作业：引论

2019 年 9 月 17 日

1　拿石子问题

1. **证明** n 为 Fibonacci 数时乙有必胜策略

   proof 数学归纳法

   设 Fibonacci 序列为 $\{F_n\}_{n=2}^{\infty}, F_2 = 2$

   (a) $n = 2$ 时，可能的取法只有一种，甲取 1 个，乙取 1 个，显然乙必胜。

   (b) 假设 $k \le n$ 时乙必胜，当 $k = n + 1$ 时，由 Fibonacci 数的性质，

   $$F_{n+1} = F_n + F_{n-1}.$$

   i. 若 $m^1_{甲} \ge F_{n-1}$，由于 $F_{n+1} = F_n + F_{n-1} < 3F_n$，$m^1_{乙} = F_{n+1} - m^1_{甲} < 2F_{n-1} < 2m^1_{甲}$，故此时乙可胜利

   ii. 若 $m^1_{甲} < F_{n-1}$，则可以将石子看做两堆，一堆 F_n 个（记作 A），一堆 F_{n-1} 个（记作 B），由于 $m^1_{甲} < F_{n-1}$，因此可以看作甲、乙两人先从 B 堆中开始拿。由归纳假设，此时乙有策略使乙恰好拿完 B 的最后一颗石子。

   下面说明在乙取完 B 堆的最后一颗石子之后，甲下次不可能取完 A 堆。设乙第 l 次时正好取完了 B 堆的最后一颗石子，则易知 $m^1_{乙} \le \frac{2}{3}F_{n-1}$，因此只需要比较 $\frac{4}{3}F_{n-1}$ 和 F_n 的大小即可, 相当于比较 $4F_{n-1}$ 和 $3F_n$ 的大小

   $$3F_n - 4F_{n-1} = 3F_{n-2} - F_{n-1} > 0.$$

   即 $F_n > \frac{4}{3}F_{n-1}$ ，因此，下次甲最多拿 $\frac{4}{3}F_{n-1}$ 个石子，不会取完 A 堆。

   这说明，在乙取完 B 堆后，问题转化为甲乙两从石子数为 F_n 重新开始游戏，由数学归纳法原理，乙有必胜策略。

2. **证明** n 不是 Fibonacci 数时，甲有必胜策略。不会

2　求递归式的渐近表示

1. $T(n) = T(n-1) + n$

solution:

$$\begin{aligned} T(n) - T(n-1) &= n \\ T(n-1) - T(n-2) &= n-1 \\ &\cdots \\ T(2) - T(1) &= 2 \\ &. \end{aligned}$$

将上式累加，有

$$T(n) = T(1) - 1 + \frac{(n+1)\,n}{2}.$$

因此，$T(n) = \Theta(n^2)$

2.
$$\begin{cases} T(1) = T(2) = 1 \\ T(n) = T(n-1) + T(n-2), n \geq 3 \end{cases}.$$

solution:

上述表达的特征方程为

$$x^2 = x + 1.$$

方程的解为

$$x_{1,2} = \frac{1 \pm \sqrt{2}}{2}.$$

因此，$T(n)$ 的表达式为

$$T(n) = a\left(\frac{1+\sqrt{2}}{2}\right)^n + b\left(\frac{1-\sqrt{2}}{2}\right)^n.$$

所以，$T(n)$ 的渐近表达式为 $T(n) = \Theta(2^n)$

3. $T(n) = T(\lfloor\sqrt{n}\rfloor) + 1$

solution:

设 $n = 2^k$ 则 $k = \log_2 n$ ，有

$$\begin{aligned} T(n) &= T\left(\sqrt{2^k}\right) + 1 \\ &= T\left(2^{k-1}\right) + 1 \\ &= T\left(2^{k-2}\right) + 2 \\ &\cdots \\ &= T\left(2^0\right) + k \\ &= T(1) + k \\ &= \log_2 n. \end{aligned}$$

因此，$T(n) = \Theta(\log n)$

4. $T\left(n\right)=\frac{n}{n-1}T\left(n-1\right)+1$

solution:

$$
\begin{aligned}
T\left(n\right) &= \frac{n}{n-1}T\left(n-1\right)+1\\
&= \frac{n}{n-2}T\left(n-2\right)+\frac{n}{n-1}+1\\
&= \dots\\
&= \frac{n}{1}T\left(1\right)+\frac{n}{2}+\dots+\frac{n}{n-1}+1\\
&= nT\left(1\right)+\sum_{i=2}^{n}\frac{1}{i-1}\\
&= nT\left(1\right)+n\ln n+\epsilon_n-1.
\end{aligned}
$$

其中 $\lim_{n\to\infty}\epsilon_n=\epsilon$ ， ϵ 为欧拉常数

因此 $T\left(n\right)=\Theta\left(n\log n\right)$

3 有序拆分问题

下面是用 python 语言实现的算法:

```python
import copy

def OrderedSplit(n):
    if n == 1:
        return [[1]]
    else:
        result = [[n]]
        for element in OrderedSplit(n-1):
            result.append([1] + element)
            result.append(element + [1])
        # 去重
        new_result = copy.deepcopy(result)
        for i in range(len(result)):
            for j in range(i+1, len(result)):
                if result[i] == result[j]:
                    new_result.remove(result[j])
        return new_result
```

分析：这个算法中使用到了递归，$n+1$ 时的输出是通过 n 时得到的。并且还包含了去重操作，设其时间复杂度为 $T\left(n\right)$，则有下面的式子成立

$$
\begin{cases}
T\left(1\right)=1\\
T\left(n\right)=T\left(n-1\right)+\Theta\left(n^2\right)
\end{cases}.
$$

其中 $\Theta(n^2)$ 是去重操作的时间复杂度。有

$$\begin{aligned}
T\left(n\right) &= T\left(n-1\right) + \Theta\left(n^2\right) \\
&= T\left(n-1\right) + cn^2 \\
&= \cdots \\
&= T\left(1\right) + c\sum_{i=2}^{n} i^2 \\
&= c\sum_{i=1}^{n} i^2 \\
&= c\frac{1}{6}n\left(n+1\right)\left(2n+1\right) \\
&= \Theta\left(n^3\right)
\end{aligned}.$$