CS180 Homework 6

Due: 8:00pm, 2/21/2019

1. Given an array $A[1..n]$ of n numbers, we can calculate the maximum in A only by comparing numbers. Give an algorithm that finds the maximum and second maximum using exactly $n + O(\log n)$ comparisons.

2. Suppose $A_n = a_1, a_2, ..., a_n$ is a set of distinct coin types, where each a_i is a positive integer. Suppose also that $a_1 < a_2 < ... < a_n$. The coin-changing problem is defined as follows. Given a positive integer C, find the smallest number of coins from A_n that add up to C, given that an unlimited number of coins of each type are available.

   (a) Suppose $a_1 = 1$. A greedy algorithm will make change by using the larger coins first, using as many coins of each type as it can before it moves to the next lower denomination. Show that this algorithm does not necessarily generate a solution with the minimum number of coins.

   (b) Show that if $A_n = \{1, c, c^2, ..., c^{n-1}\}, c \geq 2$, then the greedy algorithm always gives a minimal solution for the coin-changing problem.

   (c) Design an $O(nC)$ algorithm that gives a minimum solution for the general case of A_n.

3. Given n items $\{1, ..., n\}$, and each item i has a nonnegative weight w_i and a distinct value v_i. We can carry a maximum weight of W in a knapsack. The items are blocks of cheeses so that for each item i, we can take only a fraction x_i of it, where $0 \leq x_i \leq 1$. Then item i contributes weight $x_i w_i$ and value $x_i v_i$ in the knapsack. The continuous knapsack problem asks you to take a fraction of each item to maximize the total value, subject to the total weight does not exceed W. The original knapsack problem is equivalent to say x_i is either 0 or 1 depends whether the item is in or not. Give an $O(n)$ algorithm for the continuous knapsack problem.

4. Recall the problem of finding the number of inversions. We are given an array $A[1..n]$ of n numbers, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $A[i] > A[j]$. The problem is to count the number of inversions in A. Let's call a pair (i, j) is a *significant inversion* if $i < j$ and $A[i] > 2A[j]$. Give an $O(n \log n)$ algorithm to count the number of significant inversions in A.

★ Express your algorithm in a well-structured manner. Use pseudo code and the textbook has good examples to follow. Avoid using a long continuous piece of text to describe your algorithm. Start each problem on a NEW page. Unless specified, you should justify the time complexity of your algorithm and why it works. For grading, we will take into account both the correctness and the clarity. Your answers are supposed to be in a simple and understandable manner and sloppy answers are expected to receiver fewer points.

★ Homework assignments are due on Gradescope. Email attachments or paper submissions are NOT acceptable.

★ Raw photo is NOT acceptable. Upload your homework as a PDF scan by using a scanner or mobile scanner app. Match each problem with your answer on Gradescope. Use dark pen or pencil and your handwriting should be clear and legible.

★ We recommend using LaTeX, LyX or other word processing software for writing the homework. This is **NOT** a requirement but it helps us to grade and give feedback.