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.

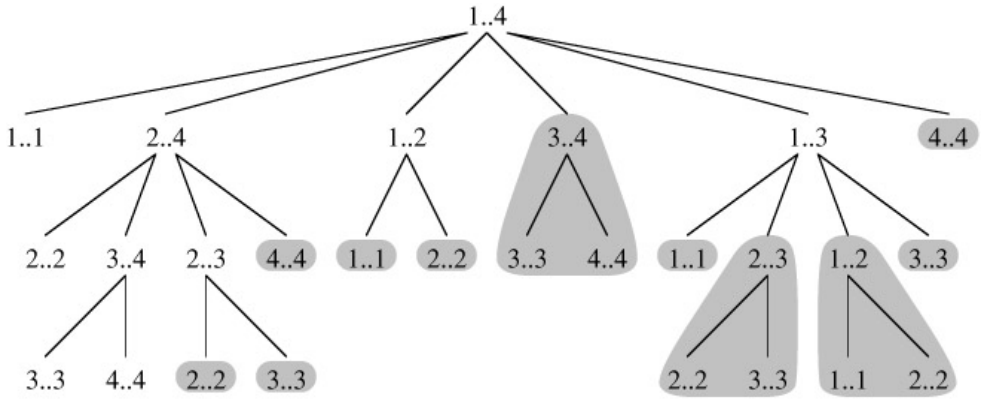
Minimum number of multiplication needed to multiply a chain of size n = Minimum of all n-1 placements (these placements create subproblems of smaller size)

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 ar
       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:

- Calculate the angle between hour hand and minute hand
- Dynamic Programming | Set 37 (Boolean Parenthesization Problem)
- Find the smallest number whose digits multiply to a given number n
- Write a function that generates one of 3 numbers according to given probabilities
- Horner's Method for Polynomial Evaluation
- Count trailing zeroes in factorial of a number
- Program for nth Catalan Number
- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)

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

Sort by Newest ▾ Share Favorite ★

Join the discussion…

Guest • 20 days ago
My code with complexity O(n^2)
http://ideone.com/if7iRp
^ | v • Reply • Share ›

danny ➔ Guest • 16 days 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...
^ | v • Reply • Share ›

rb001 • a month ago
How is the Auxiliary space of the order O(n^2)?
^ | v • Reply • Share ›

Gaurav Gupta • a month ago
My approach with complexity of O(n^2);
http://ideone.com/tU8ckl
^ | v • Reply • Share ›

nitishjn • a month 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&...
^ | v • Reply • Share ›

Aveek Biswas • a month 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].
1 ^ | v • Reply • Share ›

ASHISH • 3 months ago
for (i=1; i<=n-L+1; i++).....
it should be for (i=1; i < n-L+1; i++) .....
plz.... check once .....
3 ^ | v • Reply • Share ›

Shubham • 4 months 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]

• Reply • Share ›

Guest • 4 months ago

what is "INT_MAX" used in the above programs ?

• Reply • Share ›

Tarzan ➚ Guest • 4 months ago

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

• Reply • Share ›

paras_meena • 6 months 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 • 7 months 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

• Reply • Share ›

Ali • 7 months 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 • 11 months 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 ›

Kartik ➔ Vinodhini • 3 months 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 • 7 months ago

yes, you r right.

1 • Reply • Share ›

bhumit • 11 months 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 • a year 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 • a year 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;
  cin>>N;
  dim d[N+1];
  int dp[N+1][N+1];
  cout<<"Enter size array\n";
```

```
 cin>>d[1].left;
 for(i=2;i<=N;i++)
 {
    cin>>pp;
    d[i-1].right=pp;
    d[i].left=pp;
 }
 cin>>pp;
 d[N].right=pp;



 //matrix print
// for(i=1;i<=N+1;i++)
 // cout<<d[i].left<<"   "<<d[i].right<<endl;
 //cout<<"done\n\n";

 //main logic
 for(i=1;i<=N;i++)
 {

   dp[i][i]=0;

 }

 for(i=N-1;i>=1;i--)
 {
    for(j=i+1;j<=N;j++)
    {
        int s1=dp[i][j-1];
        int s2=dp[i+1][j];
        int cost1=d[i].left*d[j-1].right*d[j].right;
        int cost2=d[i+1].left*d[j].right*d[i].left;
        dp[i][j]=min(s1+cost1,s2+cost2);
   }
 }
//print Dp
for(i=1;i<=N;i++)
{
   for(j=1;j<=N;j++)
    cout<<dp[i][j]<<"  ";
    cout<<"\n";
}
cout<<"cost          "<<endl;
cout<<dp[1][N]<<endl;
```

```
  return 0;
} */
```

• Reply • Share ›

piki • a year 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;
  cin>>N;
  dim d[N+1];
  int dp[N+1][N+1];
  cout<<"Enter size array\n";
  cin>>d[1].left;
  for(i=2;i<=N;i++)
  {
     cin>>pp;
     d[i-1].right=pp;
     d[i].left=pp;
  }
  cin>>pp;
  d[N].right=pp;


  //matrix print
 // for(i=1;i<=N+1;i++)
  // cout<<d[i].left<<"   "<<d[i].right<<endl;
  //cout<<"done\n\n";
```

```
  //main logic
  for(i=1;i<=N;i++)
  {

    dp[i][i]=0;

  }

  for(i=N-1;i>=1;i--)
  {
     for(j=i+1;j<=N;j++)
     {
         int s1=dp[i][j-1];
         int s2=dp[i+1][j];
         int cost1=d[i].left*d[j-1].right*d[j].right;
         int cost2=d[i+1].left*d[j].right*d[i].left;
         dp[i][j]=min(s1+cost1,s2+cost2);
    }
  }
 //print Dp
 for(i=1;i<=N;i++)
 {
    for(j=1;j<=N;j++)
     cout<<dp[i][j]<<"   ";
     cout<<"\n";
 }
 cout<<"cost          "<<endl;
 cout<<dp[1][N]<<endl;
  return 0;
} (You may delete these lines if not writing code) */
```

1 ▲ | ▼ · Reply · Share ›

equation · a year 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++){

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

    int j=i+l-1;

    m[i][j]=INT_MAX;

    for(int k=i;k<j;k++)
    {
    int q=(m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j]);

    if(m[i][j]>q)
    m[i][j]=q;

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


int main(){

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

    cout<<chain_mul(p,size-1);

    getchar();
    return 0;
}
```

• Reply • Share ›

raghson • a year 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 • a year 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 • a year 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 ^ | v • Reply • Share ›

magnet • a year ago

Can you please explain recursive version of the code.

^ | v • Reply • Share ›

magnet ➚ magnet • a year ago

beautifully explained

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

3 ^ | v • Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

^ | v • Reply • Share ›

Akilah James • a year ago

How would you display the matrix grouping information?

^ | v • Reply • Share ›

dambigan • a year ago

can anyone tell me how to do the same using one dimensional matrix m and s.

^ | v • Reply • Share ›

Ravi Rank • a year ago

lovely code yaar.

^ | v • Reply • Share ›

Shafqat Wali Khan • a year ago

good approached

^ | v • Reply • Share ›

Ajeet Ganga • a year ago

I am trying to understand if there is any reason for preferring bottom-up approach over top-down. Top down approach will be easy to write (since you just cache the function return value before leaving your recursive function) AND it is lazy initialization, which means only the calculations that are required are performed. The second part may not be applicable to this problem but in problems such as coin change it would make a lot difference.

^ | v • Reply • Share ›

Abdullah Hossain • a year ago

I don&#039t get proper answer.

• Reply • Share ›

Sandeep Jain • a year ago

Akash, please try it with a C99 standard compiler, you can also try it on ideone.com

• Reply • Share ›

Akash Porwal • a year ago

the code is asking to give constant value for n in int m[n][n]

• Reply • Share ›

Shikhar Singh • a year ago

The given algorithm is wrong

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

GeeksforGeeks ➚ Shikhar Singh • a year ago

Could you provide more details? Any input for which the algorithm doesn't give correct output?

• Reply • Share ›

Karthick • 2 years ago

This is a great web site. Lot of interesting info. Kudos.
My first comment/contribution.

A good extension to this problem would be to know the multiplication-sequence/multiplication-result (of the minimum count of multiplications that has been found).

This should be easy to do with the following IF block

```
if (q < m[i][j])
            m[i][j] = q;
```

modified to maintain one more detail (the K value that determines the parenthesis)

```
if (q < m[i][j]){
            m[i][j] = q;
            mult_order[i][j]= k;
}
```

see more

• Reply • Share ›

swan • 2 years ago

I think i should stop such that j becomes less than n

```
/* Paste your code here (You may delete these lines if not writing code) */
```

// L is chain length.
for (L=2; L<n; L++)
{
for (i=1; i<=n-L+1; i++) <---------- may be i<n-L+1
{
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;
}
}
}

• Reply • Share ›

kartikaditya • 2 years ago

[sourcecode language="JAVA"]
public class MatrixChainMult {
private static class Matrix {
int n, m;
}

public int leastCostToMult(Matrix mats[]) {
// Validate mult
for (int i = 1; i < mats.length; ++i) {
if (mats[i].n != mats[i - 1].m) {
return -1;
}
}

int dp[mats.length][mats.length];
for (int i = 0; i < mats.length; ++i) {
dp[i][i] = 0;

```
}
```

see more

• Reply • Share ›

vikas368 • 2 years ago

```
 for (L=2; L<n; L++)
should be
for (L=2; L<=n; L++)

return value should be
return m[1][n];
```

• Reply • Share ›

kartik ➚ vikas368 • 2 years ago

@vikas368: The conditions given in the original program look correct. Could you please let us know why you felt that the conditions are incorrect? Did the original program fail for any case?

• Reply • Share ›

PsychoCoder • 2 years ago

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

Here loop must iterate upto n. i.e. as index goes from 1 to n

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

• Reply • Share ›

kartik ➚ PsychoCoder • 2 years ago

@PsychoCoder: Take a closer look at the program. Size of m[][] is nxn as there will n-1 matrices represented by an array p[] of size n.

• Reply • Share ›

nileshbits ➚ kartik • 2 years ago

Well there is a little contradiction in the code... the first line says - "// Matrix Ai has dimension p[i-1] x p[i] for i = 1..n" which seems to imply there are n matrices and p is filled from 0 to n.

I was surprised to see it return m[1][n-1]. I did a dry run and I'm convinced that there is a flaw somewhere in that code. Here's how:

Lets take the last iterations for all the loops:
L = n-1
i = n-L+1 = n-(n-1)+1 = 2
j = i+L-1 = 2+(n-1)-1 = n

Wait - that makes it [2][n] - which means you are calculating upto m[2][n] --> which is bound to be erroneous since you only declared m[n][n] and that means it should be used only upto (n-1) like you pointed out.

You might want to check that piece of code.

On the other hand - great resource mate! Gotta thank you for this website :)

 • Reply • Share ›

g1 • 2 years ago

how to print placements of parenthesis in the minimum cost solution?

 • Reply • Share ›

Ikshu ➔ g1 • 11 months ago

1) Make an auxilliary matrix s[n][n] to store "k" that gives an optimal placement of parenthesis.
2) Change the code given to one shown below

```
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;
         s[i][j]=k;
     }
 }
```

3) Use the following function to print the expression with parenthesis.

```
 /* Paste your code here (You may delete these lines if not writing code) */voi
{
    if (i==j) cout<<'A'<<i;
```

```
    else
    {
        cout<<"(";
        printparenthesis(i,s[i][j]);
        printparenthesis(s[i][j]+1,j);
        cout<<")";
    }
}
```

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding "extern" keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- Darzen

  int my_rand() // returns 1 to 9 with equal...

  Generate integer from 1 to 7 with equal probability · 0 minutes ago

- junmin

not sure how "css selectors" and "Invalid data"...

Amazon Interview | Set 93 · 26 minutes ago

- Mahek

  i think becase of u r swaping structure not...

  Pairwise swap elements of a given linked list · 1 hour ago

- Pushkar

  Correct but you have to check whether left...

  Convert an arbitrary Binary Tree to a tree that holds Children Sum Property · 1 hour ago

- Pushkar

  Instead of using increment() function we can...

  Convert an arbitrary Binary Tree to a tree that holds Children Sum Property · 1 hour ago

- junmin

  i wonder if we should do: if t[0] != 0 or t[1]...

  Given an array of strings, find if the strings can be chained to form circle · 4 hours ago