CS 5633 Analysis of Algorithms – Spring 12

3/8/12

7. Homework

Due **3/22/12** before class

1. **Matrix Chain Multiplication**

   (a) The dynamic programming approach for the matrix chain multiplication problem makes many recursive calls by trying out all possible k with $i \leq k \leq j$ in order to split $A_{ij} = A_i A_{i+1}, \ldots, A_j$. Now, consider the greedy approach which selects the k that simply minimizes the quantity $p_{i-1} p_k p_j$, and then simply recursive for this one choice of k only. Give a counter-example which shows that this greedy approach yields a suboptimal solution.

   (b) Show how to perform the traceback in order to construct an optimal parenthesization for the matrix chain multiplication problem *without* using the auxiliary s-table. How much time does this traceback algorithm need? Justify your answer.

2. **Restaurants**

   Let $x_1, x_2, \ldots, x_n$ be n locations on a line (in sorted order). These represent possible positions for opening a restaurant on a given street. Additionally, for each position x_i you are given a value p_i which represents the profit of opening a restaurant at location x_i. You are also given a number $k > 0$.

   The task is to develop a dynamic programming algorithm to determine a set of locations to open restaurants such that 1) every two restaurants are at least distance k apart, and 2) the total profit is maximized.

   You should first identify suitable subproblems and come up with a recursive formulation of the problem. (E.g., $h(i)$ = maximum total profit for opening restaurants at locations $x_1, \ldots, x_i$). Then shortly describe a dynamic programming algorithm and the proper traceback procedure. What is the runtime of your algorithm?

3. **Christmas** (This question was on a PhD qualifying exam.)
   For Christmas, I only had so much money to spend on gifts for n people, and I did not allocate my resources very well. Now, I want to be ready for next Christmas. Naturally, I want a dynamic programming solution for my problem.

   For each person, I can choose either a good, expensive gift or a bad, cheap gift. I want to maximize the happiness of the people I am giving the gifts to. I have four arrays of size n containing positive integers between 1 and n: `Cgood, Cbad, Hgood, Hbad`.

   - `Cgood[i]` indicates the cost of a good gift for person i.
   - `Cbad[i]` indicates the cost of a bad gift for person i.
   - `Hgood[i]` indicates the happiness of person i getting a good gift.
   - `Hbad[i]` indicates the happiness of person i getting a bad gift.

   You can assume `Cgood[i] > Cbad[i]` and `Hgood[i] > Hbad[i]`. I want to maximize the sum of the happiness over all n people, but I only have a total of C money to spend.

FLIP OVER TO BACK PAGE ⟹

(a) Suppose the following are the arrays for $n = 4$ and $C = 10$:

```
Cgood: [2, 3, 4, 3]
Cbad:  [1, 2, 2, 2]
Hgood: [4, 3, 3, 4]
Hbad:  [2, 2, 2, 2]
```

What gift selection maximizes happiness while not exceeding cost? What is the solution for $C = 9$?

(b) Let $h(i, c)$ be the maximum happiness for the first i people with a cost equal or less than c. For example, $h(2, 4) = 6$ in the previous example by choosing a good gift for person 1 (cost 2, happiness 4) and a bad gift for person 2 (cost 2, happiness 2). Provide a recursive definition for $h(i, c)$. That is, show how to calculate h for i people from the values for $i - 1$ people.

(c) Write a dynamic programing algorithm to compute h. What are the asymptotic running time and asymptotic space requirements for your algorithm? Explain your answer.