CS/ECE 374 Spring 2017
Homework 7 Problem 1

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

Let $G = (V, E)$ be directed graph. A subset of edges are colored red and a subset are colored blue and the rest are not colored. Let $R \subset E$ be the set of red edges and $B \subset E$ be the set of blue edges. 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 edge and at most one blue edge. Ideally your algorithm should run in $O(n + m)$ time where $n = |V|$ and $m = |E|$.

Solution: This problem is modeled as a directed, unweighted, colored graph, with vertices V and edges E in G.

We need to assign each edge with a color as "None", "Red" or "Blue" as G specified.

To check if there is an s-t path in G is just a connectivity checking problem, but with constrains that at most 1 edges each could be in R and B, which can be easily implemented by using a boolean flag.

As a result, we can use BFS algorithm, but with a little special rule, that if a red or blue edge is already added to the path, no vertices that can be reached through colored edge should be added in.

This modification functions like when the first red / blue edge was used, all the other red/blue edges will be removed from the graph, and the problem reduced to a new problem that checks the connectivity of the current vertex and the termination t with BFS, and the correctness of the algorithm in each stage is guaranteed by the correctness of BFS algorithm to check connectivity.

Here, we give a pseudocode of this algorithm:

```
IsConnectedUnderRule(s, t):
  init Queue ToExplore
  red ← 0
  blue ← 0
  add s to ToExplore
  while ToExplore is non-empty
      v ← Pull(ToExplore)
      if v = t
          return True
      for edge in GetEdges(v)
          if IsRed(edge) and red = 0
              add GetVertex(edge) to ToExplore
              red ← 1
          else if IsBlue(edge) and blue = 0
              add GetVertex(edge) to ToExplore
              blue ← 1
          else if HasNoColor(edge)
              add GetVertex(edge) to ToExplore
          else
              continue
  return False
```

In worst case, BFS need to search all vertices and edges in the graph once, so the time complexity is $O(m+n)$. ■