CMPSCI 311: Introduction to Algorithms
First Midterm Exam — SOLUTIONS

March 2, 2017.

Name: _____ ID: _____

Instructions:

- Answer the questions directly on the exam pages.
- Show all your work for each question. Providing more detail including comments and explanations can help with assignment of partial credit.
- If you need extra space, use the blank pages at the end of the exam.
- No books, notes, calculators or other electronic devices are allowed. Any cheating will result in a grade of 0.
- If you have questions during the exam, raise your hand.

Question	Value	Points Earned
1	10	
2	10	
3	10	
4	10	
5	10	
Total	50	

Question 1. (*10 points*) For each pair of functions f and g, circle the statements that are true. No justification is needed.

1.1 (*2 points*): $f(n) = 10n,\ g(n) = n + 1000$. *Circle all that apply:*

(a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Omega(g(n))$

Solution: (a) and (b) are true.

1.2 (*2 points*): $f(n) = 5\log n,\ g(n) = (\log n)^2$. *Circle all that apply:*

(a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Omega(g(n))$

Solution: Only (a) is true.

1.3 (*2 points*): $f(n) = 5\log n,\ g(n) = \log(n^2)$. *Circle all that apply:*

(a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Omega(g(n))$

Solution: (a) and (b) are true. Recall that $\log(n^2) = 2\log n$.

1.4 (*2 points*): $f(n) = \sum_{i=1}^{n} i^3,\ g(n) = n^3$. *Circle all that apply:*

(a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Omega(g(n))$

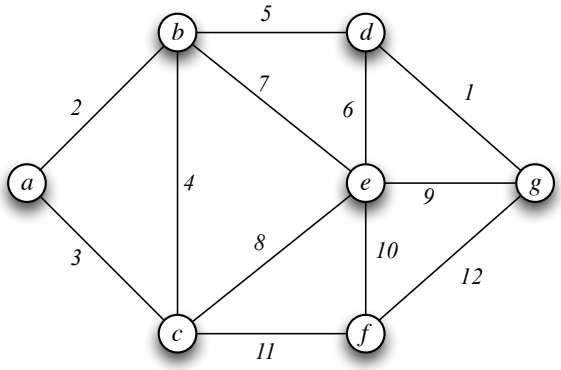
Solution: Only (b) is true. The function $\sum_{i=1}^{n} i^3$ is $\Theta(n^4)$. For the upper bound, note that $f(n) = \sum_{i=1}^{n} i^3 \leq n \cdot n^3 = n^4$, so $f(n)$ is $O(n^4)$. For the lower bound, note that $f(n) = \sum_{i=1}^{n} i^3 \geq \sum_{i=n/2}^{n} i^3 \geq (n/2) \cdot (n/2)^3 \geq (n/2)^4 = n^4/16$, so $f(n)$ is $\Omega(n^4)$. Therefore it cannot be $O(n^3)$.

1.5 (*2 points*): $f(n) = n2^n, g(n) = 3^n/n^3$. *Circle all that apply:*

(a) $f(n)$ is $O(g(n))$ (b) $f(n)$ is $\Omega(g(n))$

Solution: Only (a) is true. Multiply both functions by n^3 and divide both by 2^n, then the comparison is $\frac{n^3}{2^n} f(n) = n^4$ vs. $\frac{n^3}{2^n} g(n) = (1.5)^n$, and we are comparing a polynomial vs. an exponential.

Question 2. (*10 points*) Consider the following graph with distinct edge costs.



2.1 (*2 points*): *List the costs of the edges of the minimum-spanning tree in the order they are added by Kruskal's algorithm. Recall Kruskal's algorithm considers edges in a fixed order.*

Solution: 1, 2, 3, 5, 6, 10

2.2 (*2 points*): *List the costs of the edges of the minimum-spanning tree in the order they are added by Prim's algorithm starting from node a. Recall that Prim's algorithm maintains a single connected component containing the starting node.*

Solution: 2, 3, 5, 1, 6, 10

2.3 (*2 points*): *Recall that Dijkstra's shortest-path algorithm starts with a set* $S = \{s\}$ *of* explored nodes *and defines* $d[s] = 0$. *In each iteration, the algorithm adds some node* v *to* S. *Suppose Dijkstra's algorithm is run with* $s = g$. *List the next two nodes that are added to* S *by the algorithm.*

Solution: d, b

2.4 (*1 points*): *Suppose a BFS is performed starting at node e. How many layers will the BFS tree have?*

Solution: 3

2.5 (*1 points*): *What is the length of the shortest path from a to g?*

Solution: 8

2.6 (*1 points*): *Suppose a DFS is performed starting at node a. How deep will the DFS tree be? (Recall that the depth is the maximum number of edges in the path from the root to to any leaf.)*

Solution: 6 edges, or 7 layers. Full credit is given for "7 layers" but not for "7". (The question gave the definition of depth.)

2.7 (*1 points*): *Is this graph bipartite?*

Solution: No. It contains an odd cycle.

Question 3. (*10 points*) Indicate whether each of the following statements is TRUE or FALSE. No justification required.

3.1 (*2 points*): *Consider a stable matching instance where every student has a different top-ranked college. There is a unique stable matching in this instance.*

Solution: False. For example, suppose colleges also have distinct top-ranked students, but these do not match the student's preferences. The matching where all colleges get their top choice is also stable.

3.2 (*2 points*): *Suppose G is an undirected graph where every node has degree exactly two, and n is odd. Then G is not bipartite.*

Solution: This is true. Every connected component of G must be a cycle. Since the total number of nodes is odd, there must be at least one connected component C with an odd number of nodes. This is an odd cycle, so it is not bipartite.

3.3 (*2 points*): *Suppose G is an undirected graph with edge costs c_e. Let $(S, V-S)$ be any cut. If e is not the lowest-cost edge across cut $(S, V-S)$, then e does not belong to any minimum spanning tree.*

Solution: This is false. A counterexample is the graph with edges (s, a) with cost 1 and (s, b) with cost 2. Edge $e = (s, b)$ is not the lowest-cost edge across the cut $S = \{s\}, V - S = \{a, b\}$, but it is in the minimum-spanning tree.

3.4 (*2 points*): *Suppose G is an undirected graph with edge costs c_e. Let $(S, V-S)$ be any cut. If e is lowest-cost edge across cut $(S, V-S)$, then e belongs to every minimum spanning tree.*

Solution: True. This is the cut property we proved in class.

3.5 (*2 points*): *Let T be a minimum spanning tree of an undirected graph G with edge costs c_e. Suppose the edge costs are all increased by one, i.e., $c'_e = c_e + 1$. Then T is still a minimum spanning tree with the new edge costs c'_e.*

Solution: This is true. Every spanning tree has $n - 1$ edges, so the cost of every spanning tree will increase by exactly $n - 1$. This means that T is still minimum.

Question 4. (*10 points*) Let $G = (V, E)$ be a directed graph where every node has exactly one incoming edge, and there is a node s that has a directed path to every other node.

4.1 (*2 points*): *Draw an example graph with at least 6 nodes that satisfies the conditions stated about G. Identify s in your example.*

Solution: One example is a directed cycle on 6 nodes. The graph G must consist of a single directed cycle, with any number of directed trees rooted at nodes in the cycle and with edges oriented away from the root. A common mistake was draw incorrect generalizations that the node s must have more outgoing edges than other nodes, or that only nodes in the cycle can have multiple outgoing edges. There can be arbitrary branching in the trees leading out of the cycle.

4.2 (*2 points*): *Prove that G is not a DAG.*

Solution: If every node in G has at least one incoming edge, it is not a DAG. See the proof of (3.19) in K&T.

4.3 (*3 points*): *Suppose you want to carefully delete edges from G to turn it into a DAG. What is the smallest number of edges you need to delete to guarantee that the resulting graph G' is a DAG? Prove that your answer is correct.*

Solution: The answer is "one edge". Let s be a node with a directed path to every other node, and let $e = (u, s)$ be the directed edge incoming to s. After deleting e, the resulting graph G' is a DAG. To prove this, consider performing a DFS from s. Since s has a path to every other node, the DFS will visit all nodes. Furthermore, it will produce a tree T rooted at s, with all edges oriented away from s. I claim there are no edges in G that are not in T. Suppose such an edge (u, v) exists. If $v = s$, then this is a contradiction, since we deleted the unique incoming edge to s, so it cannot have an incoming edge from u. If $v \neq s$, this is a contradiction, because v has an incoming edge from its parent in T, so it cannot have another incoming edge from u. Therefore G has no non-tree edges, so it has no cycles.

4.4 (*3 points*): *Now give an $O(m+n)$ time algorithm to identify the specific set of edges to delete. This set should be as small as possible and result in a graph G' that is a DAG. Prove that your algorithm is correct.*

Solution: Start with any node v, and traverse the graph by repeatedly following the unique incoming edge from each node. Mark nodes visited as you do this, and stop when you discover a node t that has already been visited. Delete the unique incoming edge to t. Correctness: The node t belongs to a cycle C. Each node in the cycle has an incoming edge from its predecessor in the cycle, and so it cannot have any other incoming edges. This means that the only nodes that have a path to t are the nodes in the cycle. Since s has a path to every other node, it must be in the cycle; otherwise, it would not have a path to t. But t has a path to s, and therefore it has a path to every other node. Thus, by the reasoning in the previous part of the problem, deleting the incoming edge to t will result in a DAG.

Question 5. (*10 points*) You play on a local game show where you will complete n different challenges and try to make as much money as possible. The ith challenge takes t_i minutes to complete, and starts with a value of v_i dollars, but its value decreases by one dollar per minute. In other words, if you complete challenge i at time $f(i)$, you earn $v_i - f(i)$ dollars. Every challenge has starting value $v_i \geq \sum_{i=1}^{n} t_i$, so you will make *some* money from each challenge. You goal is to order the challenges to earn as much as possible.

5.1 (*2 points*): *Suppose* $v_1 = 10, v_2 = 15, v_3 = 20$ *and* $t_1 = 2, t_2 = 5, t_3 = 1$.

What is the optimal ordering?

How much do you earn?

Solution: The optimal ordering is 3, 1, 2. You earn $(v_3 - 1) + (v_1 - (1+2)) + (v_4 - (1+2+5)) = (20-1) + (10-3) + (15-8) = 19 + 7 + 7 = 33$.

5.2 (*3 points*): *Your friend claims that if all challenges have their value changed to the same number V, this will not change the optimal ordering. Is she right? Either prove she is right or provide a counterexample to show that she is wrong.*

Solution: Yes, she is right. The amount of money you will make is $\sum_{i=1}^{n}(v_i - f(i)) = \sum_{i=1}^{n} v_i - \sum_{i=1}^{n} f(i)$, so the values don't matter. You just want to minimize the sum of the finish times.

5.3 (*2 points*): *One of these orderings is guaranteed to be optimal. Circle the correct one.*

Sort challenges by:

increasing v_i *decreasing* v_i *increasing* t_i *decreasing* t_i

5.4 (*3 points*): *Prove that the ordering you circled always finds an optimal solution.*

Solution: We will prove this by an exchange argument. Suppose O is optimal and there is some pair of challenges i and j such that i comes before j in O but $t_j \leq t_i$. We will call this an inversion. As in the problems we did in class and on the homework, if there is an inversion, there is a consecutive inversion, i.e., a pair of challenges where i comes *immediately* before j in O and $t_j \leq t_j$. Consider swapping the order of these two challenges to create a new ordering O'. Challenge j finishes t_i time units earlier, so you earn t_i extra dollars for challenge j. Challenge i finishes t_j time units later, so you earn t_j fewer dollars for challenge j. The amount you earn for challenges other than i and j does not change. Since $t_j \leq t_i$, the increase from challenge j is at least as much as the decrease from challenge i. Therefore O' is still optimal, and has one fewer inversion. By repeating the argument we can transform O into the greedy solution while preserving optimality; therefore, the greedy solution is optimal.