CS/ECE 374 FALL 2018 Zhe Zhang (zzhan157@illinois.edu)
Homework 7 Problem 1 Ray Ying (xinruiy2@illinois.edu)
Anqi Yao (anqiyao2@illinois.edu)

Let $G = (V, E)$ be *directed* graph. A subset of vertices are colored red and a subset are colored blue and the rest are not colored. Let $R \subset V$ be the set of red vertices and $B \subset V$ be the set of blue vertices.

- Describe an efficient algorithm that given G and two nodes $s, t \in V$ checks whether there is an s-t path in G that contains at most one red vertex and at most one blue vertex. For simplicity assume that s, t do not have colors. Ideally your algorithm should run in $O(n + m)$ time where $n = |V|$ and $m = |E|$. Do not try to invent a new algorithm. Come up with a way to create a new graph G' and use a standard algorithm on G'.

- Here is a small variation where edges are colored instead of vertices. Some of the edges in G are colored red and some are colored blue and the rest are not colored. Let $R \subset E$ be the red edges and $B \subset E$ be the blue edges. Describe an efficient algorithm that given G and two nodes s, t checks whether there is an s-t path that contains at most one red edge and at most one blue edge. Reduce the problem to the one in the previous part.

No proofs necessary but your algorithms should be clear.

Solution:

1. Consider the first part of the question, that we have vertices with different colors instead of edges. The hint suggests that we should come up with a new graph G' and use standard algorithm on it. Since the graph that can applied with DFS should be a tree-like structure(instead of there maybe cycle in it) so that each node will be visited only once. This means that there shouldn't be multiple incoming edges for every vertex in the graph G. When creating G', if there are multiple edges connect to one vertex, we want to copy the vertex with the same number of incoming edges so that each incoming edge will connect to only one vertex. Then, we will have our tree structure that we can apply our DFS to find all $s - t$ path in newly create graph G'. When use DFS solving the problems, we want to have variable keeping count of red and blue vertex it passed in the path. If the variable counts of red and blue when reaching vertex t still are both less or equal to 1, we will return true and say there is a path in the graph G from s to t with at most one red and one blue vertex. The construction for G' will be following:

   - $V = \{(u, v)|$ u is blue, red or no color and $1 \leq v \leq$ number of incoming edge for v in the vertex in the original graph $G\}$. In this case, there will be at most $|E| + |v|$ of vertices if there is no vertex having single incoming edge.
   - There will be exactly one edge from (u, v) to (u', v') is and only if there exist at least one edge between two vertices in the original graph G that (u, v) and (u', v') are copied from. Each path in the graph G' will either contain cycle or not. If it doesn't contain cycle, the number of edge will be the number of vertices in that path minus 1.

Otherwise, if there has one cycle, we will have one more edge since cycle contains the same number of edge as the number of vertices. The largest cycle will be all the vertices in the path contain in the cycle. So the number of edge will be the number of vertex so it's $|E| + |v|$ as well.

The algorithm will just be DFS, here we want to compute all path from s to t and we will keep the count of number of red and blue in the path. So if there is a path reached t with count of red and count of blue both less than 1, we will return true. Otherwise, return false and stating there is not $s - t$ path with at most 1 blue and 1 red. The running time will just be $O(2(|E| + |V|)) = o(m + n)$

2. Now, the question change to that all vertices are without color but edges are in red, blue and no color. We can simply just build a new graph G'' that change the edge of G' to vertex and change the vertex of G' to edge (where G' is the graph construct from part 1). The color of edge will be the same in the color of vertex, all edges after the transition will be colorless. When we want to find the $s - t$ path, we would start from the node that was the out-edge for vertex s in the graph G'. we want to go to the any node that was the incoming edge for the vertex t. In this case, we have multiple starting nodes and multiple ending nodes, any combination that if there are at most 1 blue and 1 red will consider success. In the graph that the problem stated, edges has three types: red, blue and no color. There will be at most three incoming edge for every node since more than three will be useless. The outgoing edge for start node s will have at most number 3 times number of vertices. To make it more efficient, we should draw the graph G' each time we are giving a s and a t, having new separated non-colored vertex that connects to all the nodes that were the outgoing edges of s in graph G. We perform the DFS algorithm described in part 1, it should give running time the same $O(m + n)$.

■