动态规划专题讲义

专题九：合并石子问题

/\*

Name: 动态规划专题之石子合并

Author: 巧若拙

Description:

在一个操场上摆放着一排N堆石子。现要将石子有次序地合并成一堆。规定每次只能选相邻的2堆石子合并成新的一堆，并将新的一堆石子数记为该次合并的得分。

试设计一个算法，计算出将N堆石子合并成一堆的最小得分。

输入描述 Input Description

第一行是一个数N。1≤N≤100

以下N行每行一个数A，表示石子数目。1≤A≤200

输出描述 Output Description

共一个数，即N堆石子合并成一堆的最小得分。

样例输入 Sample Input

7

13 7 8 16 21 4 18

样例输出 Sample Output

239

\*/

#include<iostream>

using namespace std;

const int INT\_MAX = 2147483647;

const int MAX = 1000;

int A[MAX+1];//记录每堆石子的数量

int Sum[MAX+1];//记录前n堆石子的数量

int B[MAX+1][MAX+1];//记录第i堆石子至第j堆石子的最优解

int S[MAX+1][MAX+1];//记录从哪里断开的才可得到最优解

bool flag[MAX+1]; //记录A[i]是否已经被输出过

int StonesCombined(int i, int j);//自顶向下，使用备忘录数组的动态规划算法

int StonesCombined\_2(int n);//自底向上的动态规划算法：递增括号中石子堆数量

int StonesCombined\_3(int n);//自底向上的动态规划算法：逆序扫描

int StonesCombined\_4(int n);//自底向上的动态规划算法：顺序扫描

void Traceback(int i, int j);//根据s[][]记录的各个子段的最优解，将其输出

int main()

{

int n = 0;

cin >> n;

for (int i=1; i<=n; i++)

{

cin >> A[i];

Sum[i] = Sum[i-1] + A[i];

}

// cout << StonesCombined(1, n) << endl;//自顶向下，使用备忘录数组的动态规划算法

cout << StonesCombined\_2(n) << endl;//自底向上的动态规划算法

Traceback(1, n);

cout << endl;

return 0;

}

算法1：自顶向下，使用备忘录数组的动态规划算法，需要用到全局变量S[MAX+1][], 另有B[MAX+1][]初始化为0。

int StonesCombined(int i, int j)

{

if (B[i][j] != 0)

return //语句1

if (i == j)

return //语句2

int t, m = INT\_MAX;

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

{

t = StonesCombined(i, k) + StonesCombined(k+1, j) + Sum[j] - Sum[i-1];

if (t < m)

{

m = t;

S[i][j] = k; //记录从哪里断开的才可得到最优解

}

}

return B[i][j] = m;

}

问题1：将语句1和语句2补充完整。

参考答案：

问题1：语句1：return B[i][j];

语句2：return 0;

算法2：自底向上的动态规划算法：递增括号中石子堆数量，需要用到全局变量S[MAX+1][], 另有B[MAX+1][]初始化为0。

int StonesCombined\_2(int n)

{

for (int len=2; len<=n; len++) //语句1

{

for (int i=1; i<=n-len+1; i++) //左边界

{

int j = //语句2

int t, m = INT\_MAX;

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

{

t = B[i][k] + B[k+1][j] + Sum[j] - Sum[i-1];

if (t < m)

{

m = t;

S[i][j] = k;

}

}

B[i][j] = m;

}

}

return B[1][n];

}

问题1：语句1能否改为：for (int len=n; len>=2; len--) ？为什么？

问题2：将语句2补充完整。

参考答案：

问题1：不能。len代表当前被合并在一起的石子堆的数量，算法的思想是先计算所有相邻的2堆石子合并在一起的解，再计算所有所有相邻的3堆石子合并在一起的解，逐步扩大子问题的规模。在计算较大规模的问题时，可以调用较小规模子问题的解。所以len必须从小到大。

问题2：语句2：int j = i + len - 1; //右边界

算法3：自底向上的动态规划算法：逆序扫描，需要用到全局变量S[MAX+1][], 另有B[MAX+1][]初始化为0。

int StonesCombined\_3(int n)

{

for (int i=n-1; i>0; i--)

{

for (int j=i+1; j<=n; j++)// 语句1

{

int t, m = INT\_MAX;

for (int k=i; k<j; k++) // 语句2

{

t = B[i][k] + B[k+1][j] + Sum[j] - Sum[i-1];

if (t < m)

{

m = t;

S[i][j] = k;

}

}

B[i][j] = m;

}

}

return B[1][n];

}

问题1：语句1能否改为：for (int j=n; j>i; j--)？为什么？

问题2：语句2能否改为：for (int k=j-1; k>=i; k--)？为什么？

参考答案：

问题1：不能。i和j分别代表当前被合并石子堆的左，右边界，算法3的基本思想是在确定左边界的情况下，逐步扩大子问题的右边界。在计算较大规模的问题时，可以调用较小规模子问题的解。所以j必须递增。

问题2：可以。k代表分隔石子堆的位置（即把[i,k]和[k+1,j]两堆石子合并成[i,j]一堆石子），只要满足i<=k<j，k递增或递减均可。

算法4：自底向上的动态规划算法：顺序扫描，需要用到全局变量S[MAX+1][], 另有B[MAX+1][]初始化为0。

int StonesCombined\_4(int n)

{

for (int j=2; j<=n; j++)

{

for (int i=j-1; i>0; i--)// 语句1

{

int t, m = INT\_MAX;

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

{

t = B[i][k] + B[k+1][j] + Sum[j] - Sum[i-1];

if (t < m)

{

m = t;

S[i][j] = k;

}

}

B[i][j] = m;

}

}

return B[1][n];

}

问题1：语句1能否改为：for (int i=1; i<j; i++)？为什么？

参考答案：

问题1：不能。i和j分别代表当前被合并石子堆的左，右边界，算法4和算法3的思路刚好相反，它是在确定右边界的情况下，逐步扩大子问题的左边界，所以i必须递减。

拓展练习：原题只要求输出最优解，并未要求输出具体的合并方法。

但是我们在上述算法中引入了一个二维数组S[][]记录k的值，即从哪里分开石子堆才能得到最优解。现在请你根据S[][]记录的信息，设计一个递归函数void Traceback(int i,int j);// i和j分别代表当前被合并石子堆的左，右边界。

请用括号把合并在一起的石子堆括起来，输出最终合并情形。

例如：对于题目给出的样例：13 7 8 16 21 4 18，则输出：(((13 (7 8))16)(21 (4 18)))

参考答案：

void Traceback(int i,int j)//根据s[][]记录的各个子段的最优解，将其输出

{

if (i == j)

return ;

cout << "(";

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

if (flag[i] == 0)

{

cout << A[i] << " ";

flag[i] = 1;

}

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

if (flag[j] == 0)

{

cout << A[j];

flag[j] = 1;

}

cout << ")";

}

课后练习：

练习1： 3546\_矩阵链乘法

描述：给定有n个要相乘的矩阵构成的序列（链）<A1,A2,A3,.......,An>，要计算乘积A1A2.....An。一组矩阵是加全部括号的。

矩阵链加括号对运算的性能有很大影响。

仅当两个矩阵A和B相容（即A的列数等于B的行数），才可以进行相乘运算。如果A是一个p×q矩阵，B是q×r矩阵，结果C是p×r的矩阵。

计算C的时间由乘法运算次数决定的，次数为p×q×r。

矩阵链乘法问题可表述为：给定n个矩阵构成的一个链<A1，A2，A3.......，An>，其中i=1,2,3,4.....,n，矩阵Ai的维数为Pi-1 ×Pi，

对乘积A1A2A3.....An，以一种最小标量乘法次数的方式进行加全部括号。

输入描述 Input Description

如果有n个数组，则第一行输入n+1个整数值

输出描述 Output Description

所有的数组以一种最小标量乘法次数的方式进行加全部括号。

样例输入 Sample Input

1 2 3 4

样例输出 Sample Output

((A1A2)A3)