

## 3.

First we can call dijkstra for graph  $G$  from  $s$  to  $t$  to find the shortest path, and make that as our  $\beta$ . Next we call the dijkstra for graph  $G$  find shortest path from  $s$  to some vertices  $u$ , and we call dijkstra to the reversed graph  $G^{\text{rev}}$  to find shortest path from some vertices  $v$  to  $t$ . Then we go through each path between each  $u$  and each  $v$ , if total distance  $(\text{dijkstra}(G, s)[u] + \text{dijkstra}(G^{\text{rev}}, t)[v] + \text{dist}(u, v))$  is greater than or equal to  $3\beta$ , then we mark  $(u, v)$  as bad edge. And we can return all the bad edge after the calculation

```
func reverse_graph(graph):
    rev_graph = null
    for vertex u in graph:
        for vertex v, edge weight_edge in graph that connects to u:
            rev_graph.add((v, u)) with weight_edge
    return rev_graph

func bad_edges(G, s, t):
    G_rev = reverse_graph(G)
    s_to_u = dijkstra(G, s)
    v_to_t = dijkstra(G_rev, t)
    beta = s_to_u[t]

    initialize vector bad_edges as empty
    for vertex u in graph:
        for vertex v and edge weight_edge in graph that connects to u:
            dist = s_to_u[u] + v_to_t[t] + weight_edge
            if dist >= 3 * beta:
                bad_edges.append((u, v))
    return bad_edges
```