Let G(V, E) be an undirected graph with the additional property that every edge also has a color, either red or blue. Let u and v be distinct vertices in G(V, E).

(a) Design an efficient algorithm that decides whether or not there exists a path from uto v such that the path contains only red edges. Justify correctness and running time.

(b) Design an efficient algorithm that decides whether or not there exists a path from u to v such that within the path, all blue edges appear after all red edges . Justify correctness and running time.

Note: When we say ”design an efficient algorithm”, we are asking for a detailed description of the algorithm in English, i.e., without any code or pseudocode.

**Solution**

(a) If we want to find a path containing only red edges, we simply need to delete all the blue edges in the graph. We then run DFS on the resultant graph to check if u and v are in the same connected component. The algorithm is correct because a path between u and v that contains only red edges exists only if u and v are in the same connected component. Deleting all the blue edges takes O(E) time, DFS takes O(V+E) time. The running time of the algorithm is thus O(V+E).

(b) We begin by deleting all the blue edges from the graph. We then DFS on the resulting graph and check to see which nodes are in the same connected component as u. Mark all these nodes as reachable from u. We then restore all the blue edges and delete all the red edges. We then DFS on the resulting graph and check to see which nodes are in the same connected component as v. Mark all these nodes as reachable from v.

Now, we check to see if there is a vertex that is both reachable from uusing only red edges and reachable from vusing only blue edges. A path between uand vwhere all the blue edges appear after all the red edges only exists if the aforementioned condition is met. The presence of the vertex implies that the path is present, because every such path will have one such vertex. The total time taken to modify edges is O(E). The two DFS operations take O(V+E). The time taken to iterate through the vertices after is O(V). The running time of the algorithm is thus O(V+E).