INTRO TO ALGORITHMS FINAL EXAM

1 Given n distinct keys in sorted order $k_1 < k_2 < \cdots < k_n$ and we wish to build a BST from these keys. For each key, there is a probability p_i that a search is for k_i. The probabilities satisfy $\sum_{i=1}^{n} p_i = 1$.

Consider the problem of building a BST whose expected search cost is smallest.

a) Prove that this problem satisfies the optimal substructure property. (4%)

b) Let $e[i, j] =$ the expected search cost of an optimal BST for $k_i, \ldots, k_j$
Define $e[i, j]$. (4%)

c) Show how to compute $e[1, n]$ by bottom-up tabulation. (DON'T write any pseudocode, just explain how.)
What is the time complexity of your method? (4%)

2 Consider the ACTIVITY SELECTION problem:

Given a set $S = \{a_1, a_2, \cdots, a_n\}$ of activities. Activity a_i takes place during the time interval $[s_i, f_i)$, $0 \leq s_i < f_i < \infty$, where s_i and f_i are the start time and finish time, respectively. Find a maximum-size subset of mutually compatible activities.

Consider the following two greedy heuristics:

Heuristics A: Choose the first activity to finish.

Heuristics B: Choose the activity that overlaps the fewest other activities.

a) Prove that heuristics A satisfies the greedy-choice property. (4%)

b) Write pseudocode to implement heuristics A, assuming that the activities are sorted in non-decreasing order of finish time.
You may code it recursively or iteratively. (4%)

c) Explain why heuristics B doesn't have the greedy-choice property. (4%)

3 Given a sequence of n operations on a data structure.

Let $c_i =$ the cost of the ith operation, and suppose

$$c_i = i, \quad \text{if } i = 2^k \text{ for some } k$$
$$\quad = 1, \quad \text{otherwise}$$

a) Use aggregate analysis to determine the amortized cost per operation. (4%)

b) Redo a) using an accounting method of analysis. (4%)

4 Consider a sequence of insertion or deletion operations on a dynamic table T discussed in class.

a) Define the potential function for the table T. (4%)

b) Show that the amortized cost of a table insertion operation that doesn't cause an expansion is either 0 or 3. (6%)

c) What is the amortized cost of a table deletion operation that empties the table, i.e. it deletes the only element of the table? (6%)

5 For each statement below, determine if it is certainly true, certainly false, or probably true (equivalently, probably false). ***Justify***. ***No reasons, no credits.*** (16%)

a) If PRIME $\in$ P, then P = NP.

b) If CLIQUE $\in$ P, then P = NP.

c) If the Hanoi Tower problem can be solved in polynomial time, then P = NP.

d) If $\overline{\text{KNAPSACK}}$ $\in$ NP, then NP = co-NP.

6 Consider the knapsack optimization problem

$$\text{Maximize} \sum_{i=1}^{n} v_i x_i \quad \text{subject to} \sum_{i=1}^{n} w_i x_i \leq W, \qquad \text{where } x_i = 0 \text{ or } 1$$

As discussed in class, the dynamic programming solution of this problem takes a time in $O(nW)$.

Let $\text{KNAPSACK} = \left\{ \langle W, v_i, w_i, B \rangle \middle| \begin{array}{l} \text{There exists } x_i = 0 \text{ or } 1 \text{ such that} \\ \sum_{i=1}^{n} w_i x_i \leq W \text{ and } \sum_{i=1}^{n} v_i x_i \geq B \end{array} \right\}$

a) Show how to reduce KNAPSACK to the knapsack optimization problem, assuming that the latter is solved by dynamic programming. (4%)

b) Explain why this reduction doesn't yield a polynomial-time algorithm for KNAPSACK. (4%)

7 Let $G = (V, E)$ be an undirected, weighted complete graph
and $c: V \times V \to \mathbf{Z}$ be the cost function
Let TSP $= \{\langle G, c, k \rangle \mid G \text{ has a tour with cost} \leq k\}$
Prove that TSP is NPC.
(8% – TSP $\in$ NP 2%; polynomial-time reduction 6%)

8 Let LONGEST-PATH-LENGTH be the problem of determining the length of the longest simply path between two vertices u and v in a graph G.
Let LONGEST-PATH
$= \{\langle G, u, v, k \rangle$: There is a simple path from u to v in G of length $\geq k\}$
Prove that if LONGEST-PATH $\in$ P, then LONGEST-PATH-LENGTH can be solved in polynomial time. (8%)

9 Consider the following polynomial-time approximation algorithm for the △TSP problem:
APPROX-TSP-TOUR(G, c)
Select a vertex $r \in G.V$ to be a "root" vertex
Find a minimum spanning tree T for G from root r using MST-PRIM(G, c, r)
Let $L =$ the list of vertices visited in preorder tree walk of T
return the Hamiltonian cycle H that visits the vertices in the order L
a) Prove that APPROX-TSP-TOUR is a 2-approximation algorithm. (6%)
b) Show that the ratio 2 is the best possible. (6%)