算法设计与分析 实验报告

原创

sunjiangangok 于 2017-04-10 10:24:17 发布
分类专栏： C/C++

算法设计与分析
实验报告

题目一：矩阵相乘
题目二：最长公共子序列

题目一：矩阵相乘
一．问题描述
给定n个矩阵｛A1，A2，…，An｝，其中这n个矩阵是可相乘的，i＝1，2，…，n-1。算出这n个矩阵的相乘积A1A2 。。。An。
补充：如果两个矩阵A和B是可相乘的，那么A的列数要和B的行数是相同的，否则，这两个矩阵是不可相乘的。它们的相乘结果矩阵C的行数是A的行数，而列数是B的列数。
二．问题分析
由于矩阵乘法满足结合律，故连乘积的计算可以有许多不同的计算次序。这种计算次序可以用加括号的方式来确定。若一个矩阵连乘积的计算次序已完全确定，也就是说该连乘积已完全加括号，则我们可以通过反复调用两个矩阵相乘的标准算法计算出矩阵连乘积。
1.分析最优解的结构
为了方便起见，我们将矩阵连乘AiAi+1 。。。Aj记为A[i:j]。经分析，计算　A[1:n]的一个最优次序所包含的计算矩阵子链A[1:k]和A[k:n]的次序也是最优的。因此，矩阵连乘计算次序问题的最优解包含着子问题的最优解。
2.建立递归关系
用矩阵m[n][n]来存放A[i:j]相乘的计算次数，用p[n+1]用来存放矩阵的行数和列数。
 0　　i=j

min={ m[i][k]+m[k+1][j]+pi-1pkpj }　　i
3.计算最优值
　另外，用一个数组s[n][n]来记录相应m[i][j]的分割下标
注：数组元素的下标都是从0开始的，而不是1

```
void MatrixChain(int *p, int n, int **m, int **s) {
    int i, j, r, k, t;
    for(i = 0; i < n; i++) {
        m[i][i] = 0;
        s[i][i] = 0;
    }
    for(r = 1; r < n; r++) {
        for(i = 0; i < n-r; i++) {
            j = i+r;
            m[i][j] = m[i][i]+m[i+1][j]+p[i]*p[i+1]*p[j+1];
            s[i][j] = i;
            for(k = i+1; k < j; k++) {
                t = m[i][k]+m[k+1][j]+p[i]*p[k+1]*p[j+1];
                if(t < m[i][j]) {
                    m[i][j] = t;
                    s[i][j] = k;
                }
```

```
        }
      }
    }
```

4.构造最优解

根据上一步的m和s中的值，构造出最优解。

```
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"< <<","<
    cout<<"and A"<<(s[i][j] +1)<<","< <
}
```

三．程序源代码

使用c++语言来实现

```
#include
#include
#define X 10       //The count of array
#define M 10       //The Row
#define N 10       //The Column
#define Stack_Size 20
using namespace std;
void MatrixChain(int *p, int n, int **m, int **s) {
    int i, j, r, k, t;
    for(i = 0; i < n; i++) {
        m[i][i] = 0;
        s[i][i] = 0;
    }
    for(r = 1; r < n; r++) {
        for(i = 0; i < n-r; i++) {
            j = i+r;
            m[i][j] = m[i][i]+m[i+1][j]+p[i]*p[i+1]*p[j+1];
            s[i][j] = i;
            for(k = i+1; k < j; k++) {
                t = m[i][k]+m[k+1][j]+p[i]*p[k+1]*p[j+1];
                if(t < m[i][j]) {
                    m[i][j] = t;
                    s[i][j] = k;
                }
            }
        }
    }
}
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"< <<","<
    cout<<"and A"<<(s[i][j] +1)<<","< <
}
int main()
{
    ifstream fin("data.in");
    int i, j, data[X][M][N], p[X+1], pos, count;
    int **m, **s;
    m = newint*[X+1];
    s = newint*[X+1];
    for(i = 0; i < X+1; i++) {
        m[i] = newint[X+1];
        s[i] = newint[X+1];
    }
    for(i = 0; i < X+1; i++) {
        for(j = 0; j < X+1; j++) {
```

```
            m[i][j] = s[i][j] = 0;
        }
    }
    if(!fin.is_open()) {
        cout<<"Can not open the file data.in"<
        return 1;
    }
    //read the data from the file
    fin>>count;
    for(pos = 0; pos < count; pos++) {
        fin>>p[pos]>>p[pos+1];
    }
    MatrixChain(p, count, m, s);
    Traceback(0, count-1, s);
    fin.close();
    for(i = 0; i < X+1; i++) {
        delete m[i];
        delete s[i];
    }
    delete m;
    delete s;
    return 0;
}
```

四．输入

输入的一个名为"data.in"的文件，其中的一个测试内容为:

```
5
4 5
5 3
3 3
3 4
4 6
```

五．输出

在屏幕上输出的结果是

```
Multiply A0,0and A1,1
Multiply A2,2and A3,3
Multiply A2,3and A4,4
Multiply A0,1and A2,4
```

题目二：最长公共子序列

一．问题描述

一个给定序列的子序列是在该序列中删去若干元素后得到的序列。在本题中，给定了两个序列X和Y，当另一个序列Z既是X的子序列又是Y的子序列时，我们就称Z为X和Y的公共子序列。当然这可能会有许多种，而我们又把其中最长的一个称为最长公共子序列。在本题中我们给了X和Y的序列，要计算出X和Y的最长公共子序列。

二．问题分析

1.最长公共子序列的结构

设序列X＝{x1,x2,...,xm},Y={y1,y2,...,yn}，它们的最长公共子序列是 Z={z1,z2,...,zk}，则

若xm＝yn，则zk＝xm＝yn，且zk－1是Xm－1和Yn－1的最长公共子序列

若xm≠yn,且zk≠xm，则z是Xm－1和Y的最长公共子序列

若xm≠yn,且zk≠Yn，则z是X和Yn－1的最长公共子序列

2.子问题的递归结构

```
0                   i=0,j=0
c[i-1][j-1]+1           i,j>0;xi=yj
max{c[i][j-1],c[i-1][j]}    i,j>0;xi≠yj
```

三．程序源代码

使用c++语言来完成

```
#include
#include
#include
using namespace std;
#define N 100
ofstream fout;
void LCSLength(int m, int n, char *x, char *y, int **c, int **b)
{
    int i, j;

    for(i = 1; i <= m; i++) {
```

```
        c[i][0] = 0;
    }
    for(i = 1; i <= n; i++) {
        c[0][i] = 0;
    }
    for(i = 1; i <= m; i++) {
        for(j = 1; j <= n; j++) {
            if(x[i] == y[j]) {
                c[i][j] = c[i-1][j-1]+1;
                b[i][j] = 0;
            } else if(c[i-1][j] >= c[i][j-1]) {
                c[i][j] = c[i-1][j];
                b[i][j] = 1;
            } else {
                c[i][j] = c[i][j-1];
                b[i][j] = 2;
            }
        }
    }
}
void LCS(int i, int j, char *x, int **b) {
    if(i == 0 || j == 0) {
        return;
    }
    if(b[i][j] == 0) {
        LCS(i-1, j-1, x, b);
        fout<
    } else if(b[i][j] == 1) {
        LCS(i-1, j, x, b);
    } else {
        LCS(i, j-1, x, b);
    }
}
int main() {
    ifstream fin("data.in");
    char *x, *y;
    int m, n, **c, **b, i, j;
    x = newchar[N];
    y = newchar[N];
    c = newint*[N];
    b = newint*[N];
    for(i = 0; i < N; i++) {
        c[i] = newint[N];
        b[i] = newint[N];
    }
    if(!fin.is_open()) {
        cout<<"Can not open the file \'data.in\'"<
        return 1;
    }
    fout.open("data.out");
    if(!fout.is_open()) {
        cout<<"Can not open the file \'data.in\'"<
        return 1;
    }
    while(!fin.eof()) {
        fin.getline(x, N);
        fin.getline(y, N);
    }
    m = strlen(x);
    n = strlen(y);
    for(i = 0; i <= m; i++) {
        for(j = 0; j <= n; j++) {
            c[i][j] = b[i][j] = 0;
        }
    }
```

```
    LCSLength(m, n, x, y, c, b);
    LCS(m, n, x, b);
    fin.close();
    fout.close();
    delete x;
    delete y;
    for(i = 0; i < N; i++) {
        delete c[i];
        delete b[i];
    }
    delete c;
    delete b;
    return 0;
}
```

四．输入

文件"data.in"

```
sunjiangangoksunjiangnag
baixiaotongabasdfasfassunjiangns
```

五．输出

文件"data.out"

```
iangasunjiangn
```


上一篇：用c语言模拟进程调度

下一篇：Linux网络编程一步一步学＋基础