1 Homework 4

Name: Kangdong Yuan

1.1 problem1

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.

1.2 problem2

a).
pre and post number of each vertices

$node$	$pre-number$	$post-number$
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) in directed graph, the source is the vertex with 0 in-edge, the sink is the vertex with 0 out-edge. A, B are sources, G, H are sinks

c). the linearization of this order is the descending order of post number: B, A, C, E, D, F ,H, G

d). if we don't follow the alphabetically, only vertices C and F have fixed order, but the vertex in vertices pair (A,B), (D,E) and (G,H) can be access at any order. For example, we can put B or A as the start of linearization. So, there are 2^3 way to arrange these vertices, so there are 8 linearization for this graph.

1.3 problem3

a).First, we need to ensure the edge in graph is eligible. So, for all edge u, if $l_u > L$, then remove this edge. This step take $O(|E|)$ times.
Choosing the vertex (s) as start vertex, then, performing the dfs search, if the dfs can traverse to the vertex (t), return the True. If the vertex (t) cannot be reached, program need to return False. This dfs take $O(|V| + |E|)times$
Because if we start vertex (s) and traverse to vertex (t), it means there is a path from city s to city t, and the paths between two cites are all eligible ($l_u \leq L$).
The total running time to check eligible path is $O(|V| + |E|)$

b). if we want to find minimum tank capacity, we need to know the maximum length segment of the path.
it need us to modify the Dijkstra's algorithm to get the result
program find capacity:
Input: graph: G and start s, and end t
Output: maximum distance segment dist(v)
$find - path(G, s, t)$
put all vertex and length pair into priority queue Q
set $dist(s) = 0$
set all other dist of vertex equal to INFINITY
while Q is not empty:
    u=$Q.deleteMin()$
    for every edge (u,v)
        if dist(v) > $max\{dist(v), l_{a,b}\}$
            dist(v) = $max\{dist(v), l_{a,b}\}$
            v.pre=u
return dist(t)

this program go through all the edge from s to t, and lable the distance of each segment of path, after the last iteration of the graph it will return the maximum path length between two cities, which is the maximum tank capacity.
The running time time of this progarm as same as Dijkstra's algorithm which is $O((|V| + |E|)|log|V|)$

1.4 problem4

a). in this problem, I use directed graph $G(V, E)$ to store the data. I will use vertex V to represent intersection, and directed edge E to represent one direction road. For example vertices u, v are two intersections, and edge (u,v) is the road from u to v.
If the we can start from any intersection to any other intersections, the graph G must be strongly connected which mean every vertex is reachable from every other vertex. I use Kosaraju's DFS to check whether graph G is strongly connected. There are three steps for Kosaraju's DFS , (1) do dfs from vertex v (2) reverse the graph G to G^R (3) do the dfs from vertex v for G^R. If there is any unvisited vertex in two dfs, the graph is not strongly connected. So, the mayor's claim is false. The running time for this program is $O(|V|+|E|)$.

b).We use the same graph G as in the first part of the question and label the town hall s. Then do the same thing in part 1 (1) do dfs from vertex s (2) reverse the graph G to G^R (3) do the dfs from vertex s for G^R. If the visited vertex in step 1 is not visited in step 3, we say that town hall can drive to some intersections, but those intersections cannot drive to s. Thus, the mayor's claim is false. This algorithm take $O(|V| + |E|)$ time.