a.计算三个矩阵连乘{A1，A2，A3}；维数分别为10\*100 , 100\*5 , 5\*50

     按此顺序计算需要的次数（（A1\*A2）\*A3）:10X100X5+10X5X50=7500次

     按此顺序计算需要的次数（A1\*（A2\*A3））:10X5X50+10X100X50=75000次

它们的乘法次数相差1000倍

b.两种，蛮力算法和动态规划。

C.

代码：

#include <stdio.h> //矩阵连乘

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

const int L = 7;

int RecurMatrixChain(int i,int j,int \*\*s,int \*p);//递归求最优解

void Traceback(int i,int j,int \*\*s);//构造最优解

int main()

{

int p[L]={30,35,15,5,10,20,25};//矩阵A1...A6

printf("矩阵A1到A6：");

for(int z=0;z<L;z++)

printf("%d ",p[z]);

printf("\n");

int \*\*s = new int \*[L];

for(int i=0;i<L;i++)

s[i] = new int[L];

cout<<"矩阵的最少计算次数为："<<RecurMatrixChain(1,6,s,p)<<endl;

cout<<"矩阵最优计算次序为："<<endl;

Traceback(1,6,s);

return 0;

}

int RecurMatrixChain(int i,int j,int \*\*s,int \*p)

{

if(i==j) return 0;

int u = RecurMatrixChain(i,i,s,p)+RecurMatrixChain(i+1,j,s,p)+p[i-1]\*p[i]\*p[j];

s[i][j] = i;

for(int k=i+1; k<j; k++)

{

int t = RecurMatrixChain(i,k,s,p) + RecurMatrixChain(k+1,j,s,p) + p[i-1]\*p[k]\*p[j];

if(t<u)

{

u=t;

s[i][j]=k;

}

}

return u;

}

void Traceback(int i,int j,int \*\*s)

{

if(i==j) return;

Traceback(i,s[i][j],s);

Traceback(s[i][j]+1,j,s);

cout<<"Multiply A"<<i<<","<<s[i][j];

cout<<" and A"<<(s[i][j]+1)<<","<<j<<endl;

}