CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 1: Basics of algorithms

Steps for Dynamic Programming

Dynamic programming approach has a template.

Step 1 Figure out the types of sub-problems.

Step 2 Define a recursive procedure.

Step 3 Decide on the memoization strategy.

Step 4 Check that the sub-problem dependencies are acyclic.

Step 5 Analyse the time complexity using the recursion.

Fibonacci and Weighted Interval Scheduling

Fibonacci computation

Step 1 Figure out the types of sub-problems.

Step 2 Define a recursive procedure.

Step 3 Decide on the memoization strategy.

Step 4 Check that the sub-problem dependencies are acyclic.

Step 5 Analyse the time complexity using the recursion.

`Fib(k)`

We got the sub-problems from the recursive definition.

Memoization was done to avoid repeated computation of sub-problems.

If $i < j$ then `Fib`(i) never called `Fib`(j).

Time = Time/sub-problem $\times$ # sub-problem

$O(n)$.

Fibonacci and Weighted Interval Scheduling

Fibonacci computation

Step 1 Figure out the types of sub-problems.	`WtIntSc(n)` Some clever reasoning involved in figuring out the sub-problems.
Step 2 Define a recursive procedure.	From the sub-problems the recurrence was easy.
Step 3 Decide on the memoization strategy.	Maintain the table for the opt values of the sub-problems.
Step 4 Check that the sub-problem dependencies are acyclic.	If $i < j$ then the subroutine for the sub-problem $\{1, \ldots, i\}$ does not make calls to the sub-problem $\{1, \ldots, j\}$.
Step 5 Analyse the time complexity using the recursion.	Overall time analyses as in the case of Fibonacci numbers.

Problems on strings/sequences

Many interesting problems on strings/sequences.

- Parenthesization Figure out the paranthesization of an expression that minimises the overall cost of evaluating the expression.
- Longest increasing subsequence Given a string of positive numbers, find the longest subsequence that is increasing.
- Segmentation Given a long string of letters, find a way (if one exists) of segmenting the string into chunks such that the segmented string is a statement in English.
- Longest common subsequence Given two string of letters, find the longest subsequence that is common to both.
- Edit distance Given two strings x, y, find the smallest number of updates need to convert x into y.

Problems on strings/sequences

Many interest problems on strings/sequences and applying the DP approach

Step 1 Figure out the types of sub-problems.
- ★ Usually suff$[1, i]$, pre$[i, n]$ or substring$[i, j]$.

Step 2 Define a recursive procedure.

Usually the clever part of developing a DP.

Depends heavily on the problem.

Step 3 Decide on the memoization strategy.

Store the values of the sub-problems.

Step 4 Check that the sub-problem dependencies are acyclic.

If Step 2 is defined correctly, this is again not hard to argue.

Step 5 Analyse the time complexity using the recursion.

Use the appropriate methods to analyse recursion.

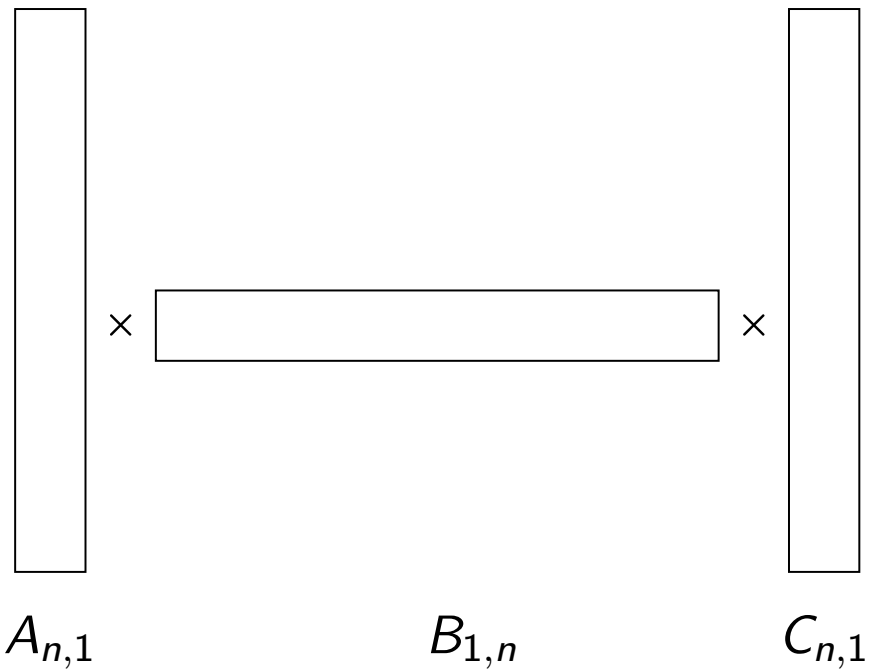
Parenthesization problem

Problem description

Input: Given a string of matrices

Output: Find the minimum cost parenthesization to multiply the matrices

Example



$(A \times B) \times C$. Costs $O(n^2)$ operations.

$A \times (B \times C)$. Costs $O(n)$ operations.

Find the best parenthesization.