CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

HW 12

Due November 22th (Friday), 11:59 pm

Formatting: Each problem should begin on a new page. When submitting in Gradescope, try to assign pages to problems from the rubric as much as you can. Make sure you write all your group members' names. For the full policy on assignments, consult the syllabus.

1. (15 pts.)

   Given a pair of strings $X = X_1X_2 \dots X_m$, $Y = Y_1Y_2 \dots Y_n$ their longest common subsequence denoted by $LCS(X, Y)$, is the longest subsequence that is present in both X and Y. Write a poly time dynamic program algorithm to compute the length of $LCS(X, Y)$.

   Solution:

   Let $L[i][j]$ represents the LCS length of $X_1X_2 \dots X_i$ and $Y_1Y_2 \dots Y_n$. We compute $L[i][j]$ recursively as below:

   1. $L[0][0] = 0$, $L[i][0] = 0$ (for $i \in [1, \dots m]$), $L[0][j] = 0$ (for $j \in [1, \dots n]$) (base case)
   2. If $X_i = Y_j$ then $L[i][j] = L[i-1][j-1] + 1$. Here we can match X_i with Y_j.
   3. If $X_i \neq Y_j$ then $L[i][j] = max(L[i][j-1], L[i-1][j])$. Here as X_i can't be matched with Y_j, at least one of them must be deleted.

   Note $L[m][n]$ represents the length of $LCS(X, Y)$. We compute the total $O(MN)$ entries of the DP table. Moreover, each entry can be computed in time $O(1)$. Thus the total running time is $O(MN)$.

2. (15 pts.)

   Suppose you have a large piece of gold foil and you need to plan on how to cut it to make the maximum profit. The size of the gold foil is $M \times N$ where M and N are positive integers. Due to technical limitations, you can only cut gold foil either horizontally or vertically. Each cut results in two smaller rectangle pieces with integral sizes, i.e. the sizes of the smaller pieces are still integers. You can make as many cuts as you want as long as sizes permit.

   When you are done cutting, you can sell your rectangles at the market. There are n buyers at the market. The i^{th} buyer is willing to pay p_i for a rectangle of size $a_i \times b_i$. You can sell as many rectangles to a buyer as you want, including not selling any.

   Design a dynamic programming algorithm that outputs the maximum profit on a piece of gold foil of size $M \times N$. What is the running time?

Solution:

For $1 \leq i \leq M$ and $1 \leq j \leq N$, let $C(i,j)$ be the maximum profit can be obtained from a piece of gold foil of size $i \times j$. Define a function rect as follows:

$$\mathrm{rect}(i,j) = \begin{cases} \max_k p_k & \text{for all products } k \text{ with } a_k = i \text{ and } b_k = j \\ 0 & \text{if no such product exists} \end{cases}$$

Then the recurrence is

$$C(i,j) = \max\{\max_{1\leq k\leq i}\{C(k,j) + C(i-k,j)\}, \max_{1\leq h\leq j}\{C(i,h) + C(i,j-h)\}, \mathrm{rect}(i,j)\}.$$

Base case:

$$C(1,j) = \max\{0, \mathrm{rect}(1,j)\}$$
$$C(i,1) = \max\{0, \mathrm{rect}(i,1)\}$$

The final solution is $C(M,N)$.

The running time is $O(MN(M+N+b))$ as there are MN subproblems and each takes $O(M+N+n)$ time to evaluate.

3. (20 pts.)

You are given a checkboard that has 4 rows and n columns and has an integer written in each square. You are also given a set of $2n$ pebbles and want to place some or all of these on the checkboard (each pebble can be placed on exactly one square) so as to maximize the sum of the integers that are covered by the pebbles.

There is one constraint: for a placement of pebbles: for a placement of pebbles to be legal, no two of them can be horizontally or vertically adjacent squares (diagonal adjacency is fine).

1. Determine the number of legal patterns that can occur in any column and describe these patterns. Call two patterns compatible if they can be placed on adjacent columns to form a legal placement. Let us consider subproblems consisting of the first k columns $1 \leq k \leq n$. Each subproblem can be assigned a type which is the pattern occurring in the last column.
2. Using the notations of compatibility and type, give an $O(n)$ time dynamic programming algorithm for computing an optimal placement.

Solution:

1. There are 8 possible patterns: the empty pattern, the 4 patterns which each have exactly one pebble, and the 3 patterns that have exactly two pebbles (on the first and fourth squares, the first and third squares, and the second and fourth squares).

   Number the 8 patterns 1 through 8, and define $S \subseteq [8]^2$ to be all (a,b) such that pattern a is compatible with pattern b. For each pattern, there are some patterns that are compatible with (for example, every pattern is compatible with the empty pattern).

   We consider the sub-problem $L[i,j], i = 0,1,2,\ldots,n$ and $j \in [8]$ to be the maximal value achievable by pebbling columns $1,\ldots,i$ such that the final column has pattern j.

2. Let $c(j)$ denote the cost of pattern j. It is easy to see that:

$$L[i+1,j] = c(j) + \max_{(k,j)\in S} L[i,k].$$

   The base case is $L[0,j] = 0$ for all j. In order to recover the optimal placement, we should also maintain a back-pointer: $P[i+1,j]$ is the value of k such that $(k,j) \in S$ and $L[i,k]$ is maximal.

   The dynamics programming algorithm is to compute every $L[i,j]$ and $P[i,j]$ in the increasing order of i using the recurrence. After the computation of all entries of L and P, we backtrack the optimal placement through the back-pointers.

   Note that the max operator in the recurrence relation can be implemented in constant time since we only choose a maximum among constantly many candidates. There are $O(n)$ entries to compute and $O(n)$ steps of backtracking. Therefore, the running is $O(n)$.