CSCI 3333 Practice Quiz DIST2

Problem 1. Determine the truth of the following statements about shortest-path algorithms.

If \( m = \Theta(n) \), the worst-case running time of BFS is \( O(m) \)  □ True □ False

If \( m = \Theta(n^2) \), the worst-case running time of Dijkstra’s is \( \Theta(n^2 \log(n)) \)  □ True □ False

Bellman-Ford is correct (i.e., “works”) for disconnected input graphs. □ True □ False

For graphs with \( n = \Theta(m) \), Dijkstra’s has lower asymptotic worst-case running time than Bellman-Ford. □ True □ False

Problem 2. Determine the truth of the following statements about shortest-path algorithms.

The best-case and worst-case running times of BFS are the same. □ True □ False

The best-case and worst-case running times of Dijkstra’s are the same. □ True □ False

The best-case and worst-case running times of Bellman-Ford are the same. □ True □ False

Dijkstra’s is correct (i.e., “works”) for input graphs with negative-weight cycles. □ True □ False
Problem 3. Complete the following implementation of Dijkstra’s algorithm.

```cpp
bool dijkstras(vector<Node*> &V, Node* source, map<Node*, int> &D) {
    D.clear();
    for (Node* v : V)
        D[v] = numeric_limits<int>::infinity();

    D[source] = 0;

    MinPriorityQueue<Node*> Q;
    for (Node* v : V)
        Q.push(v, D[v]);

    while (Q.size() > 0) {
        Node* cur = Q.front();
        Q.pop();
        for (Node* nei : cur->neighs)
        {
            int w = cur->weights[nei];

            if (D[nei] > D[cur] + w) {
                D[nei] = D[cur] + w;
                Q.decrease(nei, D[nei]);
            }
        }
    }
}
```
Problem 4. Complete the following implementation of the Bellman-Ford algorithm. 
bellman_ford should return whether the graph contains a negative-weight cycle.

```cpp
bool bellman_ford(vector<Node*> &V, Node* source, map<Node*, int> &D) {
    D.clear();
    for (Node* v : V)
        _____[v] = numeric_limits<int>::infinity();
    D[_____] = 0;

    for (int i = 1; i < _____.size(); ++i)
        for (Node* v : V)
            for (Node* nv : v->neighs)
                {
                    if (D[_____] + v->weights[nv] _____ D[nv])
                        D[nv] = D[_____] + v->weights[nv];
                }

    for (Node* v : V)
        for (Node* nv : v->neighs)
            {
                if (_____ + _____ < D[nv])
                    return _____;
            }

    return _____;
}
```
Problem 5. Fill in the blanks with answers based on the graph in Figure 2.

The number of edges in any spanning tree of $G$ is _________.

The weight of any MST of $G$ is _________.

Removing the edge (___________, __________) increases the weight of any MST of $G$.

![Figure 1: The graph for Problem 6.](image)

Problem 6. Fill in the blanks with answers based on the graph $G$ in Figure 2.

The number of edges in any spanning tree of $G$ is _________.

The weight of any MST of $G$ is _________.

Removing the edge (___________, __________) increases the weight of any MST of $G$.

![Figure 2: The graph for Problem 6.](image)