Jia Feng Yu
20908183
CS341 A2
Question 3

This problem can be interpreted with the streets being an edge and the intersections being a vertex, and in terms of the graph created by the city, as taking a undirected graph and modifying it such that it becomes a strongly connected directed graph, that is, for every vertex $u, v \in V$, there exists a directed path through u, v. Specifically, we can assume that the city and the graph created by it must be connected, as otherwise we can just pick 2 vertices in 2 components and show that you cannot form a strongly connected directed graph, therefore trivializing the problem.

Claim: The undirected graph can be transformed into a strongly connected directed graph $\Leftrightarrow$ the undirected graph contains no bridge.

Proof: $\rightarrow$: It is equivalent to show the contrapositive statement: a undirected graph containing a bridge cannot be transformed into a directed graph with the desired properties. Obviously, from figure 3, if there is a bridge, there is no solution to the problem, as we cannot transform a bridge into a directed edge such that a vertex in either of the components that would be created if the bridge were to be cut can be connected to a vertex in the other side. Hence, showing the forward direction.

$\leftarrow$: Assume for the sake of contradiction that for an undirected graph with no bridges, we cannot construst a strongly connected directed graph. This means that there exists at least one pair of vertices u, v such that u is not reachable from v or v is not reachable from u or both in the directed graph. However, since the graph must be connected, this means that there does exist precisely one path from u to v in the undirected graph. This is because if we have 2 or more paths from u to v, we'd get a cycle and thus we can create a directed cycle, thus connecting u and v. If we had no path between u, v, then the graph would be disconnected. Consider any edge along the one path we have between u and v, if we were to remove that edge, then u and v would no longer be connected in the original, undirected graph. Since that edge was part of the only path between u and v, this means u and v are no longer reachable in the undirected graph, ie, they belong in two different components. However, this would imply that the edge we removed was a bridge, which contradicts the fact that we have no bridge in the undirected graph. Hence, such construction is not possible.

Now, we just modify DFS to include both a low and parent array to detect

all edges; the low array represents the minimum discovery time it requires to reach a vertex and its subtrees from the root node (the starting node in DFS) and the parent array represents the parent of the current node. We perform DFS as standard, if we visit a non-visited vertex, we set it to visited and update the low and parent array respectively, similarly to the algorithm we used to find cut vertices. We also set the low value of the parent of the current node to be the minimum of itself and low of the current node. To consider the cut edges, we want to compute the lowest discovery time of the start and end vertex: if low of the end vertex is greater than that of the start vertex, then we have an edge. Then, from the lemma above, if we have 1 or more edges, the problem is unsolvable and if we have no edges, we can produce such a directed graph. Since we are only adding $O(1)$ operations to the basic DFS algorithm, we still retain the overall runtime of $O(|V| + |E|)$.