Name: Harshit Jain
Access ID: hmj5262
Recitation: 8

Problem 0 | **Points:**

Acknowledgements

(a) I did not work in a group.

(b) I did not consult without anyone my group members.

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

Problem 1

Points:

DFS variations

(a) **Algorthm:** It is given that the car can only hold enough gas to cover L miles while any stretch of highway (l_e) $e \in E$ may or may not be greater than L. Therefore, initially we can simply remove edges (e) of a greater value. Now, the next step is to see if the city t is still reachable from s. To check that, we will be using Breadth First Search (BFS) algorithm. So if vertex t is reachable, then the path is feasible otherwise, it is not feasable.

Runtime: Running BFS will take $O(|V|+|E|)$.

(b) Here, we will modify Dijkstra's algorithm to find paths that minimize the maximum weight of any edge on the path (instead of the path length).

Here, we take minimum of max(visited, not visited adjacent).

Algorithm 1: ModifiedDijkstra

Input : $G = (V, (E, \ell_e))$ (in adjacency list format)
Output: A feasible path exists or not

for *all* $u \in V$ **do**
    dist$(u) = \infty$
    prev$(u) =$ nil
end for
dist$(s) = 0$
Let H be a priority queue constructed with all nodes in V using *dist* as the key
while H *is not empty* **do**
    $u =$ deleteMin(H)
    forall *edges* $(u, v) \in E$ **do**
        if $dist(v) > max(dist(u), l(u, v))$ **then**
            dist$(v) = \max(\text{dist}(u), 1(u, v))$
            prev$(v) = u$
            decreaseKey(H, v)
        end if
    end forall
end while

Runtime: Time complexity is same as Dijkstra's algorithm i.e. $O((|V|+|E|)log|E|)$.

Problem 2

Points:

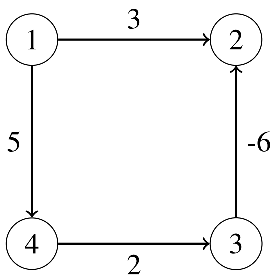
Shortest bitonic paths

(a)

(b)

Problem 3 | **Points:**

Dijkstra's on negative

(a) 

Starting node : 1

Here, the Dijkstra's algorithm give the shortest path to node 2 as 3 however if we follow along the path $\mathbf{1} \rightarrow \mathbf{4} \rightarrow \mathbf{3} \rightarrow \mathbf{2}$, we get the shortest path as $(5+2-6) = \mathbf{1}$. So, clearly the Dijkstra's algorithm does not produce the correct algorithm.

(b) When the input graph contains negative edges, the following statement from the handout does not necessarily hold:

- The *dist* for values already in R is not modified.
- Also, because x is in R, it must have removed from H during a previous iteration and had all its outgoing edges updated.

The *dist* for values already in R can still be modified if we encounter a negative edge in a different path which leads to the value already in R and results in the shorter path (value) than the one it had previously in R. If the edge weight is negative, then adding an edge can indeed make the path length shorter and this is where Dijkstra's Algorithm fails.