Algorithm Design & Analysis (CSE222)

Lecture-9

Recap

- Collection of non-adjacent heavy set.

- Vertex cover for Tree

Outline

- Matrix Chain multiplication

- Longest common subsequence

Matrix multiplication

$$\begin{bmatrix}1 & 1 & 1\\ 3 & 2 & 1\end{bmatrix} \times \begin{bmatrix}-1 & 2\\ 2 & 2\\ 1 & -3\end{bmatrix} = \begin{bmatrix}2 & 1\\ 2 & 7\end{bmatrix}$$

$\textsc{MatrixProduct}(A, B)$

Initialize $p \leftarrow \#$ rows in A
Initialize $q \leftarrow \#$ columns in A
Initialize $r \leftarrow \#$ columns in B
Initialize $C \leftarrow \{0\}^{p \times r}$
for $i = 1 \cdots p$
    for $j = 1 \cdots r$
        for $k = 1 \cdots q$
            $C[i, j] = C[i, j] + A[i, k] \cdot B[k, j]$
return C

#rows in the first matrix must be equal to the #columns in the second matrix.

Running time: O(pqr)

Matrix multiplication

Let A, B and C be three matrices of dimensions 5 x 3, 3 x 8 and 8 x 2 respectively.

Compute $M = A{\cdot}B{\cdot}C$.

Let $T = A{\cdot}B$, then $M = T{\cdot}C = (A{\cdot}B){\cdot}C$
Running time:

Let $T = B{\cdot}C$, then $M = A{\cdot}T = A{\cdot}(B{\cdot}C)$
Running time:

Matrix Chain Multiplication

Input: Sequence of matrices $A_1, A_2, \ldots, A_n$
Problem: Running time of fastest matrix chain multiplication.

Trial-1: Check all possible parenthesization of input.
Let P(k) be the all possible parenthesization of an input of k matrices.

$$P(n) = \begin{cases} 1 & \text{if } n = 1 , \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 . \end{cases}$$

Check which is the most efficient.

$$|P(n)| = \frac{\binom{2n}{n}}{n+1} = \Omega(2^n)$$

Matrix Chain Multiplication

Input: Sequence of matrices $A_1 \in R^{p0xp1}$, $A_2 \in R^{p1xp2}$, $A_3 \in R^{p2xp3}$, ..., $A_n \in R^{pn-1xpn}$.

Problem: Running time of fastest matrix chain multiplication.

To optimally multiply $A_i \cdot A_{i+1} \cdot \ldots \cdot A_j$.

- Parenthesize $A_i \cdot A_{i+1} \cdot \ldots \cdot A_j$, let the split is between A_k and A_{k+1}.
- Compute optimal parenthesization of $(A_i \cdot A_{i+1} \cdot \ldots \cdot A_k)$ & $(A_{k+1} \cdot A_{k+2} \cdot \ldots \cdot A_j)$.
- Subproblem: M[i, j] is the minimum number of scalar multiplication needed to compute $A_i \cdot A_{i+1} \cdot \ldots \cdot A_j$. For $A_1 \cdot A_2 \cdot \ldots \cdot A_n$ we look at m[1, n].
  - When i == j then it is trivial!
  - If optimal split between A_k and A_{k+1} then $m[i,j] = m[i,k] + m[k+1,j] + p_{i-1}p_k p_j$

$$m[i,j] = \begin{cases} 0 & \text{if } i = j \ , \\ \min\limits_{i \le k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{if } i < j \ . \end{cases}$$

Pseudocode

MATRIX-CHAIN-ORDER(p)

```
n = p.length - 1
let m[1..n, 1..n] and s[1..n-1, 2..n] be new tables
for i = 1 to n
    m[i, i] = 0
for l = 2 to n              // l is the chain length
    for i = 1 to n - l + 1
        j = i + l - 1
        m[i, j] = ∞
        for k = i to j - 1
            q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
            if q < m[i, j]
                m[i, j] = q
                s[i, j] = k
return m and s
```

Running time?

$O(n^3)$

Example

Input: Sequence of matrices $A_1 \in R^{2x3}$, $A_2 \in R^{3x6}$, $A_3 \in R^{6x2}$, $A_4 \in R^{2x9}$, $A_5 \in R^{9x3}$.
Problem: Running time of fastest matrix chain multiplication.

Maintain following tables to solve the problem (bottom up).

	1	2	3	4	5
1					
2					
3					
4					
5					

Maintain multiplication cost for a subset of matrices

	1	2	3	4	5
1					
2					
3					
4					
5					

Maintain multiplication order

Table Filling (or Bottom Up)

Start with simplest subproblem.

Gradually reach to the solution of original problem.

Does not require recursive function call.

Solves subproblems which might not even necessary for solving the original problem.

Recursion

```
RECURSIVE-MATRIX-CHAIN(p, i, j)
if i == j
    return 0
m[i, j] = ∞
for k = i to j − 1
    q = RECURSIVE-MATRIX-CHAIN(p, i, k)
          + RECURSIVE-MATRIX-CHAIN(p, k + 1, j)
          + p_{i−1} p_k p_j
    if q < m[i, j]
        m[i, j] = q
return m[i, j]
```

Recurrence:

$$
\begin{aligned}
T(1) &\geq 1 , \\
T(n) &\geq 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1) \qquad \text{for } n > 1
\end{aligned}
$$

Recursive running time

$$
\begin{aligned}
T(1) &\geq 1\,, \\
T(n) &\geq 1 + \sum_{k=1}^{n-1} \left(T(k) + T(n-k) + 1\right) \qquad \text{for } n > 1
\end{aligned}
$$

Memoization (or Top Down)

```
MEMOIZED-MATRIX-CHAIN(p)
n = p.length − 1
let m[1..n, 1..n] be a new table
for i = 1 to n
    for j = i to n
        m[i, j] = ∞
return LOOKUP-CHAIN(m, p, 1, n)
```

```
LOOKUP-CHAIN(m, p, i, j)
if m[i, j] < ∞
    return m[i, j]
if i == j
    m[i, j] = 0
else for k = i to j − 1
        q = LOOKUP-CHAIN(m, p, i, k)
            + LOOKUP-CHAIN(m, p, k + 1, j) + p_{i−1} p_k p_j
        if q < m[i, j]
            m[i, j] = q
return m[i, j]
```

Reference

Slides

Introduction to Algorithms by CLRS - Chp-15.2 & 15.3