CMPT307: Dynamic Programming

Week 7-2

Xian Qiu

Simon Fraser University

xianq@sfu.ca

Matrix-Chain Multiplication

to multiply a matrix chain A_1, A_2, A_3, where

$$A_1 \in \mathbb{R}^{10\times 100}, \quad A_2 \in \mathbb{R}^{100\times 5}, \quad A_3 \in \mathbb{R}^{5\times 50}$$

$((A_1A_2)A_3) : 10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7500$ multiplications

$(A_1(A_2A_3)) : 100 \cdot 5 \cdot 50 + 10 \cdot 100 \cdot 50 = 75000$ multiplications

full parenthesization:

- single matrix, *e.g.* A, or
- product of two fully parenthesized matrix products $((*)(*))$

Matrix-chain multiplication problem

- input: matrices $A_1, \ldots, A_n$, where $A_i \in \mathbb{R}^{p_{i-1}\times p_i}$
- goal: fully parenthesize $A_1 \cdots A_n$, minimizing # scaler multiplications

Optimal Substructure

- consider $A_i A_{i+1} \cdots A_j$ and an optimal solution OPT
- OPT must split the product at somewhere, say

$$\underbrace{(A_i, \ldots, A_k)}_{S_1}\underbrace{(A_{k+1}, \ldots, A_j)}_{S_2}$$

OPT_1 OPT_2

- to show: OPT_1, OPT_2 are optimal for S_1, S_2 respectively
- cut-and-paste: if there is a better solution OPT_1' for S_1, then we replace OPT_1 by OPT_1' in OPT
- define $m[i, j] =$ objective for $A_{i..j}$

$$m[i, j] = m[i, k] + m[k + 1, j] + p_i p_k p_j$$

Recursive Formula

recursive formula

$$m[i,j] = \begin{cases} 0, & i = j \\ \min_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & i \neq j \end{cases}$$

opt $= m[1,n]$

running time $O(n^3)$

- ▷ width: $O(n)$
- ▷ depth: # subproblems $= \#m[i,j] = \Theta(n^2)$

constructing optimal solution

$$s[i,j] = \text{optimal } k \text{ of } m[i..j]$$

Pseudocode

MATRIX-CHAIN-ORDER(p)

```
1  n = p.length − 1;                                   // p_0, p_1, ..., p_n
2  initialize tables m[1..n, 1..n] and s[1..n − 1, 2..n];
3  m[i, i] = 0 for i = 1, ..., n;                      // single matrix chain
   // l is chain length
4  for l = 2 to n do
5      for i = 1 to n − l + 1 do
6          j = i + l − 1;                              // A[i..j] has length exactly l
           // compute m[i, j]
7          m[i, j] = ∞;
8          for k = i to j − 1 do
9              q = m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j;
10             if q < m[i, j] then
11                 m[i, j] = q;
12                 s[i, j] = k;
13 return m and s;
```

Example

$A_1 : 30 \times 35 \quad A_2 : 35 \times 15 \quad A_3 : 15 \times 5$

$$m[i,j] = \min_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1} p_k p_j\}$$

$$m[1,3] = \min \left\{ \begin{array}{l} m[1,1] + m[2,3] + 30 \times 35 \times 5, \\ m[1,2] + m[3,3] + 30 \times 15 \times 5 \end{array} \right\}$$

$$= \min \{m[2,3] + 5250,\ m[1,2] + 2250\}$$

$$m[2,3] = \min \{m[2,2] + m[3,3] + 35 \times 15 \times 5\} = 2625 \qquad s[2,3] = 2$$

$$m[1,2] = \min \{m[1,1] + m[2,2] + 30 \times 35 \times 15\} = 15750 \qquad s[1,2] = 1$$

$$m[1,3] = \min \{2625 + 5250,\ 15750 + 2250\} = 7875 \qquad s[1,3] = 1$$

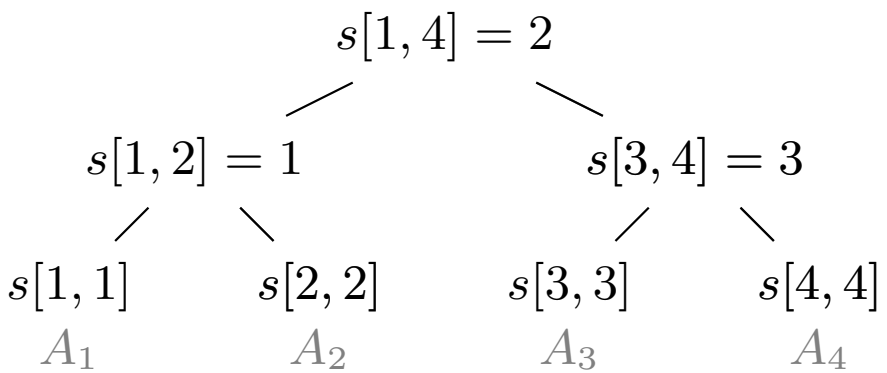
$(A_1(A_2A_3))$

Printing Optimal Solution

PRINT-OPTIMAL-PARENS(s, i, j)

```
1 if i == j then print "A_i";
2 else
3     print "(";
4     PRINT-OPTIMAL-PARENS(s, i, s[i, j]);
5     PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j);
6     print ")";
```

output: $((A_1A_2)(A_3A_4))$



Deriving Optimal Substrcture

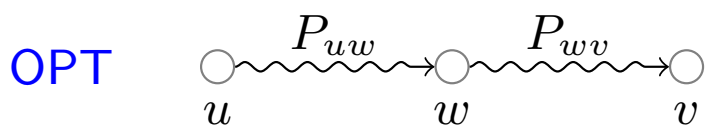
General idea

- consider OPT and one of its step, thereby yielding subproblems
- show that the solutions within OPT w.r.t. subproblems are optimal

Unweighted shortest path (USP)

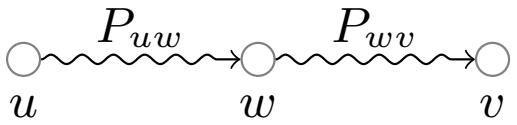
Given undirected graph $G = (V, E)$ and $u, v \in V (u \neq v)$, find a simple u-v path containing as few edges as possible.

optimal substructure of USP



Deriving Optimal Substrcture

Unweighted longest path (ULP): replace few by many

OPT 

- ▷ assume P'_{uw} longest (u, w)-path (and P_{uw} is not)
- ▷ replacing P_{uw} by P'_{uw} in OPT yields a better solution?
- ▷ not true!
- ▷ the new "solution" may be infeasible! (*i.e.* not simple path)

OPT