CS/ECE 374 Spring 2017
Homework 8 Problem 2

Lanxiao Bai (lbai5)
Renheng Ruan (rruan2)
Ho Yin Au (hoyinau2)

Let $G = (V, E)$ be a directed graph. Describe a linear-time algorithm that given G, a node $s \in V$ and an integer k decides whether there is a *walk* in G starting at s that visits at least k distinct nodes.

Solution: If $G = (V, E)$ is a DAG, set each edge's weight to be -1 and find the shortest path, then the sequence of vertices gives the longest walk. We can check if the number of vertices is more than k. So the overall time complexity is $O(|V| + |E|)$.

On the other hand, if G is a strongly connected graph, then the largest number of vertices a walk can reach is the number of vertices in the graph $|V|$. Then this problem can be solved by checking if the $|V| \geq k$ is true.

Then we combine those two cases by using DFS to split a general graph G into several DAGs and/or SCCs. Then we split each vertex that connects to a SCC and set weight to be $|V_{SCC}|$ and all other edges to have weight 1. Then we can run the topological sort algorithm to get the longest path and check if it's greater than k.

■