hw-8

0524

1. 设计算法求出n个矩阵M1、M2、…、Mn相乘最多需要多少次乘法，请给出详细的算法描述和时间复杂性

Solution :

给定n+1个正整数$c_0, c_1, \cdots c_n$，其中c_{i-1}, c_i是M_i的行数和列数，使用动态规划求解

记M_{ij}为$M_i \cdots M_j$的乘积，$Q(i,j)$为计算M_{ij}所需要的最多的乘法个数$(i<j)$，则状态转移方程为

$$Q(i,\ j)=\begin{cases}\min\limits_{i\le k<j}\{Q(i,k)+Q(k+1,j)+c_{i-1}c_kc_j\} & ,i<j\\ 0 & ,i=j\end{cases}$$

算法的时间复杂度为$O(n^3)$，伪代码如下

```
Input: c0~cn
Output: maxnum
Q[1...n][1...n]={0};
int cur,t;
int maxnum = 0;
for (int j=2; j<=n; j++)
{
    for (int i=j-1; i>=0; i--)
    {
        cur = c[i-1]*c[j];
        for (int k=i; i<=j-1; k++)
        {
            t = Q[i][k] + Q[k+1][j] + cur*c[k];
            if (t > Q[i][j])
                Q[i][j] = t;
        }
    }
}
return Q[1][n];
```

2. 将正整数n表示成一系列正整数之和：n=n1+n2+⋯+nk，其中n1⩾n2⩾⋯⩾nk⩾1，k⩾1。正整数n的这种表示称为正整数n的划分，例如正整数6有如下11种不同的划分：6；5+1；4+2，4+1+1；3+3，3+2+1，3+1+1+1；2+2+2，2+2+1+1，2+1+1+1+1；1+1+1+1+1+1。设计算法求正整数n的不同划分个数并证明其时间复杂性为$\Theta(n^2)$

Solution :

记$f(n,m)$表示满足题目要求且最大数不超过m的分划的个数

根据n和m的关系，考虑下面几种情况：

（1）当n=1时，不论m的值为多少（m>0），只有一种划分，即{1}；

（2）当m=1时，不论n的值为多少（n>0），只有一种划分，即{1,1,....1,1,1}；

（3）当n=m时，根据划分中是否包含n，可以分为两种情况：

- 划分中包含n的情况，只有一个，即{n}；
- 划分中不包含n的情况，这时划分中最大的数字也一定比n小，即n的所有（n-1）划分；
- 因此，f(n,n) = 1 + f(n, n - 1)。

（4）当n<m时，由于划分中不可能出现负数，因此就相当于f(n,n)；

（5）当n>m时，根据划分中是否包含m，可以分为两种情况：

- 划分中包含m的情况，划分个数为f(n-m, m)；
- 划分中不包含m的情况，则划分中所有值都比m小，即n的（m-1）划分，个数为f(n, m - 1)；
- 因此，f(n,m) = f(n - m,m) + f(n, m - 1)。

综上，状态转移式如下

$$f(n,\ m)=\begin{cases}1 & ,n=m=1\\ f(n,n-1)+1 & ,n=m\\ f(n,n) & ,n<m\\ f(n,m-1)+f(n-m,m) & ,n>m>1\end{cases}$$

使用数组进行动态规划，从小到大计算状态

```c
int partition(int n)
{
    for(int i=1;i<=n;i++)
        for(int j=1;j<=i;j++)
        {
            if(j==1|| i==1)
            {
                ans[i][j]=1;
            }
            else
            {
                if(j==i)
                    ans[i][j]=ans[i][j-1]+1;
                else if((i-j)<j)
                    ans[i][j]=ans[i-j][i-j]+ans[i][j-1];
                else
                    ans[i][j]=ans[i-j][j]+ans[i][j-1];
            }
        }
    return ans[n][n];
}
```

每个状态的计算需要常数时间完成，总的时间复杂度为$\Theta(n^2)$

0525

1. 区间包含问题的输入是由数轴上的区间所组成的集合，这些区间由它们的两个端点表示。利用二维平面极大点问题设计O(nlogn)算法识别所有包含在集合中其它某个区间的区间。

例：输入是（1，3），（2，8），（4，6），（5，7），（7，9），则输出为（4，6）和（5，7）

Solution :

极大点的定义：不被集合中任意其他点所支配，其中支配的定义为 $p_1:(x_1,y_1),p_2:(x_2,y_2),x_1\le x_2 \,\&\, y_1\le y_2$，则称$p_2$支配$p_1$，类比本题，包含的要求是对区间$(x_1,y_1),(x_2,y_2)$，$x1\ge x_2 \,\&\, y1\le y_2$，因此可以采用归约的思想，对于集合中的某个区间$(x,y)$，我们将其一一映射到二维平面上的一个点$(-x,y)$，则区间的包含问题转化为二维

平面的极大点问题。

极大点问题我们已经在之前的作业中解决(先按x坐标降序排序($O(nlogn)$)，再依次遍历并不断更新当前最大的y坐标($O(n)$)，最后那些非极大点的点即为我们要求的区间的映射，总的时间复杂度为$O(nlogn)$

2. 证明如果存在时间复杂度为O(T(n))的两个n*n下三角矩阵的乘法，则存在时间复杂度为O(T(n)+n2)的任意两个n*n矩阵相乘的算法。设对任意常数c，T(cn)=O(T(n))。

$Solution:$

设 A, B 均为 n×n 的矩阵

令 $C = \begin{pmatrix} 0 & 0 & 0 \\ A & 0 & 0 \\ A & A & 0 \end{pmatrix}$，$D = \begin{pmatrix} 0 & 0 & 0 \\ B & 0 & 0 \\ B & B & 0 \end{pmatrix}$

均为 3n 阶的下三角矩阵

$C \cdot D = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ AB & 0 & 0 \end{pmatrix}$

构造C,D需要$O(n)$的时间，计算C·D需要$O(T(3n)) = O(T(n))$的时间，取出AB的值需要$O(n^2)$

总的时间复杂度为$O(T(n) + n^2)$

3. 证明最小公倍数问题属于P类

$Solution:$

$$m \times n = \gcd(m,n) \times \operatorname{lcm}(m,n)$$
$$\Rightarrow \operatorname{lcm}(m,n) = \frac{m \times n}{\gcd(m,n)}$$

因此问题转化为证明求最小公约数问题属于P类

事实上，使用欧几里得算法(辗转相除)，不妨设$a > b$

$$\gcd(a, b) = \gcd(b, a\%b)$$

而$a\%b$是小于b的整数，这就意味着这个算法一定在线性时间内能结束，故可以在多项式时间内解决，因此是P类问题