CS 6363.005 Final Exam—Problems and Instructions

May 9, 2019

Please read the following instructions carefully before you begin.

- Write your name and Net ID on the *answer sheets* cover page and your Net ID on each additional page. Answer each of the six questions on the answer sheets provided. One sheet was intentionally left blank to provide you with scratch paper.
- Questions are not necessary given in order of difficulty, so read through them all before you begin writing!
- This exam is closed book. No notes or calculators are permitted.
- You have two hours and 30 minutes to take the exam.
- Please turn in these problem sheets, your answer sheets, and scratch paper at the end of the exam period.
- Feel free to ask for clarification on any of the problems.

1. **(10 points)** Describe and analyze an algorithm to determine in $O(n)$ time whether an arbitrary array of numbers $A[1 .. n]$ contains more than $n/4$ copies of any value. You do not need to justify correctness of your algorithm. An $O(n \log n)$ time algorithm is worth a good amount of partial credit.

   [Hint: Reduce to a procedure we discussed in class.]

2. A palindrome is any string that is exactly the same as its reversal, like I, or DEED, or RACECAR, or AMANAPLANACATACANALPANAMA. Note that a palindrome may have an odd number of characters.

   (a) **(5 out of 10)** Let $X[1 .. n]$ be an array of characters, and let $MaxPalSub(i, j)$ be the length of the longest subsequence of $X[i .. j]$ that is also a palindrome. Fill in the blanks to complete the following recursive definition of $MaxPalSub(i, j)$.

$$MaxPalSub(i,j) = \begin{cases} 0 & \text{if } i > j \\ \_\_\_\_\_ & \text{if } i = j \\ 2 + MaxPalSub(i+1, \_\_\_\_\_) & \text{if } i < j \text{ and } X[i] = X[j] \\ \max\left\{ \begin{array}{l} MaxPalSub(i, \_\_\_\_\_) \\ MaxPalSub(\_\_\_\_\_, j) \end{array} \right\} & \text{otherwise} \end{cases}$$

   (b) **(5 out of 10)** Use dynamic programming to write an iterative algorithm that returns the length of the longest palindrome subsequence in $X[1 .. n]$ based on the above recurrence. What is the running time of your algorithm? You may assume you filled the blanks correctly in the previous part.

3. Suppose we are given an undirected graph G in which every *vertex* has a positive weight.

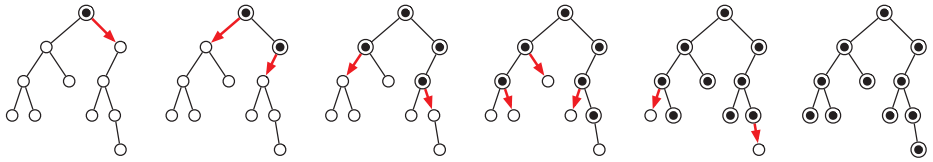
   (a) **(3 out of 10)** Describe and analyze an algorithm to find a *spanning tree* of G with minimum total weight. (The total weight of a spanning tree is the sum of the weights of its vertices.) You do not need to justify correctness of your algorithm.

   [Hint: This part is much easier than the next part.]

   (b) **(7 out of 10)** Describe and analyze an algorithm to find a *path* in G from one given vertex s to another given vertex t with minimum total weight. (The total weight of a path is the sum of the weights of its vertices.) You do not need to justify correctness of your algorithm.

   [Hint: Reduce to standard single source shortest paths by modifying the graph.]

4. **(10 points)** Suppose we need to broadcast a message to all nodes in an n-node binary tree. Initially, only the root node knows the message. In a single round, any node that knows the message can forward it to at most one of its children. See the figure below for an example of a message being distributed in five rounds.



Describe and analyze a recursive or dynamic programming algorithm to compute the minimum number of rounds required to broadcast the message to all nodes in the tree. Your algorithm should run in $O(n)$ time. You do not need to justify correctness of your algorithm.

5. **(10 points)** Suppose we are given a bipartite graph with vertex set $L \cup R$ and edge set E such that every edge joins a vertex in L to a vertex in R. Let $n = |L| + |R|$ be the number of vertices and $m = |E|$ be the number of edges. Describe and analyze an algorithm to find the size of the largest subset of edges such that every vertex in L is incident to at most 3 edges and every vertex in R is incident to at most 2 edges. You do not need to justify correctness of your algorithm. (Note we want a *subset* of edges, meaning each edge is included in the subset at most once.)

6. Both parts ask you to prove a problem is NP-hard. For both parts, you must argue that your reduction is correct.

   (a) **(5 out of 10)** Prove the following problem is NP-hard using a reduction from 3SAT. Given a boolean formula Φ in conjunctive normal form with at most four literals per clause, determine whether Φ has a satisfying assignment.

   [Hint: Read the problem description carefully.]

   (b) **(5 out of 10)** Prove the following problem is NP-hard using a reduction from HAMILTONIANCYCLE in undirected graphs. Given a *complete* undirected graph K_n over n vertices with non-negative weights on the edges, find a minimum weight cycle that includes every vertex. (The weight of a cycle is the sum of its edge weights.)