CMPS 6610 Algorithms – Fall 16

10/4/16

4. Homework

Due **10/11/16** at the beginning of class

1. **Christmas (8 points)**
   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.

   (a) (2 points) 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) (3 points) 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) (2 points) Write a dynamic programing algorithm to compute h.

   (d) (1 point) What are the runtime and the space complexity of your algorithm? Explain your answer.

FLIP OVER TO BACK PAGE ⟹

2. **Matrix Chain Multiplication (6 points)**

   (a) (3 points) 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) (3 points) 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.

3. **Intervals (9 points)**
   Let $A[1..n]$ be an array of n integers (which can be positive, negative, or zero). An *interval* with start-point i and end-point j, $i \leq j$, consists of the numbers $A[i], \ldots, A[j]$ and the *weight* of this interval is the sum of all elements $A[i] + \ldots + A[j]$.

   The problem is: Find the interval in A with maximum weight.

   (a) (2 points) Describe an algorithm for this problem that is based on the following idea: Try out all combinations of i, j with $1 \leq i < j \leq n$. What is the runtime of this algorithm?

   (b) Describe a dynamic programming algorithm for this problem. Proceed in the following steps:

      i. (2 points) Develop a recurrence for the following entity: $S(j) =$ maximum of the weights of all intervals with end-point j.

      ii. (2 point) Based on this recurrence describe an algorithm that computes all $S(j)$ in a dynamic programming fashion, and afterwards determines the end-point j^* of an optimal interval.

      iii. (2 points) Given the end-point j^* describe how to find the start-point i^* of an optimal interval by backtracking.

      iv. (1 point) What are the runtime and the space complexity of your algorithm? Explain your answer.