**实验2 动态规划**

* 1. **实验目的**

1. 熟悉动态规划的思想
2. 熟悉动态规划的步骤
   1. **实验内容**
3. **实现矩阵连乘积**

参考教材第48-50页

范例：

// 重叠子问题的递归最优解

//A1 30\*35 A2 35\*15 A3 15\*5 A4 5\*10 A5 10\*20 A6 20\*25

//p[0-6]={30,35,15,5,10,20,25}

#include <iostream>

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};

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;

}

1. **实现最长公共子序列**

参考教材第54-56页

请使用动态规划查找如下两个序列的最长公共子序列：

char x[] = {' ','A','B','C','B','D','A','B'};

char y[] = {' ','B','D','C','A','B','A'};

1. **实现最大字段和（选做）**

参考教材第59-60页

请使用动态规划求a[]={-2,11,-4,13,-5,-2}; 得最大字段和。