Computer Science >

Algorithms: Dynamic Programming - Matrix Chain Multiplication with C Program Source Code

Quick Links to Various Sections on the Main Page

The Learning Point
Computer Science
Algorithms: An introduction to Dynamic Programming
CS - Data Structures and Algorithms
CS - Programming Interviews
CS - Miscellaneous C Programs
CS - Intro to DB Queries
Electrical Science and Engineering
Electrical Sci - DC Circuits
Electrical Sci - Digital Electronics
Mathematics
Mathematics - Linear Algebra
Mathematics - Geometry
Mathematics - Single Variable Calculus
Mathematics - Game Theory
Mathematics - Operations Research
Physics
Physics - Basic Mechanics
Physics - Electrostatics
Sitemap
Recent site activity

These are Quick Links to sections of the Main Page

0

To go through the C program / source-code, scroll down to the end of this page

Matrix Chain Multiplication

Dynamic Programming solves problems by combining the solutions to subproblems just like the divide and conquer method. Dynamic programming method is used to solve the problem of multiplication of a chain of matrices so that the fewest total scalar multiplications are performed.

Given a chain (A1, A2, A3, A4….An) of n matrices, we wish to compute the product. A product of matrices is fully parenthesized if it is either a single matrix or the product of fully parenthesized matrix products, surrounded by parenthesis. Since, matrix multiplication is associative all parenthesizations yield the same product. But, the way we parenthesize a chain of matrices have an impact on the cost of evaluating the product. We divide a chain of matrices to be multiplied into two optimal sub-chains, and then the optimal parenthesizations of the sub-chains must be composed of optimal chains.

Algorithm - this has been covered in the tutorial document:

A block of memory cache is used to store the result of the function for specific values. If cache[i][j] = -1 then we do not know the result. If it is some number then that denotes the return value of multiply (i,j). We store it to avoid computing the same again.

Input Format: First integer must be the number of matrices. It has to be followed by rows of first matrix, columns of first matrix, columns for second matrix, columns for third matrix and so on.

Complete Tutorial with Examples :

Matrix Chain Multiplication

Dynamic Programming solves problems by combining the solutions to subproblems just like the divide and conquer method.

Dynamic programming method is used to solve the problem of multiplication of a chain of matrices so that the fewest total scalar multiplications are performed.

Given a chain (A_1, A_2, A_3, A_4....A_n) of n matrices, we wish to compute the product. A product of matrices is fully parenthesized if it is either a single matrix or the product of fully parenthesized matrix products, surrounded by parenthesis. Since, matrix multiplication is associative all parenthesizations yield the same product. But, the way we parenthesize a chain of matrices have an impact on the cost of evaluating the product. We divide a chain of matrices to be multiplied into two optimal sub-chains, and then the optimal parenthesizations of the sub-chains must be composed of optimal chains.

Algorithm:

A block of memory cache is used to store the result of the function for specific values. If cache[i][j] = -1 then we do not know the result. If it is some number then that denotes the return value of multiply (i,j). We store it to avoid computing the same again.

Input Format: First integer must be the number of matrices. It has to be followed by rows of first matrix, columns of first matrix, columns for second matrix, columns for third matrix and so on.

d[i] is used to store the dimension of the matrix i^{th} matrix has dimension d[i-1] * d[i]. So for no loss of generality we will put d[0] to be number of rows of the first matrix and we start the index from 1.

Matrix Chain Multiplication - C Program Source Code

```
#include<string.h>
#include<stdio.h>
#include<limits.h>
int min(int a,int b)
{
        return a < b ? a:b;
}
/*d[i] is used to store the  dimension of the matrix ith matrix has dimension d[i-1] * d[i].
  So for no loss of generality we will put d[0] to be number of rows of the first matrix and we
  start the index from 1
 */
int d[100100];

/* Cache is used to store the result of the function for specific values. If cache[i][j] = -1 then we
   do not know the result. If it is some number then that denotes the return value of multiply(i,j).
   We store it to avoid computing the same again
*/
int cache[1024][1024];
int multiply(int from,int to)
{
        if(from==to)return 0;
        if(cache[from][to]!=-1)
        {
                return cache[from][to];
        }
        int iter,result = INT_MAX;
        /*We put the paranthesis at every possible step and we take the one for which computation
          is minimum */
        for(iter=from;iter<to;iter++)
```

```
        {
                /* Update the result every time */
                result= min(result,multiply(from,iter) + multiply(iter+1,to) + d[from-1]*d[iter]*d[to]);

        }
        return result;
}
/* Input Format: First integer must be the number of matrices. It has to be followed by
   rows of first matrix, columns of first matrix, columns for second matrix, columns for third matrix,...
 */
int main()
{
        /*Initialising cache to -1 */
        memset(cache, -1,sizeof(cache));
        int number_of_matrices;
        scanf("%d",&number_of_matrices);
        scanf("%d",&d[0]);
        int iter;
        for(iter=1;iter<=number_of_matrices;iter++)
        {
                scanf("%d",&d[iter]);
        }
        printf("%d\n",multiply(1,number_of_matrices));

}
```

Related Tutorials (common examples of Dynamic Programming):

Integer Knapsack problem	An elementary problem, often used to introduce the concept of dynamic programming.
Matrix Chain Multiplication	Given a long chain of matrices of various sizes, how do you parenthesize them for the purpose of multiplication - how do you chose which ones to start multiplying first?
Longest Common Subsequence	Given two strings, find the longest common sub sequence between them.

Some Important Data Structures and Algorithms, at a glance:

Arrays : Popular Sorting and Searching Algorithms			
Bubble Sort	Insertion Sort	Selection Sort	Shell Sort
Merge Sort	Quick Sort	Heap Sort	Binary Search Algorithm
Basic Data Structures and Operations on them			
Stacks	Queues	Single Linked List	Double Linked List
Circular Linked List	1.		

Tree Data Structures			
Binary Search Trees	Heaps	Height Balanced Trees	
Graphs and Graph Algorithms			
Depth First Search	Breadth First Search	Minimum Spanning Trees: Kruskal Algorithm	Minumum Spanning Trees: Prim's Algorithm
Dijkstra Algorithm for Shortest Paths	Floyd Warshall Algorithm for Shortest Paths	Bellman Ford Algorithm	
Popular Algorithms in Dynamic Programming			
Dynamic Programming	Integer Knapsack problem	Matrix Chain Multiplication	Longest Common Subsequence
Greedy Algorithms			
Elementary cases : Fractional Knapsack Problem, Task Scheduling	Data Compression using Huffman Trees		