CMPSC 465 – Spring 2021 — Solutions to Homework 6

Josiah Kim, SID 948500821, `juk483`

March 12, 2021

1. Getting started

(a) I did not work in a group.

(b) I did not consult with any of my group members.

(c) I did not consult any non-class materials.

2. Linearization

(a) A(1, 14), B(15, 16), C(2, 13), D(3, 10), E(11, 12), F(4, 9), G(5, 6), H(7, 8)

(b) There are two source vertices and two sink vertices.
Sources: A, B
Sinks: G, H

(c) By ordering the vertices by post numbers in decreasing order we get: B, A, C, E, D, F, H, G

(d) There are four linearizations. Two vertices have two paths ($2^2 = 4$) that are possible with linearization conditions.

3.DFS Variations

(a) ALGORITHM:

This process will require the algorithm to find every path from city s to city t. The function will take the arguments G and s which signify the undirected graph and the starting city, repectively.

First, the algorithm will remove all the edges with lengths $\geq L$.

Then, the algorithm will proceed to run a BFS on the new graph.

If t exists within the queue after the completion of BFS, then there is a feasible route from s to t.

CORRECTNESS:
A route is feasible if and only if the length of every edge in the route is $\leq L$ (reachable). The algorithm will remove the non-reachable vertices at the start. We know that BFS ignores other connected components. We can be confident in the fact that if t is in the queue, then s and t are in the same connected component and that all the edges are $\leq L$.

RUNTIME:
In the worst case, the algorithm will visit every vertex and edge. This is the case if every path to t is feasible. Therefore, in the worst case, the algorithm would be a simple BFS algorithm which has a runtime of $O(|V| + |E|)$.

(b) ALGORITHM:
The function for this algorithm will take the arguments G, s, t, mpg, and cap. The undirected graph, the starting city, the ending city, the fuel economy of the vehicle, and the fuel capacity of the vehicle, respectively.

The values for minimum tank capacity can be seen in numerous ways. First, if the summation of the lengths of all the edges in the path from s to $t = mpg * cap$, then one full tank of fuel is needed for the entirety of the travel. If the summation of the lengths of all the edges in the path from s to $t = \frac{1}{7}4(mpg * cap)$ then only a quarter tank is needed for the entire travel. On the other hand, if the summation of the lengths of all the edges in the path from s to $t = 24(mpg * cap)$, then 24 tanks of gas are needed for the entire travel.

The algorithm will first need to see if there is a feasible path from s to t using the algorithm from (a). If the function returns true, then the algorithm will traverse the graph via Djikstra's Algorithm to find the shortest path. Once the shortest path is found, the algorithm will divide the shortest distance from s to t by the $mpg * cap$ and return the value. The value will tell you the minimum fuel tank capacity needed to travel from city s to t.

CORRECTNESS:
Djikstra's Algorithm will give us the shortest path between two vertices. In this case, the shortest path between s and t in miles. After we get the shortest path between the two cities, we can divide that by the new car's $mpg * cap$ to get the amount of fuel tank capacity needed to make the travel from s to t. If the number is < 0, then the travel can be completed with just one tank without filling up. If the number is > 0, then the car would need at least one re-fuel during the travel. If the number is $= 24$, then the car would need 24 re-fuels during the travel and so on.

RUNTIME:
The algorithm will run both a modified BFS and Djikstra's Algorithm. Therefore, the runtime will result in $O((|V| + |E|)log|V|)$.

4.One-Way Streets

(a) We can view the entirety of State College as a directed graph. The directed edges will be the one way streets and the vertices will be the intersections.

This can be solved in linear time with a BFS. Since the algorithm ignores all non-reachable vertices, only vertices that are reachable will be added to the queue. Therefore, we can check to see if every vertex exists within the queue that the algorithm produces. This would mean that every intersection is reachable via one-way streets.

(b) To drive from town hall, to another location(s), then back to town hall means there exists a single cycle in the theoretical graph. Therefore, the entire graph must be a strongly connected component.

Assume that the town hall, T, is a vertex in a directed graph, G. We can run $Metagraph(G^R)$ on the entire graph. If the function returns a metagraph with just one vertex, then the mayor is correct and the town hall can be reached from any where using one-way streets. However, if the function returns a meta-graph with no vertices, then the mayor is incorrect in her claim.

The metagraph function will visit each and every vertex and edge once using BFS so we know that this can be done in linear time.