问题与反馈

2014/9/12

15.2-4

Describe the subproblem graph for matrix-chain multiplication with an input chain of length n. How many vertices does it have? How many edges does it have, and which edges are they?

$$m[i,j] = \begin{cases} 0 & \text{if } i = j\ , \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1} p_k p_j\} & \text{if } i < j\ . \end{cases} \tag{15.7}$$

15.3-5

Suppose that in the rod-cutting problem of Section 15.1, we also had limit l_i on the number of pieces of length i that we are allowed to produce, for $i = 1, 2, \ldots, n$. Show that the optimal-substructure property described in Section 15.1 no longer holds.

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, which depends on the number of trades you make. Let c_k be the 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.

15.4-3

Give a memoized version of LCS-LENGTH that runs in $O(mn)$ time.

```
MEMOIZED-CUT-ROD(p, n)
1  let r[0..n] be a new array
2  for i = 0 to n
3      r[i] = -∞
4  return MEMOIZED-CUT-ROD-AUX(p, n, r)

MEMOIZED-CUT-ROD-AUX(p, n, r)
1  if r[n] ≥ 0
2      return r[n]
3  if n == 0
4      q = 0
5  else q = -∞
6      for i = 1 to n
7          q = max(q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n - i, r))
8  r[n] = q
9  return q
```

模仿一下！！！

```
LCS-LENGTH(X, Y)
1   m = X.length
2   n = Y.length
3   let b[1..m, 1..n] and c[0..m, 0..n] be new ta
4   for i = 1 to m
5       c[i, 0] = 0
6   for j = 0 to n
7       c[0, j] = 0
8   for i = 1 to m
9       for j = 1 to n
10          if x_i == y_j
11              c[i, j] = c[i - 1, j - 1] + 1
12              b[i, j] = "↖"
13          elseif c[i - 1, j] ≥ c[i, j - 1]
14              c[i, j] = c[i - 1, j]
15              b[i, j] = "↑"
16          else c[i, j] = c[i, j - 1]
17              b[i, j] = "←"
18  return c and b
```

15.4-5

Give an $O(n^2)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers.

15.4-6 ⋆

Give an $O(n \lg n)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers. (*Hint:* Observe that the last element of a candidate subsequence of length i is at least as large as the last element of a candidate subsequence of length $i - 1$. Maintain candidate subsequences by linking them through the input sequence.)

15-4 Printing neatly

Consider the problem of neatly printing a paragraph with a monospaced font (all characters having the same width) on a printer. The input text is a sequence of n words of lengths $l_1, l_2, \ldots, l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i \leq j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} l_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines. Give a dynamic-programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.

- Define $extras[i, j] = M - j + i - \sum_{k=i}^{j} l_k$ to be the number of extra spaces at the end of a line containing words i through j. Note that *extras* may be negative.
- Now define the cost of including a line containing words i through j in the sum we want to minimize:

$$lc[i, j] = \begin{cases} \infty & \text{if } extras[i, j] < 0 \text{ (i.e., words } i, \ldots, j \text{ don't fit)}, \\ 0 & \text{if } j = n \text{ and } extras[i, j] \geq 0 \text{ (last line costs 0)}, \\ (extras[i, j])^3 & \text{otherwise}. \end{cases}$$

Our subproblems are how to optimally arrange words $1, \ldots, j$, where $j = 1, \ldots, n$.

Consider an optimal arrangement of words $1, \ldots, j$. Suppose we know that the last line, which ends in word j, begins with word i. The preceding lines, therefore, contain words $1, \ldots, i-1$. In fact, they must contain an optimal arrangement of words $1, \ldots, i-1$. (Insert your favorite cut-and-paste argument here.)

$$c[j] = \begin{cases} 0 & \text{if } j = 0, \\ \min\limits_{1 \leq i \leq j} (c[i-1] + lc[i, j]) & \text{if } j > 0. \end{cases}$$