Name: _____ Reg #: _____ Section: _____

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design and Analysis of Algorithms	Course Code:	CS2009
Program:	BS(Computer Science)	Semester:	Spring 2022
Duration:	30 Minutes	Total Marks:	20
Paper Date:	1-June-2022	Weight	%
Section:	4B	Page(s):	3
Exam:	Quiz 4		

Q1) Given a weighted directed graph in adjacency list representation, write an efficient algorithm to calculate in-degree (number of incoming edges), out-degree (number of outgoing edges), sum of weight of incoming edges, and sum of weight of outgoing edges for each vertex of the graph. Analyze time complexity of your algorithm. [10 Marks]

Solution

Run BFS and use 4 different arrays of size n (n = total vertices) to store in-degree (number of incoming edges), out-degree (number of outgoing edges), sum of weight of incoming edges, and sum of weight of outgoing edges for each vertex.

```
BFS(G, start)
   Create new queue Q
   Q.push(start)
   color[1..n] = White

   while Q is not empty
      u = Q.pop()
      for each node v adjacent to u
         if color[v] = White then
            color[v] = Black
            indegree[v]++
            outdegree[u]++
            sumOfIndegree[v] += weight(u,v)
            sumOfOutdegree[u] += weight(u,v)
            Q.push(v)
```

Time complexity = O(V+E)

Name: _____ Reg #: _____ Section: _____

Q2) You are given a weighted undirected graph G(V, E) and its MST T(V, E'). Now suppose an edge (a, b) ϵ E' has been deleted from the graph. You need to devise an algorithm to update the MST after deletion of (a, b).

Describe an algorithm for updating the MST of a graph when an edge (a, b) is deleted from the MST (and the underlying graph). It's time complexity must be better than running an MST algorithm from scratch. State and explain the time complexity of your algorithm. Analyze time complexity of your algorithm.

You can assume that all edge weights are distinct, that the graph has E edges and V vertices after the deletion, that the graph is still connected after the deletion of the edge, and that your graph and your MST are represented using adjacency lists. [10 Marks]

Solution

The following algorithm has time complexity O(E). Run DFS twice on what remains of the MST, once starting on a and once starting on b to determine the two connected components A and B, which you store in two HashSets. This takes O(E). Then iterate through all edges going out from the elements of A. If an edge leads to an element of B (can be check in O(1) using HashSet), check whether it is the cheapest edge so far. If it is cheaper than any other edge so far, store it. So, finding the cheapest edge can also be done in O(E). In the end, add the cheapest edge to the MST.