Rod-cutting problem

optimize

Rod of n inches

length	1	2	----	n
	p_1	p_2		p_n

maximize total revenue.

Define

v_n := max revenue for an inch rod.

$r_n := \max(p_n, r_1+r_{n-1}, r_2+r_{n-2}, \ldots, r_{n-1}+r_1)$ ← i integer

Array R[1..n]

// put r_i into R[i] ← goal

For i := 1 to n do

// compute r_i and put it in the table

// r_j already in the table for all $j<i$

max := p_i; // worst: i inch rod [r_i]

C[i] := 0 For j := 1 to i-1 do

if R[j]+R[i-j] > max

C[i] := j then max := R[j]+R[i-j]

//end for j

R[i] := max

//end for i

Return R[n]

C[1..n] ← keep track of the actual cut

For i = 1, ..., n

$C[i] = \begin{cases} \text{some } j \text{ with } j \in \{1,\ldots,i-1\} \text{ giving optimal cut of rod of length } i \\ 0 \text{ if worst rod is optimal} \end{cases}$

Run time $\leq \Theta(n^2)$

i	1	2	3
p_i	3	2	8

R[1] = 3

R[2] = max(2, 6) = 6

R[3] = max(8, 3+6, 6+3) = 9

Matrix chain order

A, B, C, ...

m × n

rows columns

AB = C

m × p, p × q

well-defined iff n = p

result is an m × q matrix

$AB \neq BA$

$(AB)C = A(BC)$

Time to multiply an m × n matrix by an n × p matrix = mnp (scalar multiplications)

ABC

A is 10 × 10

B is 10 × 15

C = 15 × 2

10 × 15, 15 × 2

AB C

1500, 300

total time = 1500 + 300 = 1800

10 × 10, 10 × 2

A BC

300, 200

time = 300 + 200 = 500

Given matrices

$A_1, \ldots, A_n$

where A_i is $p_{i-1} \times p_i$

Determine the optimal (wrt. # of multiplications) association order of multiplication.

Input is just $\langle p_0, p_1, p_2, \ldots, p_n \rangle$ (positive integers) (A_1, A_2, A_n)

Output is optimal # of multiplications and matrices with these dimensions.

Oct 13-12:29 PM

$A_1, \ldots, A_n$

For $1 \le i \le j \le n$, define

$A_{i..j} = A_i \cdots A_j$

$= A_i A_{i+1} A_{i+2} \cdots A_j$

$A_{i..j}$ has dimensions $p_{i-1} \times p_j$

NumMults($\langle p_0, \ldots, p_n \rangle$)
// p_i are all positive integers.
// return optimal # of multiplications
// for a product of matrices
// with these dimensions
// (as explained above)
// ~~Idea:~~ assuming last
// mult is $A_{1..i} A_{i+1..n}$ (optimal time for $A_{1..i}$)
// NumMults($\langle p_0, \ldots, p_i \rangle$)
// + NumMults($\langle p_i, \ldots, p_n \rangle$) (optimal time for $A_{i+1..n}$)
// + $p_0 p_i p_n$ (time of final mult.)

NumMults($\langle p_0, \ldots, p_n \rangle$)

$= \min_{i=1}^{n-1} \left(\text{NumMults}(\langle p_0, \ldots, p_i \rangle) + \text{NumMults}(\langle p_i, \ldots, p_n \rangle) + p_0 p_i p_n \right)$

Base case: NumMults($\langle p_0, p_1 \rangle$) = 0

Possible inputs to NumMults are sublists of $\langle p_0, \ldots, p_n \rangle$

Dynamic programming solution:
Build a table

$m[1..n, 1..n]$

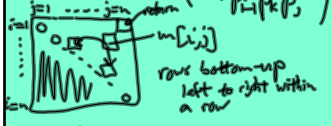
where $m[i,j]$ will hold the optimal cost of computing $A_{i..j}$.

$m[1,n]$ is the final result.

$m[i,i] = 0$ for all $1 \le i \le n$

For $i < j$,

$m[i,j] = \min_{k=i}^{j-1} \left(m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \right)$ ($A_{i..k}$, $A_{k+1..j}$)



Example: $A_1 A_2 A_3 A_4$

$5 = p_0, p_1, p_2, p_3, p_4$
2 3 1 6

0+18+5·2·6
30+18+5·3·6
16+0+5·1·6

30+0+5·3·1 = 45 $(A_1 A_2) A_3$
0+6+5·2·1 = 16 $A_1 (A_2 A_3)$
0+18+2·3·6 = 54
6+0+2·1·6 = 18

Oct 13-1:12 PM