Tecnológico de Monterrey

Fundamentos de computación

Homework 3

Student:
Jacob Rivera

Professor:
Dr. Hugo Terashima

March 21, 2019

1 Problems

Solve the following problems:

1. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions are $p_0 = 10$, $p_1 = 19$, $p_2 = 29$, $p_3 = 33$, $p_4 = 9$ and $p_5 = 17$. Show and explain each step in the procedure.

$$A_1 = 10 \times 19$$
$$A_2 = 19 \times 29$$
$$A_3 = 29 \times 33$$
$$A_4 = 33 \times 9$$
$$A_5 = 9 \times 17$$

	1	2	3	4	5
1	0				
2	5510	0			
3	15080	18183	0		
4	15282	13572	8613	0	
5	16812	16479	13050	5049	0

	1	2	3	4
2	1			
3	2	2		
4	1	2	3	
5	4	4	4	4

2. Show that a full parenthesization of an n element expression has exactly $n - 1$ pairs of parentheses

   We can see that a parenthesization of one element is none and of two elements is trivial, as the only possibility is (A_1A_2), but when we add a third element, we have the possibilities of $((A_1A_2)A_3)$ and $(A_1(A_2A_3))$, so, by in induction we can see that the number of pairs parentheses is dependent of the number of terms, as $n - 1$

3. Solve problem 15-4 from Cormen et al. Book.

   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 $l_1, l_2, ..., l_n$, 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 non negative 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.

```
PRINT-NEATLY(l[], n, M)
    spaces_at_the_end = new [n][n]
    line_cost = new [n][n]
    c = new [n]
    for i = 1 to n
        spaces_at_the_end[i][i] = M - l[i]
        for j = i + 1 to n
            spaces_at_the_end[i, j] = spaces_at_the_end[i][j - 1] - l[j] - 1
    for i = 1 to n
        for j = i to n
            if spaces_at_the_end[i][j] < 0
                line_cost[i][j] = inf
            else if j == n and spaces_at_the_end[i][j] >= 0
                line_cost[i][j] = 0
            else line_cost[i][j] = spaces_at_the_end[i][j]^3
    c[0] = 0
    p = new [n]
    for j = 1 to n
        c[j] = inf
        for i = 1 to j
            if c[i - 1] + line_cost[i][j] < c[j]
                c[j] = c[i - 1] + line_cost[i][j]
                p[j] = i
    return c and p
```

   We can see that the complexity is simply $O(n^2)$

4. Provide a comparative study on an investigation over algorithm-design strategies: Divide and Conquer, Dynamic Programming and Greedy Algorithms. State their definition, characteristics, advantages, disadvantages, examples of problems where each strategy is best applied, and a description to characterize problems within each technique. Add references to your work.