CS 3100 Quiz Day 1

This packet contains the quizzes for this quiz day. This **cover sheet** is here to provide instructions, and to cover the questions until the quiz begins. **do not remove this cover sheet** until your proctor instructs you to do so.

You will have the entire class period to complete these quizzes. Each quiz is two pages (front and back of one sheet of paper) worth of questions. Make sure to **write your name and computing id at the top of each individual quiz**.

When you are done, you will come to the front of the room and cut off the staple to this quiz booklet. Afterward, you will discard this cover sheet and submit each quiz separately in a different pile. The proctors will be available at the front of the room to clarify this if you have any questions.

This quiz is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

In theory, there is no difference between theory and practice.
But, in practice, there is.

THIS COVER SHEET WILL NOT BE SUBMITTED. DO NOT PUT WORK YOU WANT GRADED ON THIS PAGE

Quiz - Module 1: Basic Graphs

Name ______________________________

1. [8 points] Answer the following True/False questions regarding *graphs and their basic algorithms*.

All *trees* are also *graphs*.	**True**	**False**
All *linked-lists* are also *graphs*	**True**	**False**
All *graphs* are also *trees*	**True**	**False**
A directed graph (without loops) always satisfies $\|E\| \leq \frac{\|V\|^2}{2}$	**True**	**False**
For any graph, it is always the case that $\|E\| \in O(\|V\|^2)$	**True**	**False**
A *Directed Graph* cannot have cycles, which makes them convenient to use.	**True**	**False**
When *BFS* executes, it is possible that a node is put on the queue a second time, but it can never happen a third time.	**True**	**False**
For *BFS*, the queue can contain nodes with two unique distances (e.g., distance 2 and 3), but NOT three unique distances.	**True**	**False**

2. [3 points] Briefly list one disadvantage of representing a graph as an *adjacency matrix*. Also list one disadvantage of the *adjacency list*.

3. [3 points] In one or two sentences, describe how you would modify *Depth-First Search* to detect the existence of a *cross-edge* (that's the one that crosses to a different sub-tree). *HINT: Suppose we see an edge $e = (u, v)$ and v is black. How can we use the start and end times of u and v to determine if e is a cross-edge?*

4. [6 points] This question is about the runtime of *Depth-First Search*. Answer each question below as best you can. In total, this is worth *6 points*. For each question, assume you have an *undirected* graph $G = (V, E)$ unless otherwise stated. You have implemented *dfs-sweep()* and *dfs-visit()* as described in class. *dfs-sweep()* will attempt to call *dfs-visit()* on every node until the entire graph has been discovered. Leave all of your answers as a constant (e.g., 3) or as a function of $|V|$, $|E|$ (e.g., $|V| + |E| - 3$). For these questions, only count calls from dfs-sweep TO dfs-visit (don't count additional recursive calls to dfs-visit).

When running *dfs-sweep*, what is the **fewest** number of calls *dfs-visit* that is possible?	
When running *dfs-sweep*, what is the **most** number of calls to *dfs-visit* that is possible?	
Now suppose the graph is *directed* and *weakly connected* (this means the undirected version of the graph is connected). What is the *most* number of calls to *dfs-visit*?	

NOTHING BELOW THIS POINT WILL BE GRADED

Quiz - Module 2: Advanced Graphs

Name ______________________________

1. [8 points] Answer the following True/False questions regarding *graphs and their basic algorithms*.

Dijkstra's **might** still work with negative cost edges (but not guaranteed).	**True**	**False**
In *Dijkstra's Algorithm*, a node that is not reachable from the start node will still eventually become *known*.	**True**	**False**
When running *Dikstra's Algorithm*, distances to nodes sometimes have to be updated / overwritten with better (smaller) values.	**True**	**False**
A graph can have more than one *minimum spanning tree*	**True**	**False**
A *spanning tree* does not contain *cycles*	**True**	**False**
Prim's Algorithm must execute from a specific *start node* or it may not work.	**True**	**False**
An *indirect heap* increases the space complexity of the min-heap to $\Theta(n^2)$	**True**	**False**
When implementing an *indirect heap*, if two items are swapped in the *heap*, then they must be swapped in the *indirect array* as well.	**True**	**False**

2. [3 points] List the runtimes of each of the algorithms below. Make sure to consider the implementation notes described in column 2.

Algorithm	**Implementation Notes**	**Runtime**
Dijkstra's	Naive array scan to find next known node	
Prim's	Normal (not indirect) min-heap used	
Dijkstra's	Indirect heap used	

3. [3 points] Draw an example graph that has *more than one minimum spanning tree*. Your graph should have *no more than 4 nodes and 4 edges*. Also, make sure to draw what the two unique minimum spanning trees are (this will make it much easier for us to grade)

Original Graph	**MST 1**	**MST 2**

This question will ask you to show your understanding of the *proof of Dijkstra's Algorithm* that we did in class. This question is worth **6 points**, and those points are spread out over multiple questions. Recall that $\delta(s, v)$ is the shortest distances from s to v, $c(e)$ represents the cost of an edge e, and Δ is a constant.

4. [2 points] First, prove the *base case of Dijkstra's Algorithm*. Specifically: $s.d = \delta(s, s)$. A verbal explanation is fine here.

5. [4 points] Eventually, we saw the following formula: $\delta(s, u') + c(e') + \Delta < \delta(s, u) + c(e)$. Explain why we were able to remove the Δ term from the left side of the inequality? What constraint does this put on the *input to the algorithm*?

NOTHING BELOW THIS POINT WILL BE GRADED