CSCI 200

HW #3: Design techniques - greedy method, divide and conquer, dynamic programming

Directions: *Complete your work on a separate sheet of paper. Submit the physical copy of your work at the beginning of class on the specified due date. Show your work. You may work in groups of up to 3 students provided that all students participate in each question. Provide a short preliminary explanation of how an algorithm works before running an algorithm or presenting a formal algorithm description, and use examples or diagrams if they are needed to make your presentation clear.*

1. For each of the following recurrence equations which describe the running time $T(n)$ of a recursive algorithm, use the master method to express the asymptotic complexity (assuming that $T(n) = c$ for $n < d$, for constants $c > 0$ and $d \geq 1$).
   (a) $T(n) = 2T(n/2) + \log n$
   (b) $T(n) = 8T(n/2) + n^2$
   (c) $T(n) = 7T(n/3) + n$
   (d) $T(n) = 4T(n/2) + n^2$
   (e) $T(n) = 3T(n/2) + n^2$
2. For each of the following recurrence equations which describe the running time $T(n)$ of a recursive algorithm, use the master method to express the asymptotic complexity (assuming that $T(n) = c$ for $n < d$, for constants $c > 0$ and $d \geq 1$).
   (a) $T(n) = T(7n/10) + n$
   (b) $T(n) = 16T(n/4) + n^2$
   (c) $T(n) = 4T(n/3) + n \log n$
   (d) $T(n) = 4T(n/2) + n^2\sqrt{n}$
   (e) $T(n) = 2T(n/2) + n^4$
3. For each of the following recurrence equations which describe the running time $T(n)$ of a recursive algorithm, use the master method to express the asymptotic complexity (assuming that $T(n) = c$ for $n < d$, for constants $c > 0$ and $d \geq 1$).
   (a) $T(n) = 9(n/3) + n$
   (b) $T(n) = 8(n/2) + n\sqrt{\log n}$
   (c) $T(n) = T(n/3) + 1$
   (d) $T(n) = 3T(n/3) + \sqrt{n}$
   (e) $T(n) = 9(n/3) + n^2 \log^3 n$
4. Suppose we are given a set of activities specified by pairs of start times and finish times as $T = \{(5,6),(9,11)(3,7),(1,2),(10,12),(6,8),(1,3),(7,9),(1,4),(11,14),(2,5),(4,9),(7,10)\}$. Solve the activity scheduling problem for these tasks.
5. Let $S = \{a, b, c, d, e, f, g\}$ be a collection of items with weight-benefit values as follows: $a(3, \$12)$, $b(6, \$12)$, $c(6, \$9)$, $d(1, \$5)$, $e(2, \$5)$, $f(10, \$10)$, $g(3, \$9)$. For example, item a weighs 3 lbs and is worth a total of \$12. Assume you have a knapsack that can hold a total of 11 lbs. What is an optimal solution to the **0-1** knapsack problem for S? Show your work.
6. Let $S = \{a, b, c, d, e, f, g\}$ be a collection of items with weight-benefit values as follows: $a(3, \$12)$, $b(6, \$12)$, $c(6, \$9)$, $d(1, \$5)$, $e(2, \$5)$, $f(10, \$10)$, $g(3, \$9)$. For example, item a weighs 3 lbs and is worth a total of \$12. Assume you have a knapsack that can hold a total of 11 lbs. What is an optimal solution to the **fractional** knapsack problem for S? Show your work.

7. What is the best way to multiply a chain of matrices with dimensions that are 10×5, 5×2, 2×20, 20×12, 12×4, and 4×60? Show your work (including two tables - one indicating the index k which gives the final multiply index for each subproblem, and the second table which indicates the optimal number of multiplications for each subproblem).

8. Consider the activity selection problem. Suppose that instead of always selecting the first activity to finish, that we instead selected the activity with the latest start time which is compatible with all previous selected activities. Describe how this approach is a greedy algorithm. Either prove that this approach yields an optimal solution or provide a counterexample to disprove.

9. Consider the activity selection problem. Suppose that instead of always selecting the first activity to finish, that we instead select the activity with the smallest interval (finish time minus start time) which is compatible with all previous selected activities. Either prove that this approach yields an optimal solution or provide a counterexample to disprove.

10. Consider the change problem in Austria. The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L shillings of change (that is, you want to use as few coins as possible). In Austria the coins are worth 1, 5, 10, 20, 25, 50 Shillings. Assume that you have an unlimited number of coins of each type. Formally prove or disprove that the greedy algorithm (that takes as many coins as possible from the highest denominations) correctly solves the Change Problem. So for example, to make change for 234 Shillings the greedy algorithms would take four 50 shilling coins, one 25 shilling coin, one 5 shilling coin, and four 1 shilling coins.

For each of the following questions, give an efficient algorithm to solve the problem and specify what kind of approach it takes (e.g., greedy method, dynamic programing, divide and conquer, etc.). Give a correctness argument (explanation, if it is relatively simple, or proof if not) and time analysis. You may use any well-known algorithm or data structure, or algorithm from the text or from class, as a sub-routine without needing to provide details.

11. You are given an array of n numbers, each of which may be positive, negative, or zero. Return the index positions i and j to the maximum sum of the ith through jth numbers.

12. Given two arrays A and B, return the length of their longest common subsequence. If there is no common subsequence, return 0.

13. Given an integer k and an unsorted integer array A, where A consists of n distinct integers each in the range $[-10^n, 10^n]$, return the kth largest element in the array. You must solve it in $O(n)$ time.

14. There are n trading posts along a river, numbered 1 to n as you travel downstream. At any trading post i you can rent a canoe to be returned at any of the downstream trading posts j, where $j \geq i$. You are given a table $R[i, j]$ defining the cost of a canoe which is picked up at post i and dropped off at post j for $1 \leq i \leq j \leq n$. Assume that $R[i, i] = 0$ and that you can't take a canoe upriver (so perhaps $R[i, j] = \infty$ when $i > j$). However, it can happen that the cost of renting from i to j is higher than the total cost of a series of shorter rentals. In this case, you can return the first canoe at some post k between i and j and continue your journey in a second (and maybe third, fourth ...) canoe. There is no extra charge for changing canoes this way. Describe an efficient algorithm to determine the minimum cost of a trip by canoe from each possible departure point i to each possible arrival point j.

15. Suppose we have a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient algorithm to determine which activity should use which lecture hall. (Fun fact! This problem is also known as the interval-graph coloring problem. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible actiities. The smallest number of colors required to color every vertex so that no two adjacent vertices have the same color corresponds to finding the fewest lecture halls neeeded to schedule all of the given activities.)