算法与复杂性 第七次课程作业

UNIkeEN

2023 年 5 月 18 日

问题解答

1. 海报墙由 n 块宽度相同高度不同的木板组成，那么在此海报墙上能够张贴的最大海报面积是多少？设木板宽度为 1，高度为 $h_1, h_2, ..., h_n$，海报必须整体都粘贴在墙上，并且不能斜贴。

解 使用单调栈方法。建立一个单调递增的栈，栈中保存木板的索引。

- 遍历木板的高度，对于每一个高度 hi，如果栈为空或者 hi 大于栈顶索引对应的木板高度，将当前木板的索引入栈；否则，重复以下操作直到 hi 大于栈顶索引对应的木板高度:
- 从栈顶取出一个索引，假设为 j，计算以栈顶索引对应的木板为最低木板的最大海报面积：height[j] * (i - (栈顶新的索引 + 1))。更新最大海报面积
- 遍历完成后，如果栈中仍有索引，重复以下操作直到栈为空:
- 从栈顶取出一个索引，假设为 j，计算以栈顶索引对应的木板为最低木板的最大海报面积：height[j] * (n - (栈顶新的索引 + 1))。更新最大海报面积。

最后得到的即最大海报面积。

这一算法是寻找单调递增的木板序列并逐个计算以各个木板为最短木板的海报面积，时间复杂度为 O(n)。

2. 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 $\log n$ 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 首先证明每个大于 2 的整数 n 可以表示成若干个不连续的 Fibonacci 数（不包括第一个）之和，采用数学归纳法证明之。

当 $m = 1, 2, 3$ 时，因为 $1 = F(2), 2 = F(3), 3 = F(4)$, 命题成立。

假设定理对任何小于 n 的正整数都成立。下证命题对 n 也成立。

- 若 n 是 Fibonacci 数，则命题对 n 也成立。
- 若 n 不是 Fibonacci 数，设 t_1 是满足 $F(t_1) < m < F(t_1 + 1)$ 的最大正整数。

设 $n' = n - F(t_1)$, 则 $n' = n - F(t_1) < F(t_1+1) - F(t_1) = F(t_1-1)$, 即 $n' < F(t_1-1)$。

由归纳假设, n' 可以表示成不连续的斐波那契数之和, 即 $n' = F(t_2)+F(t_3)+...+F(t_k)$, 其中 $t_2 > t_3 > ... > t_k$, 且是不连续的整数。又 $n' < F(t_1 - 1)$ $t_2 < t_1 - 1$, 即 t_2 与 t_1 也是不连续的整数。

证明不连续的原理也可简单的阐述为: 如果分解结果中有连续的 Fibonacci 数, 则他们的和也是 Fibonacci 数, 用后者取代前者能得到数量更少的分解结果, 这正是我们想要的。

每个 Fibonacci 数至多使用一次, 且不会有连续两个 Fibonacci 出现在结果中。(第二个之后) 非相邻的两个 Fibonacci 数之间的倍数一定超过 2。易知可以用不超过 $\log_2 n$ 个 Fibonacci 数和表示 n。

通过上述回溯搜索的算法, 即可得到分解方案。

3. 设有复数 $x = a + b\boldsymbol{i}$ 和 $y = c + d\boldsymbol{i}$, 设计算法, 只用 3 次乘法计算乘积 xy

解 易得

$$xy = (a + b\boldsymbol{i})(c + d\boldsymbol{i}) = (ac - bd) + (ad + bc)\boldsymbol{i}$$

分别做如下三次乘法得 A B C:

1. $A = ac$
2. $B = bd$
3. $C = (a + b)(c + d) = ac + ad + bc + bd$

此时由非乘法运算即可求得原式, 符合题意。

$$xy = (a + b\boldsymbol{i})(c + d\boldsymbol{i}) = (A - B) + (C - A - B)\boldsymbol{i}$$

4. 用 C/C++ 如何在 int 范围内计算二项式系数 C_n^k, 能够计算的 n 和 k 的范围越大越好

解 根据 $C_n^k = C_n^{n-k}$, , 首先选取 k 和 $n - k$ 中较小的数作为新的 k 减少运算规模。

经典的方法是使用杨辉三角递推, 逐步扩大问题规模可以避免溢出。递推式: $C_n^k = c_{n-1}^{k-1} + c_{n-1}^k$

也可以直接使用定义 $C_n^k = \frac{n(n-1)...(n-k+1)}{k(k-1)...1}$, 在循环时每次执行一次乘法和一次除法, 而非一次性计算完分子和分母后再做除法。为了避免浮点数运算带来的精度影响, 这可能需要创建存储分数并有通分的数据结构, 但这一方法也可以降低溢出风险, 使计算的 n 和 k 范围大。

5. 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 的 2 乘积有 $1 \times 234 = 234$、$12 \times 34 = 408$ 和 $123 \times 4 = 492$ 三个解。

2）使用动态规划求解。

设 $suffix(t,i)$ 是整数 t 的后 i 位构成的数，$ans(t,k)$ 是整数 t 的最大 k 乘积。则

$$ans(t,k) = \begin{cases} n & ,k = 1 \\ \max_i \{t - suffix(t,i) \times ans(Num(t,i), k-1) \times 10^{-i}\} & ,k > 1 \end{cases}$$

递归计算 $ans(n,k)$ 即可得解。

6. 如何快速计算 $1 \oplus 2 \oplus 3 ... \oplus n$ 的值，其中 $\oplus$ 表示按位异或

解 手动计算了 n 取值从 1 到 16 的结果，寻找规律可得：

- $n \mod 4 = 0$ 时，原式结果为 n
- $n \mod 4 = 1$ 时，原式结果为 1
- $n \mod 4 = 2$ 时，原式结果为 $n + 1$
- $n \mod 4 = 3$ 时，原式结果为 0

如此，通过对 n 做取余运算，即可快速计算原式。