a node and the conveyor belts are edges with associated costs. Introduce a virtual node that connects to each banana center with an edge whose cost is the cost of building a farm at that center. This allows us to consider the option of building a farm as part of the MST. Use Kruskal's algorithm taught in lectures to find the MST of this graph, which now includes the virtual node. The cost of the MST will be the minimum cost to populate all banana centers with farms.
- *Correctness: By constructing the MST, we ensure that all nodes are connected with the minimum possible cost. The MST will ensure that all nodes are connected with the minimum possible cost, considering both the conveyor belts and the option to build farms. The algorithm naturally handles bidirectional edges, as each edge is considered independently in the MST construction.*
- *Time Complexity Analysis: The sorting step takes O(Elog(E)) where E is the number of edges. (not sure if I understood the question correction but for each banana center i and conveyer[i] gives the info that there are a total of n variables in the conveyers array. Thus E = 2n (adding fi into the array)) since adding total cost would be O(n) at most, the algorithm would be O(nlog(2n)).*

Problem 3: Currency Exchange (25 points)

You are given a list of currencies in Middle Earth: $C = [c_1, c_2, c_3, \ldots, c_n]$. The Middle Earth Bank releases a list of valid exchanges between currencies as a list R of size m where entry $R[i] = [c_u, c_v, e]$ denotes that 1 unit of currency c_u yields e units of currency c_v. For example, at the time of writing this exam, 1 U.S. dollar is equivalent to 0.93 Euros, so you might see an entry [USD, Euro, 0.93], if Middle Earth had USDs and Euros. Given 1 unit of a start currency c_s and a target currency c_t, design an algorithm to compute the most amount of units of c_t you could obtain by converting through valid exchanges. Provide a runtime and correctness analysis.

Note: it is not necessarily true that if you can convert from c_u to c_v, you can convert from c_v to c_u. Additionally, you can assume it's not possible to generate infinite money using valid exchanges.

Solution:

- *Algorithm:*
  *The algorithm is based off of Bellman-ford algorithm which finds the shortest path in a graph and detect if there exist a negative weight cycle. Here our update equation is dist(v) = max(dist(v), dist(u) * l(u, v)). Also dist(u) = infinity is changed to dist(u) = 0 and dist(s) = 1. And we find c_t and return its value for the most amount of units.*
- *Correctness: The algorithm computes the maximum amount of the target currency by using the optimal substructure and updates the maximum amounts obtainable for each currency based on the exchange rates. After n-1 iterations, the dist dictionary stores the optimal values, assuming no negative cycles exist.*
- *Time Complexity Analysis: Initialization of dist and graph takes O(n + m) time. The problem is just a bellman-ford algorithm at heart which is O(n * m). O(n + m + n * m) so runtime will be O(n * m)*

Problem 4: Research Stations (25 points)

There are several research stations marked with numbers from 1 to n. These stations are linked by m cables, each having a specific time t it takes for a message to travel through. Each research station is configured to simply split and relay a received message from one cable to all the other cables.

Due to the poor quality of the wires, a message that takes more than t_{limit} time to transmit will simply be lost and corrupted. For example, if station 1 is linked by a 5 second cable to station 2 and station 2 is linked by another 5 second cable to station 3, station 3 will not be able to receive a message from station 1 if $t_{limit} < 10$. Therefore, we need to find the research station that can transmit a message that will be received by the most stations. This station will be designated as the central station.

Design an algorithm to find this central station given the number of stations n, the m cables each having specific transmission time t, and the message transmission time limit t_{limit}. Provide a runtime and correctness analysis.

Solution:

- *Algorithm:*
  A variant of Dijkstra's algorithm is used. Changing the most inner if statement from if (adjacent vertex is not visited) enqueue adjacent vertex and cumulative distance t to if (adjacent vertex is not visited and cumulative distance t $< t_{limit}$*) enqueue adjacent vertex and cumulative distance t. Then returning the visited sets.*
  This will ensure that nodes are only added if it is below the t $< t_{limit}$ *value. This entire method is called in a for loop from 1 to n, using each node as the start node. And the Max value is used as the return node.*
- *Correctness: The modified Dijkstra's algorithm ensures that we only consider paths where the cumulative transmission time is less than tlimit. By iterating over all stations as potential starting points and running Dijkstra's algorithm, we systematically explore all possible paths and count the reachable stations for each starting point.*
- *Time Complexity Analysis: Building the graph would be O(m) runtime, and the outer loop of using every node as starter node would be O(n). Each Dijkstra's algorithm would be* $O(n^2) runtime, thus in total would be O(m + n^3), which would be O(n^3) runtime.$