Algorithmic Strategies 2024/25

Week 5 – Dynamic Programming

· U C ·

UNIVERSIDADE DE COIMBRA

Matrix-chain multiplication

- Given a sequence of matrices, find the most efficient way to multiply these matrices together.
- The number of operations depends of the order of multiplication.

Example: Let A_1 be a 20×40 matrix, A_2 be a 40×10 matrix and A_3 be a 10×70 matrix.

$(A_1 A_2) A_3$ has $(20 \times 40 \times 10) + (20 \times 10 \times 70) = 22000$ operations

$A_1 (A_2 A_3)$ has $(40 \times 10 \times 70) + (20 \times 40 \times 70) = 84000$ operations

The problem consists of finding the optimal parenthesisation!

Matrix-chain multiplication

- Given n matrizes $A_1 A_2 \dots A_n$, where A_i has size $p_{i-1} p_i$, assume that the optimal parenthesisation is known. Then, it must split the product at some position k:

$$(A_1 \dots A_k)(A_{k+1} \dots A_n)$$

- Then, the parenthesisation of both $(A_1 \dots A_k)$ and $(A_{k+1} \dots A_n)$ must also be optimal: optimal substructure!

- Proof by contradiction: Assume that parenthesisation above of $(A_1 \dots, A_k)$ takes x operations, but it is still possible to find another with $x' < x$. Then, it is also possible to improve the optimal parenthesisation, which is a contradiction!

Matrix-chain multiplication

A recursive solution:

- Let $mult(i, j)$ be the minimum number of scalar multiplications needed to compute $A_i A_{i+1} \dots A_j$.
  (A_i has size $p_{i-1} \times p_i$)
- Base case: $mult(i, i) = 0$ for all $i = 1, 2, ..., n$
- Recursion: $mult(i, j) = mult(i, k) + mult(k + 1, j) + p_{i-1} p_k p_j$.
- As k is not known, must compute minimum over $i \leq k < j$.

Matrix-chain multiplication

A simple recursive solution:

```
Function mult(i, j)
  if j ≤ i then                                        {base case}
    return 0
  cost = ∞
  for k = i to j − 1 do
    cost = min(cost, mult(i, k) + mult(k + 1, j) + p_{i−1} p_k p_j)
  return cost
```

- This solution takes exponential time.

Matrix-chain multiplication

Top-down approach:

Function $mult(i, j)$

  if $j \leq i$ **then** {base case}

    return 0

  if $M[i, j] \geq 0$ **then**

    return $M[i, j]$

  $M[i, j] = \infty$

  for $k = i$ **to** $j - 1$ **do**

    $M[i, j] = \min(M[i, j], mult(i, k) + mult(k + 1, j) + p_{i-1} p_k p_j)$

  return $M[i, j]$

Matrix-chain multiplication

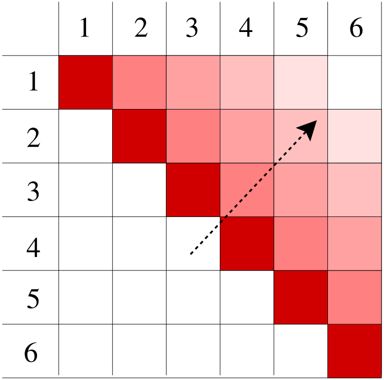
Bottom-up approach:

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

$M[i,j] = \infty$
for $k = i$ **to** $j-1$ **do**
    $M[i,j] = \min(M[i,j], \mathit{mult}(i,k) + \mathit{mult}(k+1,j) + p_{i-1}p_k p_j)$

Matrix-chain multiplication

Bottom-up approach:



Matrix-chain multiplication

Bottom-up approach:

Function $mc(n)$
  for $i = 1$ **to** n **do**
    $M[i, i] = 0$
  for $d = 2$ **to** n **do**
    for $i = 1$ **to** $n - d + 1$ **do**
      $j = i + d - 1$
      $M[i, j] = \infty$
      for $k = i$ **to** $j - 1$ **do**
        $M[i, j] = \min(M[i, j], M[i, k] + M[k + 1, j] + p_{i-1}p_k p_j)$
  return $M[1, n]$

- Sweeps the array M diagonally
- It has $O(n^3)$ time complexity