Name: PID:

COT5405 Design & Analysis of Algorithms

Second Midterm Exam – 11/10/2010

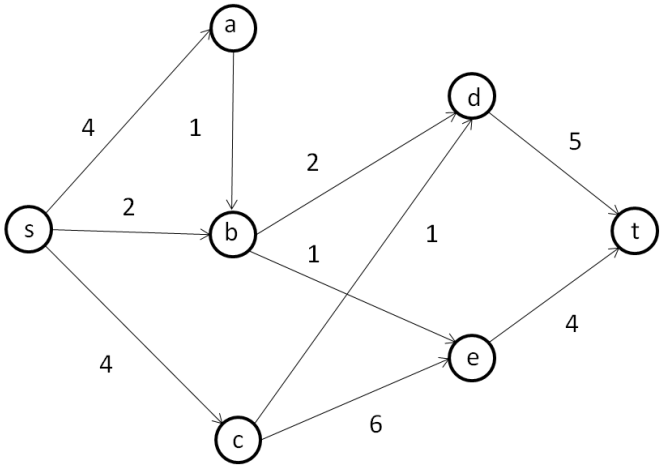
Problem	Points	Points Received
1	30	
2	30	
3	20	
4	20	
Total	**100**	

Name: PID:

1. **[30 points]** Show that the following variation of the maximum flow problem can be reduced to linear programming. Each edge has not only a capacity, but also a lower bound on the flow it must carry. Use the following notations: V for vertex set, E for edge set, s for source, t for sink, $e=(u,v)$ for an edge, f_e for the flow along edge e, c_e for the capacity of e, and l_e for the lower bound on the flow along e.

Name: PID:

2. **[30 points]** Run the Fulkerson-Ford algorithm on the graph below. Always select the *fattest* augmenting path, i.e., the one that admits the largest flow. (Do not use the BFS algorithm to select the shortest augmenting path.)



s-->c-->e-->t flow=4

s-->b-->d-->t flow=2

s-->b-->e-->c-->d-->t flow=1

Max flow = 7

Name: PID:

Name: PID:

3. [**20 points]** You have to cut a wooden stick into pieces. The most affordable company charges money according to the length of the stick being cut. Their procedure requires that they only make one cut at a time.

   It is easy to notice that different selections in the order of cutting can lead to different prices. For example, consider a stick of length 10 meters that has to be cut at 2, 4 and 7 meters from one end. There are several choices. One can be cutting first at 2, then at 4, then at 7. This leads to a price of 10 + 8 + 6 = 24 because the first stick was of 10 meters, the resulting of 8 and the last one of 6. Another choice could be cutting at 4, then at 2, then at 7. This would lead to a price of 10 + 4 + 6 = 20, which is a better price.

   The input to the problem will consist of a positive integer L denoting the length of the stick and array A containing $N \geq 2$ positive integers c_i ($0 \leq c_i \leq L$), representing the places where the cuts have to be done, given in strictly increasing order. For simplicity, you may assume that $A(1) = 0$ and $A(N) = L$ (though of course you do not actually need to make cuts at these locations).

   Your task is to design an *efficient* dynamic programming algorithm to compute the minimum cost for cutting a given stick.

   Let $C(i,j)$ be the be the minimum cost of making cuts at positions $A[i+1], \ldots, A[j-1]$ by considering only the stick starting at $A[i]$ and ending at $A[j]$.

   If $i+1 = j$ (this is the base case)

   $C(i,j) =$ _____

   Else (this is the recurrence)

   $C(i,j) =$ _____

   Output:

   _____

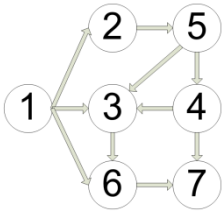
   What is the run-time of this algorithm?

   What is the memory requirement of this algorithm?

Name: PID:

4. [**20 points]** The Hamiltonian Path (HP) Problem is that of deciding whether there exists a *simple* path in a graph which visits each vertex in a graph *exactly* once. In a general graph, this problem is NP-Complete. However, if the given graph is a directed acyclic graph (DAG), then this problem can be solved efficiently.

   Your task is to design an efficient algorithm to solve the HP Problem on a DAG. Note that a HP can start at any vertex, can end at any vertex, and does *not* need to form a cycle (that is, there does not need to be an edge from the end vertex to the start vertex). For instance, on the graph



   to the left, the output to the HP Problem is "yes", due to the path 1␣2␣5␣4␣3␣6␣7.

   The input to the problem is a DAG, and your output should be "yes" or "no" depending on whether the graph has a HP.

   Your answer should consist of the following steps:

   a. Give and define the state of your recurrence (for instance, *L[j]* is the state of the longest increasing subsequence problem and is defined as the longest increasing subsequence beginning at node *j*).
   b. Define the base case(s) of your state.
   c. Give the recurrence.
   d. State the output (or how to use the output to arrive at an answer of "yes" or "no" if your state is not defined as a Boolean value).
   e. State the run-time and memory requirement of your algorithm.

   Hint: It may be easier to solve a related graph problem, and then use that to answer "yes" or "no" to the HP problem.

Name: PID: