Advanced Algorithms

Midterm Examination

CSIE210048

National Chi Nan University

Dec. 2, 2008

Problem 1 Obtain a set of optimal Huffman codes for the eight messages $(M_1, M_2, \ldots, M_8)$ with access frequencies $(q_1, q_2, \ldots, q_8) = (5, 10, 6, 2, 7, 9, 4, 8)$.

Problem 2 The 0/1 knapsack problem is defined as follows:
Given positive integers $P_1, P_2, \ldots, P_n$, $W_1, W_2, \ldots, W_n$, and M, find $x_1, x_2, \ldots, x_n$ where $x_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} P_i x_i$$

is maximized subject to

$$\sum_{i=1}^{n} W_i x_i \le M \quad .$$

The greedy algorithm which attempts to solve the 0/1 knapsack problem is described as follows:

a. Let $C_i = \frac{P_i}{W_i}$ and $x_i = 0$ for $1 \le i \le n$. Let $S = \{1, 2, \ldots, n\}$.

b. Choose $j \in S$ such that $C_j \ge C_i$ for all $i \in S$. If $M \ge W_j$, then subtract W_j from M, remove j from S, and set $x_j = 1$.

c. Repeat Step b until the 'if condition' does not hold (i.e., $W < W_j$).

Show that the above greedy algorithm does not always yield an optimal solution for the 0/1 knapsack problem.

Problem 3 Show that to find the largest number in a list of n numbers requires at least $n - 1$ comparisons.

Problem 4 Solve the recurrence $T(n) = 2T(\sqrt{n}) + 1$ by following these steps:

a. Set $n = 2^m$ and get a recurrence for T over m;

b. Replace $T(2^m)$ by $S(m)$ and get a recurrence for S;

c. Solve $S(m)$ and get back the original result for $T(n)$.

Problem 5 Show that the minimum spanning tree for a graph is unique when the weights on edges are all different.

Problem 6 Can Prim's algorithm be applied to finding the minimum spanning tree for a graph with negative weights? That is, we are given any connected graph whose edges have weights that can be either positive or negative, and we want to apply Prim's algorithm to this graph in order to find its minimum spanning tree. Please do not simply give a true or false answer. Try to JUSTIFY your answer.

Problem 7 Suppose we are given two algorithms A and B that can solve the same problem, and we know that the time complexity for A is $O(n^2)$ and for B is $O(n^3)$. Can we conclude that algorithm A is more efficient than B? Justify your answer.

To show = to prove!