Dynamic Programming | Set 8 (Matrix Chain Multiplication)

Given a sequence of matrices, find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.

We have many options to multiply a chain of matrices because matrix multiplication is associative. In other words, no matter how we parenthesize the product, the result will be the same. For example, if we had four matrices A, B, C, and D, we would have:

```
(ABC)D = (AB)(CD) = A(BCD) = ....
```

However, the order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the efficiency. For example, suppose A is a 10 × 30 matrix,

B is a 30 × 5 matrix, and C is a 5 × 60 matrix. Then,

```
(AB)C = (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations
A(BC) = (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations.
```

Clearly the first parenthesization requires less number of operations.

Given an array p[] which represents the chain of matrices such that the ith matrix Ai is of dimension p[i-1] x p[i]. We need to write a function MatrixChainOrder() that should return the minimum number of multiplications needed to multiply the chain.

```
Input: p[] = {40, 20, 30, 10, 30}
Output: 26000
There are 4 matrices of dimensions 40x20, 20x30, 30x10 and 10x30.
Let the input 4 matrices be A, B, C and D.  The minimum number of
multiplications are obtained by putting parenthesis in following way
(A(BC))D --> 20*30*10 + 40*20*10 + 40*10*30

Input: p[] = {10, 20, 30, 40, 30}
Output: 30000
There are 4 matrices of dimensions 10x20, 20x30, 30x40 and 40x30.
Let the input 4 matrices be A, B, C and D.  The minimum number of
multiplications are obtained by putting parenthesis in following way
((AB)C)D --> 10*20*30 + 10*30*40 + 10*40*30

Input: p[] = {10, 20, 30}
Output: 6000
There are only two matrices of dimensions 10x20 and 20x30. So there
is only one way to multiply the matrices, cost of which is 10*20*30
```

1) Optimal Substructure:
A simple solution is to place parenthesis at all possible places, calculate the cost for each placement and return the minimum value. In a chain of matrices of size n, we can place the first set of parenthesis in n-1 ways. For example, if the given chain is of 4 matrices. let the chain be ABCD, then there are 3 way to place first set of parenthesis: A(BCD), (AB)CD and (ABC)D. So when we place a set of parenthesis, we divide the problem into subproblems of smaller size. Therefore, the problem has optimal substructure property and can be easily solved using recursion.

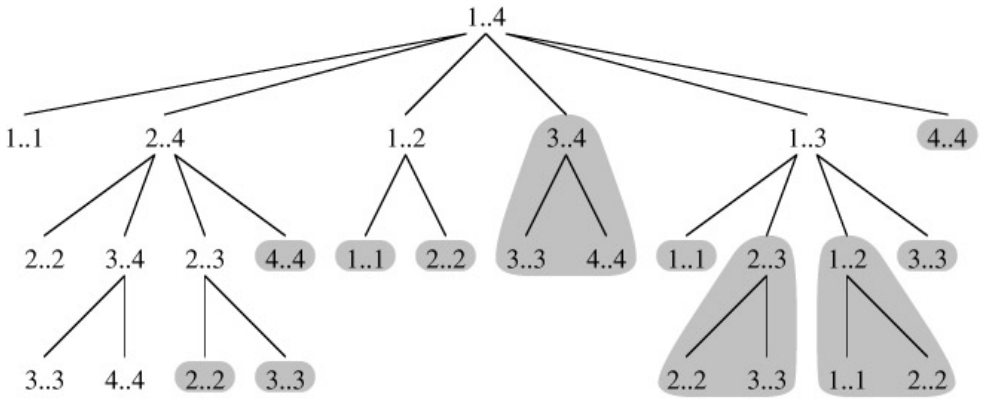
Minimum number of multiplication needed to multiply a chain of size n = Minimum of all n-1 placements (these placements create subproblems of smaller size)

2) Overlapping Subproblems
Following is a recursive implementation that simply follows the above optimal substructure property.

```
/* A naive recursive implementation that simply follows the above optimal
 substructure property */
#include<stdio.h>
#include<limits.h>

// Matrix Ai has dimension p[i-1] x p[i] for i = 1..n
int MatrixChainOrder(int p[], int i, int j)
{
    if(i == j)
        return 0;
    int k;
```

```
    int min = INT_MAX;
    int count;

    // place parenthesis at different places between first and last matrix,
    // recursively calculate count of multiplcations for each parenthesis
    // placement and return the minimum count
    for (k = i; k <j; k++)
    {
        count = MatrixChainOrder(p, i, k) +
                MatrixChainOrder(p, k+1, j) +
                p[i-1]*p[k]*p[j];

        if (count < min)
            min = count;
    }

    // Return minimum count
    return min;
}

// Driver program to test above function
int main()
{
    int arr[] = {1, 2, 3, 4, 3};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("Minimum number of multiplications is %d ",
                           MatrixChainOrder(arr, 1, n-1));

    getchar();
    return 0;
}
```

Time complexity of the above naive recursive approach is exponential. It should be noted that the above function computes the same subproblems again and again. See the following recursion tree for a matrix chain of size 4. The function MatrixChainOrder(p, 3, 4) is called two times. We can see that there are many subproblems being called more than once.



Since same suproblems are called again, this problem has Overlapping Subprolems property. So Matrix Chain Multiplication problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be

avoided by constructing a temporary array m[][] in bottom up manner.

Dynamic Programming Solution
Following is C/C++ implementation for Matrix Chain Multiplication problem using Dynamic Programming.

```
// See the Cormen book for details of the following algorithm
#include<stdio.h>
#include<limits.h>

// Matrix Ai has dimension p[i-1] x p[i] for i = 1..n
int MatrixChainOrder(int p[], int n)
{

    /* For simplicity of the program, one extra row and one extra column are
       allocated in m[][].  0th row and 0th column of m[][] are not used */
    int m[n][n];

    int i, j, k, L, q;

    /* m[i,j] = Minimum number of scalar multiplications needed to compute
       the matrix A[i]A[i+1]...A[j] = A[i..j] where dimention of A[i] is
       p[i-1] x p[i] */

    // cost is zero when multiplying one matrix.
    for (i = 1; i < n; i++)
        m[i][i] = 0;

    // L is chain length.
    for (L=2; L<n; L++)
    {
        for (i=1; i<=n-L+1; i++)
        {
            j = i+L-1;
            m[i][j] = INT_MAX;
            for (k=i; k<=j-1; k++)
            {
                // q = cost/scalar multiplications
                q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
                if (q < m[i][j])
                    m[i][j] = q;
            }
        }
    }

    return m[1][n-1];
}

int main()
{
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr)/sizeof(arr[0]);
```

```
    printf("Minimum number of multiplications is %d ",
                        MatrixChainOrder(arr, size));

    getchar();
    return 0;
}
```

Time Complexity: O(n^3)
Auxiliary Space: O(n^2)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

References:
http://en.wikipedia.org/wiki/Matrix_chain_multiplication
http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Dynamic/chainMatrixMult.htm

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Writing code in comment? Please use **ideone.com** and share the link here.

Mohammed AL Enazi • 2 days ago
Hi

I have this java code for Matrix Chain which print the optimal solution with parenthesis expression. I need your help how to modify my code to print 5 best solutions with their parenthesis instead of one optimal solution

```
import java.io.*;

import java.text.DecimalFormat;

import java.text.NumberFormat;

import java.util.*;

import java.lang.StringBuffer;

class Matrix

{
```

see more

• Reply • Share ›

anonymous • a month ago

even we can do this also. Can someone please tell me time complexity of my solution.

http://ideone.com/Zokrzv

1 • Reply • Share ›

sk anonymous • a month ago

It's O(n^2) .

Explanation:
1st for loop(one inside other) : O(n^2) "Initialization"
2nd for loop (one inside other) : O(n^2) "Computation"
3rd for loop (one inside other) :O(n^2) "printing"
So O(n^2).
Note:Test your algorithm with some test cases .Generate test case and run both algorithm check which one gives correct answer .

• Reply • Share ›

anonymous sk • a month ago

i am not able to understand your explanation.

1st for loop(one inside other)
2nd for loop (one inside other)
3rd for loop (one inside other). there is no any 3rd for loop in my program.

• Reply • Share ›

sk anonymous • a month ago

For printing you have used 3rd for loop correct

For printing you have used 3rd for loop correct ?

• Reply • Share ›

anonymous ➚ sk • a month ago

okey.. i got it. i thought you were saying that i am using 3rd nested loop somewhere that's why.. thank you so much ...

• Reply • Share ›

sk ➚ anonymous • a month ago

Did you check whether your algorithm is correct or not ?
Since may be your algorithm is giving wrong answer for some test cases .

• Reply • Share ›

Mission Peace • 2 months ago

https://www.youtube.com/watch?... Here is my video on explanation of this question.

• Reply • Share ›

Aditya Goel • 3 months ago

The non-optimized program looks confusing on boundaries.
Check out modified program-
http://ideone.com/CnCWV9

• Reply • Share ›

Bhuwnesh Dhakar • 3 months ago

Here is a code with time complexity of O(n). Check it out
http://ideone.com/a1Rmaz

1 • Reply • Share ›

Aditya Goel ➚ Bhuwnesh Dhakar • 3 months ago

This is not Run Length encoding program thread!

• Reply • Share ›

AJ • 3 months ago

The answer for

Input: p[] = {40, 20, 30, 10, 30}

Output: 26000
There are 4 matrices of dimensions 40x20, 20x30, 30x10 and 10x30.
Let the input 4 matrices be A, B, C and D. The minimum number of
multiplications are obtained by putting parenthesis in following way
(A(BC))D --> 20*30*10 + 40*20*10 + 40*10*30

should be 24000 , by order A((BC)D)

should be 24000 ; by order A((BC)D).

• Reply • Share ›

Manu Chandel ➚ AJ • 3 months ago

A(BC)D= {40,20,10,30}+20*30*10={40,20,10,30}+6000
A((BC)D)={40,20,30}+20*10*30+6000={40,20,30}+12000
(A((BC)D))=12000+40*20*30=36000
so minimum is 26000 only

• Reply • Share ›

practofath • 4 months ago

This link explains the solution in a step by step manner.

https://www.cse.ust.hk/~dekai/...

6 • Reply • Share ›

lavish • 4 months ago

for (L=2; L<n; l++)="" it="" should="" be="" :="" for="" (l="2;" l<="n;" l++)="">

• Reply • Share ›

Vinay Pai • 6 months ago

Hi

In the space optimized version of the code above, I was not able to understand why we are using this version of 3 loops instead of using 3 straight loops from 1->n. Any inputs will be appreciated. Thanks!

```
// L is chain length.
for (L=2; L<n; l++)="" {="" for="" (i="1;" i<="n-L+1;" i++)="" {="" j="i+L-1;" m[i]
[j]="INT_MAX;" for="" (k="i;" k<="j-1;" k++)="" {="" <="" code="">
```

• Reply • Share ›

Guest • 6 months ago

Hi,

I am not able to understand why we need to use

// L is chain length.
for (L=2; L<n; l++)="" {="" for="" (i="1;" i<="n-L+1;" i++)="" {="" j="i+L-1;" m[i][j]="INT_MAX;" for="" (k="i;" k<="j-1;" k++)="" {="" instead="" of="" using="" 3="" straight="" loops?="" any="" inputs="" will="" be="" appreciated.="" thanks!="">

1 • Reply • Share ›

guest • 7 months ago

I think following recurrence should be sufficient for Matrix chain multiplications which is O(N^2)

in DP

---

int f(int a[],int l,int r){

if(r-l == 2) {

return a[l]*a[l+1]*a[r];

}

return Min(a[l]*a[r-1]*a[r]+f(a,l,r-1),a[l]*a[l+1]*a[r]+f(a,l+1,r));

}

• Reply • Share ›

Rajmohan • 7 months ago

In the DP Solution, the following line has an error

for (i=1; i<=n-L+1; i++)

It should have been

for (i=1; i<n-l+1; i++)=""">

10 • Reply • Share ›

sfsf Rajmohan • 3 months ago

yes u r right i should be in second number loop for(i=1; i

• Reply • Share ›

darkside • 8 months ago

please reply to this one
if you will implement this code in java arrayindex outof bound error will come so its not a proper logic code
in c array size doesn't create problem if required on run allocate more
space

here n = number of matrices not (number of matrices+1)
when we use n+1 condition should change accordingly

```
for (L=2; L<=n; L++)
{
for (i=1; i<=n-L+1; i++)
{
j = i+L-1;
m[i][j] = Integer.MAX_VALUE;
for (k=i; k<=j-1; k++)
```

```
{
q = m[i][k] + m[k+1][j] + p[i-1]*p[k]*p[j];
if (q < m[i][j])
m[i][j] = q;
}
}
}
```

work fine in java

Reply • Share ›

Guest • 10 months ago

My code with complexity O(n^2)
http://ideone.com/if7iRp

3 Reply • Share ›

Sanchita Tiwari Guest • 7 months ago

your answer for {1, 2, 3, 4} will give 30,but Actual answe is :18
leave about complexity,but first output the correct answer !!

1 Reply • Share ›

danny Guest • 10 months ago

this question cannot be solved in less then O(n^2.78)(Strassen's algorithm), refer to Cormen book for proof. So, your solution is wrong...

4 Reply • Share ›

rb001 • 10 months ago

How is the Auxiliary space of the order O(n^2)?

Reply • Share ›

randomguy202 rb001 • 3 months ago

An extra 2D array of size n x n is used in the dynamic programming approach.

1 Reply • Share ›

Gaurav Gupta • 10 months ago

My approach with complexity of O(n^2);
http://ideone.com/tU8ckI

Reply • Share ›

jaincod • a year ago

Hey Aveek, the given algorithm is having minor problem as inner for loop should be for (i = 1; i <= n-L; i++) the doubt is because of the fact that n is one more than the number of matrices, better given at wikipedia,,, so refer there www.google.co.in/url?sa=t&...

Reply • Share ›

Aveek Biswas • a year ago

In the dynamic programming solution, is there any need for running the loop of "i" from i=1 to i<=n-L+1 ? I think it is unnecessary. for(i=1;i<=n-L;i++) is working fine as well. I am stressing on this as this extra loop unnecessarily adds a (n+1)th column or row in the matrix m[n][n].

3 ^ | v • Reply • Share ›

ASHISH • a year ago

for (i=1; i<=n-L+1; i++).....
it should be for (i=1; i < n-L+1; i++) .....
plz.... check once .....

5 ^ | v • Reply • Share ›

Shubham • a year ago

Why couldn't you just write a program where for loop starts indexing from
i=0 instead of first start from i = 1 then access array element by a[i-1]

^ | v • Reply • Share ›

Guest • a year ago

what is "INT_MAX" used in the above programs ?

^ | v • Reply • Share ›

Tarzan ➚ Guest • a year ago

its a macro defined in limits.h. It is the maximum value integer can take. Corresponding to the same we have INT_MIN.

^ | v • Reply • Share ›

paras_meena • a year ago

//Simple and neat And Same as this tutorial =P

```
#include <bits stdc++.h="">

#define _ ios_base::sync_with_stdio(0);cin.tie(0);

using namespace std;

#define ll long long

int main()

{

int N;

scanf("%d", &N);
```

int dp[N+4][N+4];

int arr[N+4];

see more

1 • Reply • Share ›

its_dark • a year ago

Bottom-Up Dp :

```
int dp[10][10];
int calc(int p[], int b, int e){
    int &x = dp[b][e];
    if(x != -1)
        return x;
    if(b==e) return x=0;
    int min=INT_MAX, ans;
    for (int i = b; i < e; ++i)
    {
        ans = calc(p,b,i) + calc(p,i+1,e) + p[b-1]*p[i]*p[e];
        if(ans < min)min = ans;
    }
    return x=min;
}
```

call with b=0, e=n-1

1 • Reply • Share ›

Ali • a year ago

"n a chain of matrices of size n, we can place the first set of parenthesis in n-1 ways." - this seems to be wrong. It should be nsquare/2.

• Reply • Share ›

Vinodhini • 2 years ago

I guess the following loop is wrong. i < n-L+1 should be correct. Because it starts accessing the index that are out of bounds.

for (i=1; i<=n-L+1; i++)

Please correct me if I am wrong.

11 • Reply • Share ›

ankit ➔ Vinodhini • 8 months ago

yes you are right for L=2,i =n-1, j = i+L-1 j=n while nth index does not exists

• Reply • Share ›

Kartik ➚ Vinodhini • a year ago

The loop doesn't seem to be going out of bound. Note that the value of L varies from 2 to n-1.

• Reply • Share ›

jugal ➚ Vinodhini • a year ago

yes, you r right.

1 • Reply • Share ›

bhumit • 2 years ago

in dp approach the middle for loop should be
for(i=1;i<n-l+1;++i) and="" not="" for(i="1;i<=n-L+1;++i)" because="" j="i+L-1">

1 • Reply • Share ›

piki • 2 years ago

is my program correct which is taking O(n^2) times i think less than O(n^3).....please check

```
  //Actually i want to learn DP
#include<stdio.h>
#include<iostream>
#include<limits.h>

using namespace std;

struct dim
{
  int left;
  int right;
};

int main()
{
  int N,i,j,x,y,z,w,cost1,pp;
```

see more

• Reply • Share ›

piki • 2 years ago

is my program correct which is taking O(n^2) times i think less than O(n^3).....please check

```
 /* //Actually i want to learn DP
#include<stdio.h>
#include<iostream>
```

```
#include<limits.h>

using namespace std;

struct dim
{
  int left;
  int right;
};

int main()
{
  int N,i,j,x,y,z,w,cost1,pp;
```

see more

• Reply • Share ›

piki • 2 years ago

this is correct or not please check this approach is taking O(n^2) time.....

```
 /* //Actually i want to learn DP
#include<stdio.h>
#include<iostream>
#include<limits.h>

using namespace std;

struct dim
{
  int left;
  int right;
};

int main()
{
  int N,i,j,x,y,z,w,cost1,pp;
```

see more

1 • Reply • Share ›

equation • 2 years ago

working code is here.....
source -Cormen book

```
#include<iostream>
#include<conio.h>
using namespace std;

int chain_mul(int p[],int n){

    int m[n+1][n+1];

    for(int i=1;i<=n;i++)
    m[i][i]=0;


    for(int l=2;l<=n;l++){
```

see more

• Reply • Share ›

raghson • 2 years ago

I think the condition in second for loop should be

for(i=1;i<=n-l+1;i++)

should be replaced by

for(i=1;i<n-l+1;i++)

otherwise it goes out of the bound and starts calculating m[i][4] (as far as this example is concerned.) But, the maximum value j can hold is 3 (n-1). Please update the code above or correct me.

• Reply • Share ›

magnet ➚ raghson • 2 years ago

Let L = j ? i + 1 denote the length of the subchain being multiplied. The subchains of length 1 (m[i, i]) are trivial. Then we build up by computing the subchains of length 2, 3, ..., n. The final answer is m[1, n].

Now set up the loop: Observe that if a subchain of length L starts at position i, then j = i + L ? 1. Since, we would like to keep j in bounds, this means we want j ? n, this, in turn, means that we want i + L ? 1 ? n, actually what we are saying here is that we want i ? n ? L +1. This gives us the closed interval for i. So our loop for i runs from 1 to n ? L + 1.:)

• Reply • Share ›

raghson ➚ magnet • 2 years ago

Hey, Thanks for reply.
Though, I would request you to do the following.

Please paste this line
printf ("\n%d %d %d",i,j, m[i][j]);

just before the inner loop exits.

i.e.,

```
if (q < m[i][j])
{ printf("\nChanged!");
m[i][j] = q;}
}
printf ("\n%d %d %d",i,j, m[i][j]);
}
}
```

You will get line for j =4 and m[i][j] will give garbage values which is obvious because m[i][4] does not exist in the current matrix. Max. value hold by column variable is 3. Please correct me if I am wrong.

1 • Reply • Share ›

magnet • 2 years ago

Can you please explain recursive version of the code.

• Reply • Share ›

magnet magnet • 2 years ago

beautifully explained

http://www.personal.kent.edu/~...

4 • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

i need help for coding this function in java...

Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago