CSCI-GA.1170-001/002 Fundamental Algorithms	November 19, 2015

Solutions to Problem 4 of Homework 8 (8 (+3) points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, November 17*

Let $\star : \{1, \ldots, k\} \times \{1, \ldots, k\} \mapsto \{1, \ldots, k\}$ be a binary operation. Below we assume the values of $a \star b$ for $a, b \in \{1, \ldots, k\}$ are stored in some $k \times k$ array M such that $M[a][b] = a \star b$. Consider the problem of examining a string $x = x_1 x_2 \ldots x_n$, where each $x_i \in \{1, \ldots, k\}$, and deciding whether or not it is possible to parenthesize the expression $x_1 \star x_2 \star \ldots \star x_n$ in such a way that the value of the resulting expression is a given target element $t \in \{1, \ldots, k\}$. Notice, the multiplication table is neither commutative or associative, so the order of multiplication matters (and, hence, the result of the expression is not even well defined unless a complete "parenthesization" is specified). For example, consider the following multiplication table and the string $x = 2221$.

Table 1: Multiplication table

	1	2	3
1	1	3	3
2	1	1	2
3	3	3	3

Parenthesizing it $(2 \star 2) \star (2 \star 1)$ gives $t = 1$, but $((2 \star 2) \star 2) \star 1$ gives $t = 3$. On the other hand, no possible parenthesization gives $t = 2$ (you may check this).

(a) (8 points) Assume you are given as input the following: n, k, t, $x[1 \ldots n]$ and M. Give a dynamic programming algorithm that runs in time polynomial in n and k and outputs YES if there exists a paranthesization for x that results in the product equal to t, and NO otherwise. For instance, in the above example with $x = 2221$, the answer is YES if $t = 1$ or $t = 3$, but NO if $t = 2$.

Solution:

Define a 3d-array R and initialize it to 0 such that $R[i, j, k]$ denotes the number of paranthesizations of $x_i \star x_{i+1} \star \ldots \star x_j$ such that the product evaluates to k.

Therefore, the base case is $R[i, i, x_i] = 1$ i.e the number of paranthesizations of x_i such that the product evaluates to x_i is 1

Otherwise, if $i < j$, let $p \in \{1, \ldots, k\}$ and let $S_p = \{(\alpha_y^p, \beta_y^p)\}$ where $1 \leq \alpha_y^p \leq k$, $1 \leq \beta_y^p \leq k$ and $|S_p| \leq k$ be the set of all tuples such that $\alpha_y^p \star \beta_y^p = p$.

Then for any given expression $(x_i \star x_{i+1} \star \ldots \star x_j)$, we can divide it into two partitions $(x_i \star \ldots \star x_t)$ and $(x_{t+1} \star \ldots \star x_j)$ where $i \leq t < j$.

If $x_i \star x_{i+1} \star \ldots \star x_j = p \implies (x_i \star \ldots \star x_t) = \alpha_y^p$ and $(x_{t+1} \star \ldots \star x_j) = \beta_y^p$.

The number of ways in which α_y^p can be obtained from $(x_i \star \ldots \star x_t)$ using different paranthesizations is $R[i, t, \alpha_y^p]$.

Similarly, the number of ways in which β_y^p can be obtained from $(x_{t+1}\star\ldots\star x_j)$ using different paranthesizations is $R[t+1, j, \beta_y^p]$

Note that t ranges from i to $j-1$ and y ranges from 1 to $|S_p|$. Therefore, the total number of ways to obtain p from $(x_i \star x_{i+1} \star \ldots \star x_j)$ using different paranthesizations is

$$R[i, j, p] = \sum_{y=1}^{|S_p|} \sum_{t=i}^{j-1} R[i, t, \alpha_y^p] R[t+1, j, \beta_y^p]$$

Using the above recurrence equation and the base cases fill all the entries of R and at then end the answer is YES if $R[1, n, t] > 0$ and NO otherwise

□

(b) (3 points (**Extra credit**)) Analyze the running time of your algorithm.

Solution:

From the above recurrence relation, it is clear that evaluating $R[i, j, p]$ i.e one entry of the 3d matrix R takes $(j-i)|S_p|$ steps. Now we have k number of 2d arrays of size $n \times n$. Evaluating some p^{th} matrix out of these k number of 2d array takes time $O(n^3)|S_p|$ time where $1 \leq p \leq k$

Therefore total time taken to calculate all the k number of 2d arrays i.e a 3d array R of size $n \times n \times k$ will be $O(n^3)(|S_1| + \ldots + |S_k|)$. Time taken to constructs the sets $S_1, \ldots, S_k$ is $O(k^2)$ as we need to traverse the multiplication table of size k^2 and therefore $|S_1| + \ldots + |S_k| = k^2$.

Therefore the running time of the algorithm is $O(n^3 k^2)$ □