CMPSCI 311: Introduction to Algorithms
Second Midterm Practice Exam

November 17, 2016.

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 back of a page.

- 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.

- **Hint:** To make sure you get as many points as possible, remember that it is often possible to answer later parts of a question without being able to answer earlier parts of a question.

Question 1. (*10 points*) Indicate whether each of the following statements is TRUE or FALSE. No justification required.

**1.1** (*2 points*): *In any max flow in any graph, the flow along every edge equals the capacity.*

1.2 (*2 points*): *Given a bipartite graph, the maximum matching can be found in polynomial time.*

1.3 (*2 points*): *The nodes of a bipartite graph can always be colored with two colors such that endpoints of each edge get different colors.*

1.4 (*2 points*): *If G is a graph with negative edge weights, the shortest path from s to t has at most $n - 1$ edges.*

1.5 (*2 points*): *The Subset Sum problem can be solved in polynomial time by dynamic programming.*

Question 2. (*10 points*) In the first two parts of this question we consider the following directed, capacitated graph G where the first number on an edge indicates the capacity of that edge and the second number on an edge indicates the current flow on that edge.
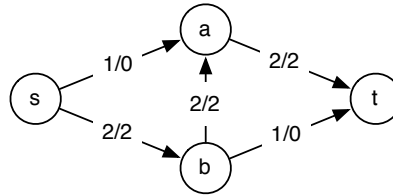
2.1 (*6 points*):   Let (A, B) be an s-t cut where $A = \{s, a\}$ and $B = \{b, t\}$.

1. What is the capacity of the cut?

2. What is the flow across the cut?

3. What is the flow out of s?

4. What is the capacity of the minimum cut?

5. Draw the residual graph corresponding to the above flow:

2.2 (*2 points*):   What is the value of the maximum flow from s to t?

2.3 (*2 points*):   TRUE or FALSE? Suppose (A, B) is a minimum capacity s-t cut in a flow network G, and let e be an edge from A to B. Suppose the capacity on edge e is doubled. Then the value of the maximum flow in G necessarily increases.

Question 3. (*12 points*) Suppose you want to open some coffee shops along one side of a street. The possible locations are labeled $1, 2, \ldots, n$ in that order and you may open a coffee shop at any subset of these locations subject to the constraint that no two coffee shops are adjacent. For example, if $n = 5$ then the set of locations $\{1, 3, 5\}$ is allowed but $\{1, 2, 4\}$ is not. If you open a coffee shop at location i, you earn profit $p_i > 0$ and if you open coffee shops at a set S of locations, your total profit is $\sum_{i \in S} p_i$. You want to find the set of locations that maximizes your total profit.

3.1 (*2 points*): *Your friend suggests a greedy algorithm: consider locations in order of decreasing profit and add the next location unless you've already added an adjacent location. Give an example with $n = 3$ and p_1, p_2, p_3 all different where this approach doesn't maximize your total profit.*

$$p_1 = \qquad p_2 = \qquad p_3 =$$

3.2 (*2 points*): *Another friend suggests that considering opening coffee shops at all even locations or opening coffee shops at all odd locations and picking whichever gives most profit will result in maximizing your total profit. Give an example with $n = 4$ where this is not true.*

$$p_1 = \qquad p_2 = \qquad p_3 = \qquad p_4 =$$

3.3 (*2 points*): *Let $M[i]$ be the max total profit you can earn if you are only allowed to choose locations in the set $\{1, 2, \ldots, i\}$ subject to the constraint that no two are adjacent. If you have already computed $M[1], \ldots, M[i-1]$, how can you directly compute $M[i]$?*

$$M[i] =$$

3.4 (*4 points*): *Write pseudo-code for an efficient dynamic programing algorithm that finds the max total profit. What is the running time of your algorithm? You don't need to prove correctness.*

3.5 (*2 points*): **Extra Credit:** *How could you use your algorithm if the street was circular and this meant that location 1 and location n were neighboring?*

Question 4. (*10 points*)

Let $T = (V, E)$ be a balanced binary tree with $n = 2^k - 1$ nodes including a root named r. Suppose every node $v \in V$ has an associated weight $w(v)$ that can be either positive or negative. We say $T' = (V', E')$ is a *rooted subtree* of T if

1. $V' \subseteq V$ and $E' \subseteq E$.

2. $r \in V'$ and whenever u is a node in T' then the parent of u is also in V'.

3. Whenever $u, v \in V'$ and (u, v) in an edge of T then (u, v) is an edge of T'.

We say the weight of T' is $\sum_{v \in V'} w(v)$.

4.1 (*2 points*): *If $n = 3$, how many rooted subtrees does T have?*

4.2 (*2 points*): *Solve the recurrence $f(n) \leq 2f(n/2) + c$ and $f(1) = c$ where c is a positive constant.*

4.3 (*6 points*): *Design and analyze an efficient algorithm for finding the rooted subtree of maximum weight. Remember to explain why your algorithm is correct and give the running time.*

Question 5. (*10 points*) Here we consider finding the length of the shortest path between all pairs of nodes in an undirected, weighted graph G. For simplicity, assume that the n nodes are labeled $1, 2, \ldots, n$, that the weight w_{ij} of any edge $e = (i, j)$ is positive and that there is an edge between every pair of nodes. In this question, the goal is to solve this via dynamic programming. Note that the algorithm you will develop is *not* the Bellman-Ford algorithm.

5.1 (*2 points*): *Let $D[i, j, k]$ be the length of the shortest path from node i to node j in which the intermediate nodes on the path are all in the set $\{1, 2, \ldots, k\}$. Note that $D[i, i, k] = 0$ for all i and k and that $D[i, j, 0] = w_{ij}$ for all $i \neq j$. A formula for $D[i, j, k]$ with two missing arguments is:*

$$D[i, j, k] = \min\left(D[i, j, k-1], D[i, \quad, k-1] + D[k, j, \quad]\right)$$

when $k \geq 0$. Fill in the two missing arguments.

5.2 (*4 points*): *Write pseudo-code for an efficient dynamic programming (using the above formula) to compute the shortest path between every pair of nodes.* **Hint:** *You need to specify how each $D[i, j, k]$ is computed, the order in which they are computed, and the final output.*

5.3 (*2 points*): *What is the running time of your algorithm? Justify your answer.*