数据与算法的故事: 2

吕正华

2019.10

自我介绍

- Pivotal 资深软件工程师，开发 Greenplum 内核
- kainwen@gmail.com
- 个人主页: https://kainwen.com
- 2014 年毕业于电子系智能感知实验室, 工学硕士
- 2012 年第一次担任数据与算法助教

一些推荐 1

Finding a closed form for this sum is tougher than what we've done so far (except perhaps for the discrepancy problem we just looked at). But it's instructive, so we'll hack away at it for the rest of this chapter.

As usual, especially with tough problems, we start by looking at small cases. The special case $n = 1$ is (3.26), with x replaced by x/m:

$$\left\lfloor \frac{x}{m} \right\rfloor + \left\lfloor \frac{1+x}{m} \right\rfloor + \cdots + \left\lfloor \frac{m-1+x}{m} \right\rfloor = \lfloor x \rfloor .$$

And as in Chapter 1, we find it useful to get more data by generalizing downwards to the case $n = 0$:

$$\left\lfloor \frac{x}{m} \right\rfloor + \left\lfloor \frac{x}{m} \right\rfloor + \cdots + \left\lfloor \frac{x}{m} \right\rfloor = m \left\lfloor \frac{x}{m} \right\rfloor .$$

Our problem has two parameters, m and n; let's look at some small cases for m. When $m = 1$ there's just a single term in the sum and its value is $\lfloor x \rfloor$. When $m = 2$ the sum is $\lfloor x/2 \rfloor + \lfloor (x+n)/2 \rfloor$. We can remove the interaction between x and n by removing n from inside the floor function, but to do that we must consider even and odd n separately. If n is even, $n/2$ is an integer, so we can remove it from the floor:

$$\left\lfloor \frac{x}{2} \right\rfloor + \left(\left\lfloor \frac{x}{2} \right\rfloor + \frac{n}{2} \right) = 2 \left\lfloor \frac{x}{2} \right\rfloor + \frac{n}{2} .$$

If n is odd, $(n-1)/2$ is an integer so we get

$$\left\lfloor \frac{x}{2} \right\rfloor + \left(\left\lfloor \frac{x+1}{2} \right\rfloor + \frac{n-1}{2} \right) = \lfloor x \rfloor + \frac{n-1}{2} .$$

The last step follows from (3.26) with $m = 2$.

Be forewarned: This is the beginning of a pattern, in that the last part of the chapter consists of the solution of some long, difficult problem, with little more motivation than curiosity.
—Students

Touché. But c'mon, gang, do you always need to be told about applications before you can get interested in something? This sum arises, for example, in the study of random number generation and testing. But mathematicians looked at it long before computers came along, because they found it natural to ask if there's a way to sum arithmetic progressions that have been "floored."
—Your instructor

Figure: 具体数学的某句话

一些推荐 2

If somebody said what advice would I give to a young person, they always ask that funny kind of a question. And I think one of the things that I would, that would sort of come first to me is this idea of, don't just believe that because something is trendy, that it's good.

D. E. Knuth

- My advice to young people——By D.E.Knuth
- Knuth 教授介绍

今天的话题

Fibonacci 数列

- 符号构造方法
- Fibonacci 数列的第 N 项计算和复杂度分析
- Fibonacci 数列的模周期性

数论话题

- 费马小定理
- Stern Brocot Tree

总结

Fibonacci 数列

- 递归定义: $F_n = F_{n-1} + F_{n-2}, n \geq 2$
- 初始条件: $F_0 = 0, F_1 = 1$
- 前 10 项: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34$
- 更多信息浏览: 维基百科 Fibonacci 数列

符号构造和生成函数

Q: 每次只能跨一步或者两步，有多少种不同的办法走完 N 步?

- $\mathcal{F}$ 是一个 *class*, 每个元素 p 是一套走步的方法, 每个元素 p 定义了 size, $|p|$ 为总步长
- 如 $p = 1, 2, 2$ 是走了三次，第一次走一步，后面两次都走了两步，$|p| = 5$
- $\mathcal{F}(z) = \sum_{p \in \mathcal{F}} z^{|p|} = \sum_{N \geq 0} F_N z^N$
- $\mathcal{F} = \emptyset + (\bullet + \bullet\bullet) \times \mathcal{F} \implies \mathcal{F}(z) = 1 + (z + z^2)\mathcal{F}(z)$
- $\mathcal{F}(z) = \frac{1}{1-z-z^2}$, 这个称为生成函数，包含了 F_N 的全部信息

直接递归

```
1  int fib(int n) {
2      if (n == 0 || n == 1)
3          return n;
4      return fib(n-1) + fib(n-2);
5  }
```

- 递归，且不是尾递归，编译器无法进行优化
- 时间复杂度是指数级别，空间复杂度呢？
- 严格分析时空复杂有助于我们理解计算模型
- 如何给上面的加上缓存进行优化呢?
- 为啥不能直接用公式计算？

矩阵乘法

- $\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$
- $\mathbf{Q}_n = \mathbf{A}\mathbf{Q}_{n-1} \implies \mathbf{Q}_n = \mathbf{A}^{n-1}\mathbf{Q}_1$
- 二分法计算 $\mathbf{A}^n$ 能达到的复杂度是对数级别的
- 精确的计算出，我们究竟需要多少次加法和乘法？
- 和之前的递推类比，在同一个机器上测试速度，并预测。

6 年前的故事

- 由于斐波那契数列是指数增长的，所以第 N 项会非常大，通常 online judge 系统会输出会要求取模（一般按素数取模，想想看为啥)
- 取模后状态有限，状态有限的马尔科夫序列，一定是周期的（抽屉原理）
- 那么周期是多少？这个需要一些数学技巧。6 年前，电子系的韩衍隽结合网络资料解决了这个问题

一些关于同余的知识

- $a \equiv b \pmod{m} \iff (a-b) = km, k \in \mathcal{Z}$
- $ab \equiv (a \mod m)(b \mod m) \pmod{m}$
- $a \equiv b \ \& \ c \equiv d \iff ac \equiv bd \pmod{m}$
- $ad \equiv bd \iff a \equiv b \pmod{m}, d \perp m$

关键技巧和思路

Good mathematicians see analogies between theorems; great mathematicians see analogies between analogies.

- 首先斐波那契数列数列的通项公式含有无理数，不好处理，想办法转换成整数的
- $F_n = \frac{1}{2^{n-1}} \sum_{k=0}^{\lfloor \frac{n-1}{2} \rfloor} \binom{n}{2k+1} 5^k$
- 这时候找规律非常重要了，编程打表，找规律，然后猜想
- 猜出来周期，然后试图证明 $(F_1, F_2) \equiv (F_{1+c}, F_{2+c}) \pmod{p}$
- 关键定理: Lucas 定理，高斯二次互反律和费马小定理

素数测试的工具

Fermat's Little Theorem

Given n is a prime number, $\forall a < n, a^n \equiv a \pmod{n}$.

- 如何证明费马小定理？研究 $n \mod m, 2n \mod m, \ldots, (m-1)n \mod m$
- 梅森数 $2^{2^k}+1$ 是素数么？比如 $2^{32}+1$
- 检测一个数字是否通过费马检测的复杂度是多少？
- 一个数字通过了费马检测，但是还不是素数的概率是？Carmichael 数，遇上它的概率小于宇宙射线让你程序出 bug 的概率（写一个程序找找看有哪些？）

写程序的建议: 构造过程的抽象

```
1   (define (expmod base exp m)
2     (cond ((= exp 0) 1)
3           ((even? exp)
4            (remainder (square (expmod base (/ exp 2) m))
5                       m))
6           (else
7            (remainder (* base (expmod base (- exp 1) m))
8                       m))))
9
10  (define (fermat-test n)
11    (define (try-it a)
12      (= (expmod a n n) a))
13    (try-it (+ 1 (random (- n 1)))))
14
15  (define (fast-prime? n times)
16    (cond ((= times 0) true)
17          ((fermat-test n) (fast-prime? n (- times 1)))
18          (else false)))
```

排列出所有的有理数

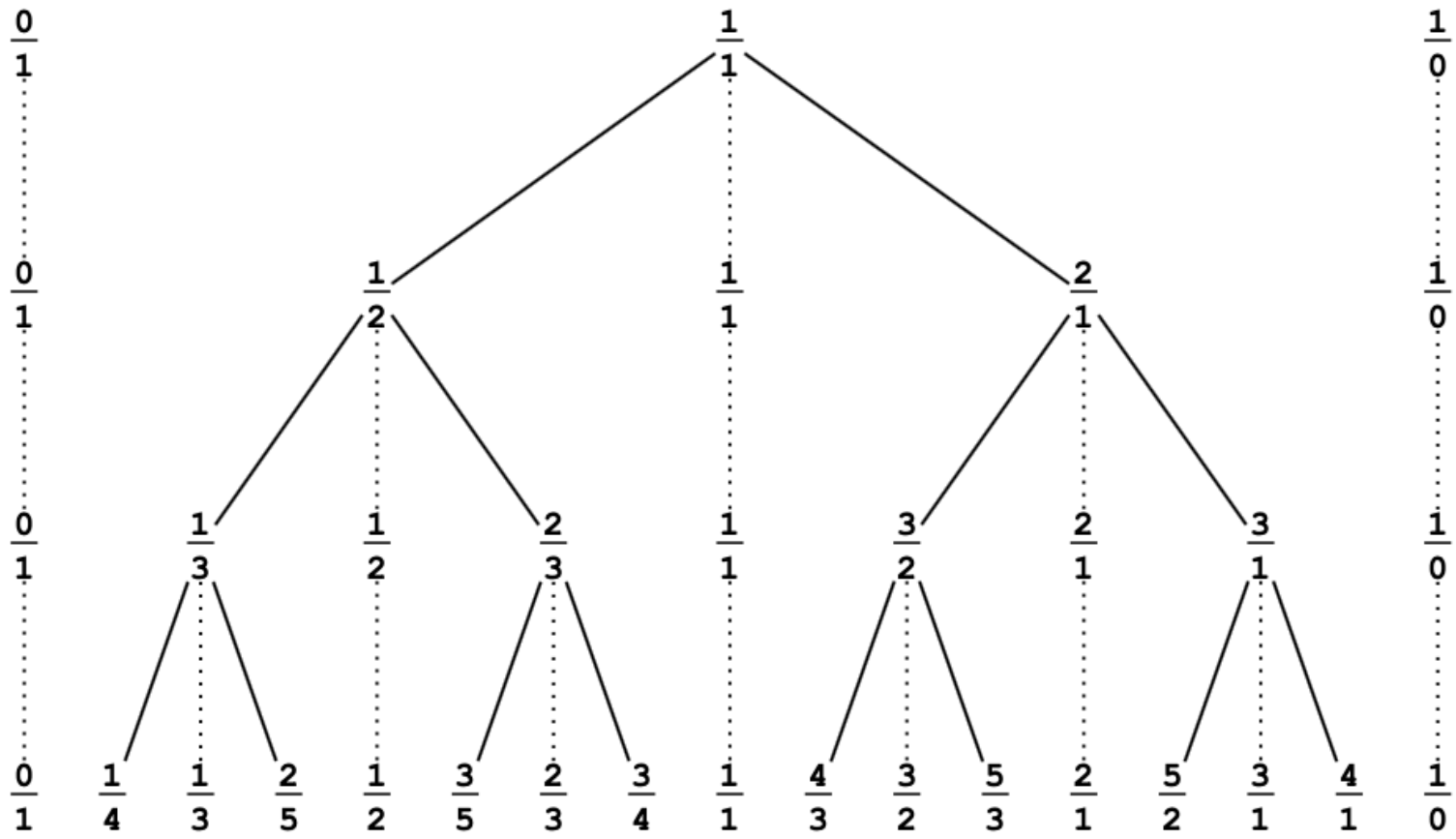

Figure: Stern–Brocot tree, 图选择维基百科

Stern Brocot Tree 的性质

- 作为列举有理数的一种算法，会有遗漏么?
- 会有重复的数出现么?
- 自然底数 e 的编码是 $RRLRRLRLLLLRLRRRRRRLRLLLLLLLLRLR\ldots$，有啥规律? (欧拉 24 岁发现了这个)
- 编写程序给定任意编码，可以输出有理数；或者给定某个实数，输出编码 (Google 的笔试题)
- 和连续分式的关系

参考资料

- 算法分析导论
- 具体数学
- 计算机程序构造和解释

结束

Q & A

谢谢！