CS473: Algorithms I **11.07.2011**

Problems

1. Suppose that instead of always selecting the first activity finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

2. Not just any greedy approach to the activity selection problem produces a maximum size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time.

3. Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue that your algorithm is correct.

4. Describe an efficient algorithm that, given a set $\{x_1, x_2, \ldots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

5. What is an optimal Huffman code for the following set of frequencies?
a: 12 b: 23 c: 8 d: 35 e: 5 f: 21 g: 15

6. Show that if a Decrement operation were included in the k-bit counter example, n operations could cost as much as $\Theta(nk)$ time.

7. If the set of stack operations included a Multipush operation, which pushes k items onto the stack, would the O(1) bound on the amortized cost of stack operations continue to hold?

8. Suppose we wish not only to increment a counter but also to reset it to zero (i.e. make all bits in it 0). Counting the time to examine or modify a bit as $\Theta(1)$, show how to implement a counter as an array of bits so that any sequence of n Increment and Reset operations takes time O(n) on an initially zero counter. (Hint: Keep a pointer to the high-order 1.)

9. Consider an ordinary binary min-heap data structure with n elements supporting the instructions Insert and Extract-Min in O(log n) worst case time. Give a potential function Φ such that the amortized cost of Insert is O(log n) and the amortized cost of Extract – Min is O(1), and show that it works.

10. Suppose that instead of contracting a table by halving its size when its load factor drops below ¼, we contract it by multiplying its size by 2/3 when its load factor drops below 1/3. Using the potential function $\Phi(T) = |2T.num - T.size|$, show that the amortized cost of a TABLE-DELETE that uses this strategy is bounded above by a constant.