Algorithm Analysis and Design (CS1.301)

Monsoon 2021, IIIT Hyderabad
25 September, Saturday (Lecture 11)

Dynamic Programming (contd.)

Chain Matrix Multiplication

Given a set of matrices $A_1, \dots, A_n$ to be multiplied (in that order), we must find the optimal way to parenthesise them (assuming that multiplying a $m \times n$ with an $n \times p$ matrix takes mnp multiplications).

We can consider a pairing as a binary tree whose leaves are the original matrices, which share a parent node when multiplied. Each node is associated with the weight of multiplying its children.

Let $C(i, j)$ be the minimum cost of computing $\prod_{k=i}^{j} A_k$. We can then say that $C(i, i) = 0$, and that

$$C(i, j) = \min_{i \leq k < j} \{C(i, k) + C(k + 1, j) + m_{i-1} \cdot m_k \cdot m_j\},$$

where A_i has the dimensions $A_{i-1} $\times A_i$.

Each entry of the table takes $O(n)$ time, which means that the overall running time is $O(n^3)$.

The Knapsack Problem

Given a bag of capacity W units, and n items of values v_i and weights w_i, we need the most valuable combintion of items that can be contained in the bag.

With Repetition

Let $K(w)$ be the maximum achievable value with a knapsack of capacity w. Clearly,

$$K(w) = \max_{w_i \leq w} \{K(w - w_i) + v_i\}.$$

Without Repetition

Let us redefine $K(w, j)$ as being the maximum value achievable with a knapsack of capacity w with items 1 to j. Then, $K(w, j) = \max\{K(w - w_j, j - 1) + v_j, K(w, j - 1)\}$.$

For this algorithm we need a table of size $O(nW)$, whose fillng time is constant.