Problem Set #1

Failed

**0.8/5** points earned (15%)

You haven't passed yet. You need at least 80% to pass.

Review the material and try again!  You have 2 attempts every 12 hours.

[Back to Week 1](https://www.coursera.org/learn/algorithms-graphs-data-structures/home/week/1)

Incorrect

0 / 1 points

1.

Given an adjacency-list representation of a directed graph, where each vertex maintains an array of its outgoing edges (but \*not\* its incoming edges), how long does it take, in the worst case, to compute the in-degree of a given vertex? As usual, we use n and m to denote the number of vertices and edges, respectively, of the given graph. Also, let k denote the maximum in-degree of a vertex. (Recall that the in-degree of a vertex is the number of edges that enter it.)

* θ(n)

**This should not be selected**

In Θ(n) time, how can use be sure you've seen all the relevant arcs?

* θ(m)

**Correct**

Without explicitly maintaining a list of incoming edges, you might have to scan all the edges to identify the incoming arcs.

* θ(k)
* Cannot determine from the given information

**This should not be selected**

Incorrect

0 / 1 points

2.

Consider the following problem: given an undirected graph G with n vertices and m edges, and two vertices s and t, does there exist at least one s-t path?

If G is given in its adjacency list representation, then the above problem can be solved in O(m+n) time, using BFS or DFS. (Make sure you see why this is true.)

Suppose instead that G is given in its adjacency \*matrix\* representation. What running time is required, in the worst case, to solve the computational problem stated above? (Assume that G has no parallel edges.)

* θ(m+n)

**This should not be selected**

If m=O(n), this would imply that you're solving the problem despite looking at only a tiny fraction of the adjacency matrix.

* θ(n2)
* θ(n∗m)

**This should not be selected**

You can do better!

* θ(m+nlogn)

**This should not be selected**

If m=O(n), this would imply that you're solving the problem despite looking at only a tiny fraction of the adjacency matrix.

0.8 / 1 points

3.

This problem explores the relationship between two definitions about graph distances. In this problem, we consider only graphs that are undirected and connected. The *diameter* of a graph is the maximum, over all choices of vertices s and t, of the shortest-path distance between s and t. (Recall the shortest-path distance between s and t is the fewest number of edges in an s-t path.)

Next, for a vertex s, let l(s) denote the maximum, over all vertices t, of the shortest-path distance between s and t. The *radius* of a graph is the minimum of l(s) over all choices of the vertex s.

Which of the following inequalities always hold (i.e., in every undirected connected graph) for the radius r and the diameter d? [Select all that apply.]

* r≤d

**Correct**

By the definitions, l(s)≤d for every single choice of s.

* r≥d

**Un-selected is correct**

* r≤d/2

**Un-selected is correct**

* r≥d/2

**Correct**

Let c minimize l(s) over all vertices s. Since every pair of vertices s and t have paths to c with at most r edges, stitching these paths together yields an s-t with only 2r edges; of course, the shortest s-t path is only shorter.

Incorrect

0 / 1 points

4.

Consider our algorithm for computing a topological ordering that is based on depth-first search (i.e., NOT the "straightforward solution"). Suppose we run this algorithm on a graph Gthat is NOT directed acyclic. Obviously it won't compute a topological order (since none exist). Does it compute an ordering that minimizes the number of edges that go backward?

For example, consider the four-node graph with the six directed edges (s,v),(s,w),(v,w),(v,t),(w,t),(t,s). Suppose the vertices are ordered s,v,w,t. Then there is one backwards arc, the (t,s) arc. No ordering of the vertices has zero backwards arcs, and some have more than one.

* If and only if the graph is a directed cycle

**This should not be selected**

The algorithm might get lucky even if the graph is not a directed cycle.

* Always
* Never
* Sometimes yes, sometimes no

**Correct**

Incorrect

0 / 1 points

5.

On adding one extra edge to a directed graph G, the number of strongly connected components...?

* ...will definitely not change (no matter what G is)
* ...never decreases (no matter what G is)

**This should not be selected**

What if the graph is a directed path?

* ...never decreases by more than 1 (no matter what G is)

**This should not be selected**

What if the graph is a directed path?

* ...could remain the same (for some graphs G)

**Correct**

For example, the graph might already be strongly connected.