600.363/463 Introduction to Algorithms/Algorithms I October 12, 2009

Handout 7: Homework 4

Instructor: Susan Hohenberger *TA: Ming Chuang*

This assignment is due by the start of lecture on October 21, 2009. Please clearly indicate your collaborators.

1. (20 points) *Greedy Rod Cutting.* (CLRS Exercise 15.1-2) Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the *density* of a rod of length i to be p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

2. (20 points) *Currency Exchange.* (CLRS Exercise 15.3-6) Imagine that you wish to exchange one currency for another. You realize that instead of directly exchanging one currency for another, you might be better off making a series of trades through other currencies, winding up with the currency you want. Suppose that you can trade n different currencies, numbered $1, 2, \ldots n$, where you start with currency 1 and wish to wind up with currency n. You are given, for each pair of currencies i and j, an exchange rate r_{ij}, meaning that if you start with d units of currency i, you can trade for dr_{ij} units of currency j. A sequence of trades may entail a commission that you are charged when you make k trades. Show that, if $c_k = 0$ for all $k = 1, 2, \ldots, n$, then the problem of finding the best sequence of exchanges from currency 1 to currency n exhibits optimal substructure. Then show that if commissions c_k are arbitrary values, then the problem of finding the best sequence of exchanges from currency 1 to currency n does not necessarily exhibit optimal substructure.

3. (20 points) *Fibonacci numbers.* The Fibonacci numbers are defined by the following recurrence:

$$\begin{aligned} F_0 &= 0, \\ F_1 &= 1, \\ F_i &= F_{i-1} + F_{i-2} \qquad \text{for } i \geq 2. \end{aligned}$$

   Given an $O(n)$-time dynamic-programming algorithm to compute the nth Fibonacci number.

4. (40 points) *Sonic Fence.* There is a sonic fence around your compound. You have wireless sensor nodes around the compound and the sensors indicate that "the others" are approaching. At minute i your sensors predict that x_i others will reach your fence. Once they cross the fence, the others will attack your headquarters. But right at the moment they are at the fence, you can choose to "zap" them by discharging the huge capacitors that power the fence. Unfortunately, as soon as you zap, the fence takes

time to recharge. In particular, if you let the fence charge up for j minutes, then it will have enough power to zap d_j of the others.

Example: Suppose $(x_1, x_2, x_3, x_4) = (1, 10, 10, 1)$ and $(d_1, d_2, d_3, d_4) = (1, 2, 4, 8)$. The best solution is to zap at times $3, 4$. This would zap a total of 5 others.

(a) Construct an instance of the problem on which the following "greedy" algorithm returns the wrong answer:

BadZap$((x_1, \ldots, x_n), (d_1, \ldots, d_n))$

i. Compute the smallest j such that $d_j \geq x_n$. Set $j = n$ if no such j exists.

ii. Zap fence at time n.

iii. If $n > j$ then BadZap$((x_1, \ldots, x_{n-j}), (d_1, \ldots, d_{n-j}))$.

Intuitively, the algorithm figures out how many minutes (j) are needed to zap the others in the last time slot. It zaps during that last time slot, and then accounts for the j minutes required to recharge for that last slot, and recursively considers the best solution for the smaller problem of size $n - j$.

(b) Given arrays holding the $(x_1, \ldots, x_n)$ and $(d_1, \ldots, d_n)$ values, devise an algorithm that zaps the most "others." Analyze the running time of your solution.