Home Exercise 5: Dynamic Programming

Algorithms and Complexity lecture
at CentraleSupelec/ESSEC

Dimo Brockhoff
firstname.lastname@inria.fr

due: Friday, October 25, 2019 at 11:59:59pm Paris time

Abstract

Please send your solutions by email to Dimo Brockhoff (preferably in PDF format) with a clear indication of your full name until the submission deadline on October 25, 2019 (a Friday!). Groups of up to 4 students are explicitly allowed and even encouraged. In the case of group submissions, please make sure that you submit maximally four times with the same partner!

1 Greedy Algorithm vs. Dynamic Programming (5 points)

Decide for each of the following statements, whether "a greedy algorithm" or "a dynamic programming approach" can replace the variables "X" and "Y" to make the statement correct.

1. In "X", we make at each step a decision considering the current problem and the solution(s) to previously solved sub-problem(s).
2. It is guaranteed that "X" will generate an optimal solution as it generally considers all possible cases and then choose the best.
3. "X" follows the problem solving heuristic of making the locally optimal choice at each stage.

4. A problem should possess the property of overlapping subproblems to make "X" an efficient alternative.

5. "X" is more efficient in terms of memory than "Y" as it never looks back or revises previous choices.

2 Matrix Chain Multiplications (15 points)

Consider the multiplication of n matrices $A_1 \cdot A_2 \cdots A_n$ where matrix A_i is an a_i-by-b_i matrix. The problem is relevant in several applications areas such as 3D-graphics, physics, machine learning, or mathematical finance. Here, we are less interested in the actual final number of the multiplication, but rather in *h*ow we calculate the result.

We know that matrix multiplication is not commutative ($A \cdot B \neq B \cdot A$ in general), but it is associative, i.e., $A \cdot (B \cdot C) = (A \cdot B) \cdot C$. By deciding in which order the multiplications are computed, we can potentially save a lot of computational effort in terms of the number of needed basic multiplications. For simplicity, we assume that the multiplication of a p-by-q matrix with a q-by-r matrix costs pqr many *basic* multiplications (when using the naive matrix multiplication we all know from basic algebra and exercise 1).

If we had for example to multiply a 10-by-1 matrix A with a 1-by-30 matrix B and a 30-by-2 matrix C, the computation as $(A \cdot B) \cdot C$ needs $10 \cdot 1 \cdot 30 + 10 \cdot 30 \cdot 2 = 900$ multiplications while the computation as $A \cdot (B \cdot C)$ needs only $1 \cdot 30 \cdot 2 + 10 \cdot 1 \cdot 2 = 80$ multiplications!

The problem, we consider in the following, is to compute the optimal order (placement of brackets) for the multiplication of n matrices that minimizes the total number of necessary multiplications.

1. Which conditions on the a_i and b_i are needed to have a valid multiplication? [1 point]

2. Give an example where the greedy approach, which chooses the cheapest available multiplication in each step, does not find the optimal bracketing. [2 points]

The number of all possible orders of multiplications is exponential in n and, thus, an enumeration/brute force approach will not be feasible. We consider dynamic programming instead here. Let $C(i, j)$ be the optimal cost (in number of basic multiplications) to compute $A_i \cdot A_{i+1} \cdots A_j$.

3. Which values of $C(i,j)$ are easy to compute ("initialization of the dynamic programming")? [2 points]

4. Which value of $C(i,j)$ corresponds to the optimal solution (the cost of the entire matrix chain multiplication)? [1 point]

Consider now that the corresponding solution for the subproblem $C(i,j)$ is to have calculate first $A_i \cdots A_k$, then $A_{k+1} \cdots A_j$ and finally multiply the corresponding matrices ($A_i \cdot A_{i+1} \cdots A_j$ is computed as $(A_i ? \ldots ? A_k) \cdot (A_{k+1} ? \ldots ? A_j)$).

5. In this case, write how to compute $C(i,j)$. [3 points]

6. Write $C(i,j)$ in the more general case where the splitting point k is not known. [3 points]

7. Consider the example of five matrices A_1 (5-by-2), A_2 (2-by-10), A_3 (10-by-1), A_4 (1-by-10), and A_5 (10-by-2) and complete a table like the following one with the values of $C(i,j)$ as the dynamic programming approach would do. What are the actual minimum number of basic multiplications needed? [3 points]

i/j	1	2	3	4	5
1					
2	-				
3	-	-			
4	-	-	-		
5	-	-	-	-	