黎锦灏　518021910771　0425作业

1、设 Fibonacci 数列的定义为：F(1)=1, F(2)=1, F(n)=F(n-1)+F(n-2) (n>2)，证明每个大于 2 的整数 n 都可以写成至多 logn 个 Fibonacci 数之和，并设计算法对于给定的 n 寻找这样的表示方式

解：由数学归纳法易知：每个大于2的数都可以表示成若干Fibonacci数的和，

而$F(n)=F(n-1)+F(n-2)=F(n-2)+F(n-2)+F(n-3)>2F(n-2)$，

可以看出 $F(n)$ 呈指数增长。

已知每个数可以表示成数列中若干数的和，则对于呈指数增长的数列，即需要用到的Fibonacci数个数不超过 $logn$。

寻找表示方式可以用贪心算法，每次找到小于当前数的最大Fibonacci数，作为选取的一个Fibonacci数，然后反复执行。在找到小于当前数的最大Fibonacci数过程，可以用二分查找在Fibonacci数组搜索。

伪代码如下：

```
Algorithm Fibonacci_expression
Input: n,F[n]
Output: A （the set of Fibonacci number）
begin
    F[1] := 1
    F[2] := 1
    i := 2
    N := n
    while F[i] < N do
        i := i+1
        F[i] := F[i-1] + F[i-2]

    A.add(F[i-1])

    N := N - F[i-1]
    while N do
        search F[k]{ F[k]<N and F[k+1]>N }
        A.add(F[k])
        N := N - F[k]
end
```

2、设有复数 x=a+bi 和 y=c+di，设计算法，只用 3 次乘法计算乘积 xy

解：

$xy=(a+bi)(c+di)=(ac-bd)+(bc+ad)i=[(a+b)(c+d)-ac-bd]i+ac-bd$

故只需计算，ac、bd、$(a+b)(c+d)$ 这三次乘法，就能得到 xy

3、分析在一般微机上用 C/C++如何计算二项式系数C_n^k，能够计算的 n 和 k 的范围越大越好

解：$C_n^k = \frac{n!}{(n-k)!k!}$，即计算 C_n^k 实际需要计算 $\prod_{i=0}^{k-1} \frac{n-i}{i+1}$，在每一步计算时，需要乘一个新计算的分数 $\frac{n-i}{i+1}$，误差太大。在实际实现中，可以采取分别计算分子、分母，最后计算分数的方式减少误差，但是也存在乘积太大溢出的情况。

还有一种应用比较广泛的做法，即利用：

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

可以实现递归计算。

黎锦灏　518021910771　0427 作业

1、设 P 是一个 n 位十进制正整数，如果将 P 划分为 k 段，则可得到 k 个正整数，这 k 个正整数的乘积称为 P 的一个 k 乘积。1）求出 1234 的所有 2 乘积；2）对于给定的 P 和 k，求出 P 的最大 k 乘积的值。

解：1）1234可以拆分为1|234、12|34、123|4，即 2 乘积有234、408、492.

2）设 $f[i][t]$ 为 i位数分为 t 段的最大 t 乘积，则

$$f[i][t] = max_{1\leq j<i}(f[j][t-1] + P[j+1]...P[i])$$

最后 $f[n][k]$ 即为所求。

伪代码如下：

```
Algorithm:div_k
Input: p,k
Output:the largest k-product value
begin
    b := 1
    q := 1
    for i := n to 1 do
        v := 10
        b := p / q
        q := q * 10
        for j := 1 to i do
            P[j][i] := b % v
            v := v * 10

    for i := 1 to n do
        for j := 1 to i do
            if j == 1 then
                f[i][j] := P[1][i]
                continue
```

```
            for t:=1 to i do
                f[i][j]=max(f[i][j], f[t][j-1]*P[nn][i]);
    output f[n][k]
end
```

2、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

解：给定 $n+1$ 个正整数 $r_1, r_2, \ldots, r_{n+1}$，其中$r_i$ 和 r_{i+1} 为矩阵 M_i 的行数和列数，$1 \le i \le n$.

记 M_{ij} 为 $M_i M_{i+1} \ldots M_j$的乘积，$f[i][j]$表示计算 M_{ij}所需要的最多乘法数量，则

$$f[i][j] = max_{i<k\le j}(f[i][k-1] + f[k][j] + r_i r_k r_{j+1})$$

最后只需要取 $f[1][n]$即为所求。

伪代码如下：

```
Algorithm:MATCHAIN
Input: An array r[1..n+1] of positive integers corresponding to the dimensions
of a chain of n matrices, where r[1..n] are the number of rows in the n matrices
and r[n+1] is the number of columns in Mn

Output: The most number of scalar multiplications required to multiply the n
matrices

begin
    for i := 1 to n      {Fill in diagonal d0}
        f[i,i] := 0
    end for

    for d := 1 to n-1      {Fill in diagonals d1 to dn-1}
        for i := 1 to n-d      {Fill in entries in diagonal di}
             j := i+d
             comment: The next three lines computes f[i,j]
             f[i,j] := 0
             for k := i+1 to j
                f[i,j] := max{f[i,j],f[i,k-1]+f[k,j]+r[i]*r[k]*r[j+1]}
             end for
         end for
    end for
    output C[1,n]
end
```

3、设有算法 A 能够在$O(i)$时间内计算一个 i 次多项式和一个 1 次多项式的乘积，算法 B 能够在 $O(ilogi)$ 时间内计算两个 i 次多项式的乘积。现给定 d 个整数 $n_1, n_2, \ldots, n_d$，设计算法求出满足 $P(n_1) = P(n_2) = \ldots = P(n_d) = 0$ 且最高次项系数为 1 的 d 次多项式 $P(x)$，并给出算法的时间复杂性

解：依题意有：$P(x) = (x-n_1)(x-n_2)(x-n_3)\ldots(x-n_d)$，只需展开即可得多项式。

考虑子问题分解，若次数为偶数则采用算法 B 分成两个相等的部分，并分别递归；若次数为奇数则还需要调用一次算法 A ，得到最终结果。

故分析算法的复杂度：

$$T(d) = 2T(\frac{d}{2}) + O(\frac{d}{2}log\frac{d}{2}), d > 1$$

而边界条件：$T(1) = 1$。

解得：$T(d) = O(dlog^2 d)$

综上所述，时间复杂性为：$O(dlog^2 d)$