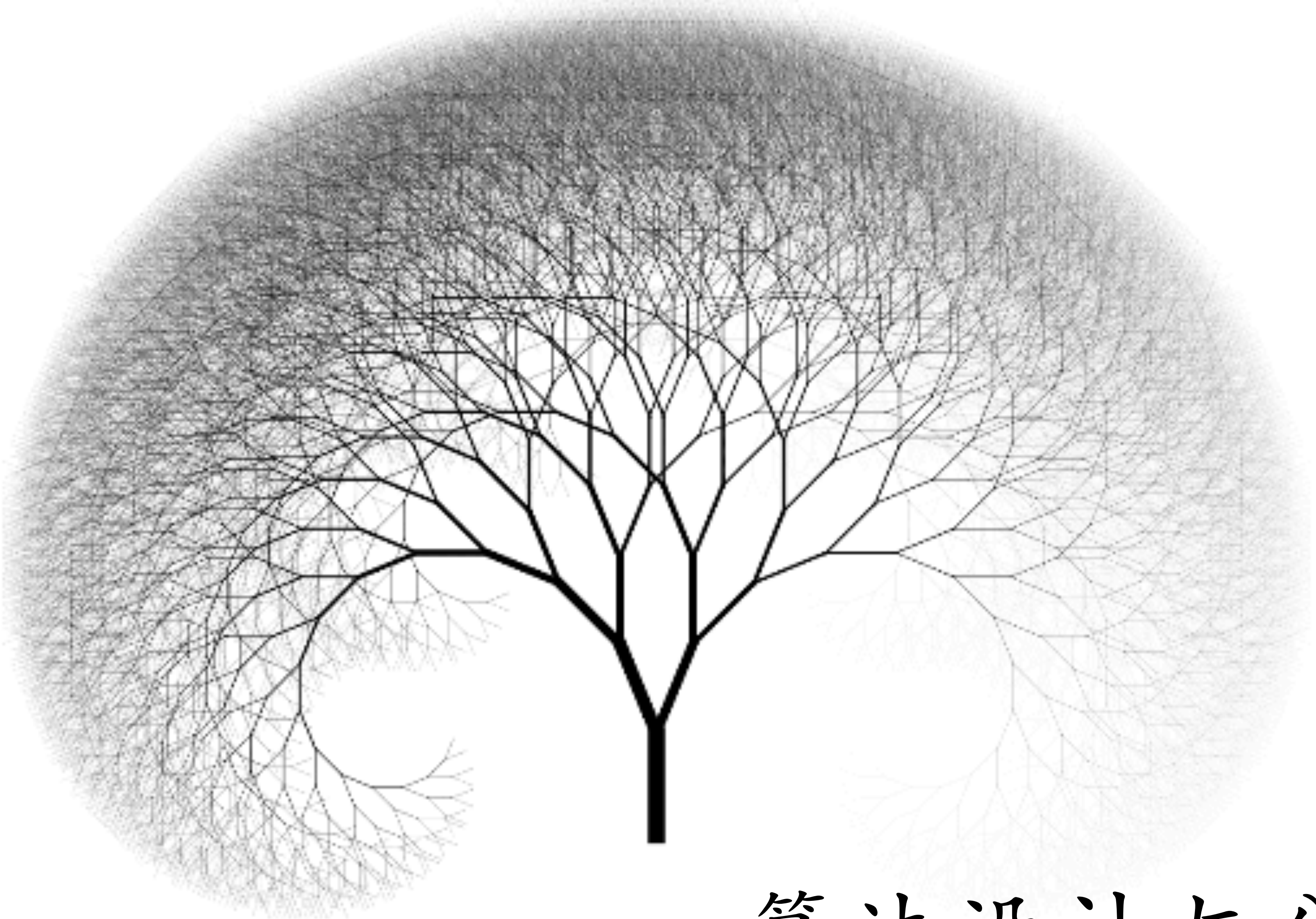


算法设计与分析

刘 安
苏州大学 计算机科学与技术学院
http://web.suda.edu.cn/anliu/

递归

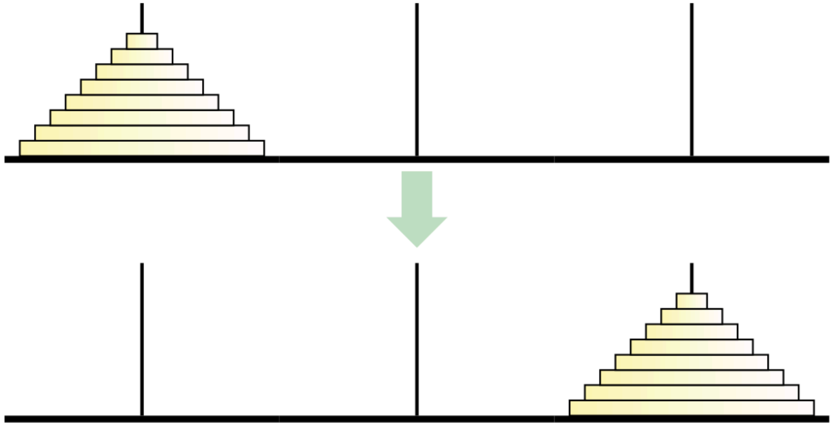
我想了解递归是什么

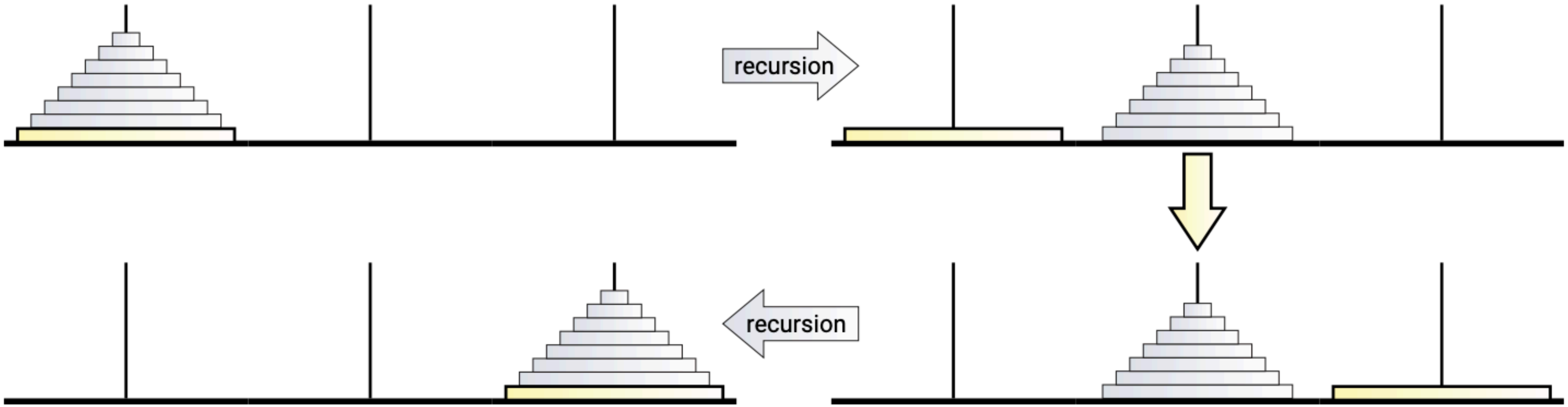
汉诺塔

Tower of Hanoi

汉诺塔

- 如何将n个圆盘从第一根柱子转移到第三根柱子，可以借助第二根柱子，但每次只能移动一个圆盘，且大圆盘不能放在小圆盘的上面





汉诺塔

- 如何将n个圆盘从第一根柱子转移到第三根柱子，可以借助第二根柱子，但每次只能移动一个圆盘，且大圆盘不能放在小圆盘的上面

```python
def hanoi(n, src, dst, tmp):
    if n > 0:
        hanoi(n - 1, src, tmp, dst)
        move(n, src, dst)
        hanoi(n - 1, tmp, dst, src)
```

- 令$T(n)$是转移n个圆盘所需的移动次数
  - $T(n) = 2T(n-1) + 1$
  - $T(0) = 0$
- $T(n) = 2^n - 1$
- $2^n - 1$也是最少移动次数（归纳法证明）

受限的汉诺塔

- 如何将n个圆盘从第一根柱子转移到第三根柱子，可以借助第二根柱子，但每次只能移动一个圆盘：要么在中间柱子上放置一个圆盘，要么从中间柱子上取走一个圆盘；且大圆盘不能放在小圆盘的上面
- 令$T(n)$是转移n个圆盘所需的移动次数
  - $T(n) = 3T(n-1) + 2$
  - $T(0) = 0$
- $T(n) = 3^n - 1$
- $3^n - 1$也是最少移动次数（归纳法证明）

Reve难题

- 如何将n个圆盘从第一根柱子转移到第四根柱子，可以借助第二和第三根柱子，但每次只能移动一个圆盘，且大圆盘不能放在小圆盘的上面。另外，移动次数应尽可能少

- $n = 1$时，至少移动1次，仅需要2个柱子
- $n = 2$时，至少移动3次，仅需要3个柱子
- $n = 3$时
  - 如果只有3根柱子，至少需要移动7次（经典汉诺塔结论）
  - 如果有4根柱子，仅需要移动5次

Reve难题

- 如何将n个圆盘从第一根柱子转移到第四根柱子，可以借助第二和第三根柱子，但每次只能移动一个圆盘，且大圆盘不能放在小圆盘的上面。另外，移动次数应尽可能少

- 将$n-k$个圆盘从第一根柱子移到第二根柱子，借助第三和第四根柱子
- 将剩下的k个圆盘从第一根柱子移到第四根柱子，借助第三根柱子 ← **经典汉诺塔**
- 将$n-k$个圆盘从第二根柱子移到第四根柱子，借助第一和第三根柱子

Frame-Stewart算法

选择最优的k使得移动次数最少

$$T_4(n) = \min_{1\le k<n} \{2T_4(n-k) + T_3(k)\}$$

Frame-Stewart算法分析

算法所需移动次数的递归关系：$T_4(n) = \min_{1 \le k < n} \{2T_4(n-k) + T_3(k)\}$

- 基本情况：$T_4(1) = 1$，$T_4(2) = 3$
- 根据递归关系计算，结果如下表

• 如何选择最优的k

- 满足$f(k) \le n$的最大整数k

令 $f(j) = \sum_{i=1}^{j} i$

			$f(2)$			$f(3)$				$f(4)$					$f(5)$	
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k	NA	NA	2	2	2	3	3	3	3	4	4	4	4	4	5	5
$T_4(n)$	1	3	5	9	13	17	25	33	41	49	65	81	97	113	129	161

Frame-Stewart算法分析

算法所需移动次数的递归关系：$T_4(n) = \min\limits_{1 \le k < n} \{2T_4(n-k) + T_3(k)\}$

- 如何求解$T_4(n)$?
    - 计算$T_4(n) - T_4(n-1)$
    - 令$r = n - f(k)$，猜测$T_4(n) = \sum\limits_{i=1}^{k} i2^{i-1} + r2^k = (k+r-1)2^k + 1$

令$f(j) = \sum\limits_{i=1}^{j} i$

			$f(2)$			$f(3)$				$f(4)$					$f(5)$	
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k	NA	NA	2	2	2	3	3	3	3	4	4	4	4	4	5	5
$T_4(n)$	1	3	5	9	13	17	25	33	41	49	65	81	97	113	129	161

2 2 4 4 4 8 8 8 8 16 16 16 16 16 32

Frame-Stewart算法分析

- Frame-Stewart算法转移n个圆盘所需的移动次数$T_4(n) = 2T_4(n-k) + T_3(n)$

最优的k已经确定

- 命题：$T_4(n) = (k + r - 1)2^k + 1$
- 归纳法证明
  - 基本步骤：$n = 3$，$T_4(3) = (2 + 0 - 1)2^2 + 1 = 5$
  - 归纳假设：命题对于值3, 4, $\cdots$, $n - 1$都成立
  - 归纳步骤：考虑命题对于值n是否成立
    - 注意k, r和n的关系：$n = 1 + 2 + \cdots + k + r$且$r < k + 1$
    - 下面考虑k', r'和$n' = n - k$的关系：$n - k = 1 + 2 + \cdots + (k - 1) + r$
      - 如果$r = k$，那么$k' = k$且$r' = 0$
      - 如果$r < k$，那么$k' = k - 1$且$r' = r$

$k = r$

$$T_4(n) = 2T_4(n-k) + T_3(k) = 2(k + 0 - 1)2^k + 2 + 2^k - 1 = (k + r - 1)2^k + 1$$

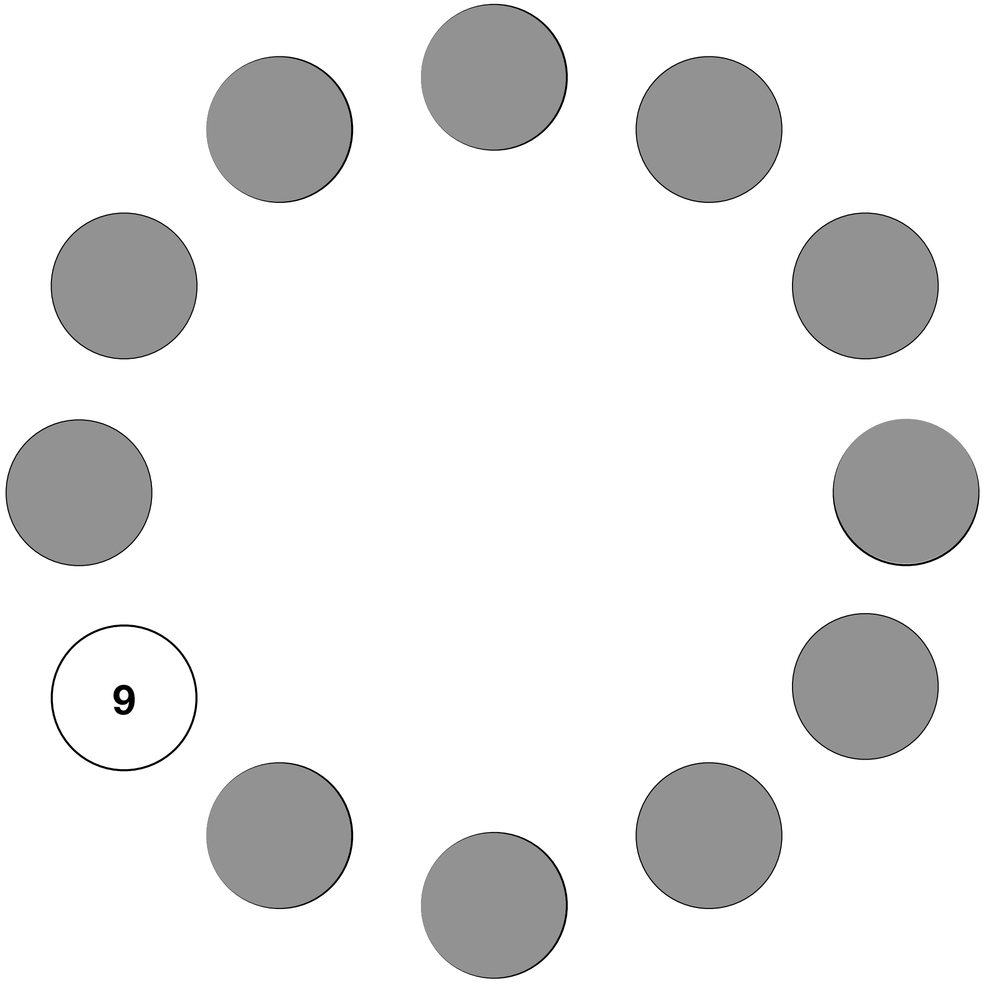
$$T_4(n) = 2T_4(n-k) + T_3(k) = 2(k - 1 + r - 1)2^{k-1} + 2 + 2^k - 1 = (k + r - 1)2^k + 1$$

约瑟夫问题

The Josephus Problem

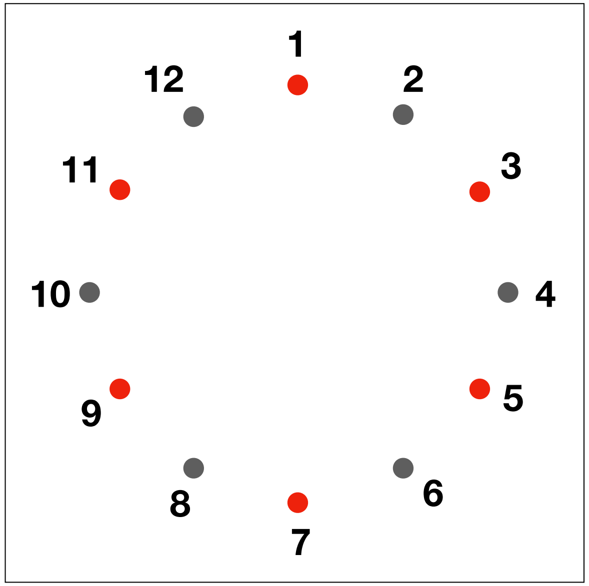
约瑟夫问题

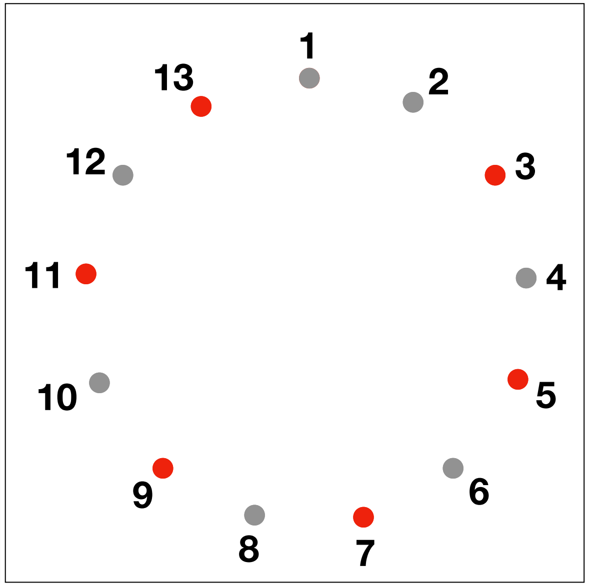
- n个人（编号从1到n）围成一圈，从1号开始报数，每次报到偶数的人将被淘汰，应该站在几号位置才能成为最后留下的人?



约瑟夫问题

- n个人（编号从1到n）围成一圈，从1号开始报数，每次报到偶数的人将被淘汰，应该站在几号位置才能成为最后留下的人?
- 令$J(n)$为最后留下的人所在的位置
  - 当$n=2k$时
    - $J(2k)=2J(k)-1$
  - 当$n=2k+1$时
    - $J(2k+1)=2J(k)+1$
- $J(1)=1$





- 猜测：$J(2^p+i)=2i+1$，$i=0,1,\cdots,2^p-1$，$p=0,1,2,\cdots$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$J(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15

约瑟夫问题

- 令$J(n)$为最后留下的人所在的位置　终极方案：将n的二进制数循环左移一位即可！
  - $J(2k) = 2J(k) - 1$
  - $J(2k+1) = 2J(k) + 1$
- 命题：$J(2^p + i) = 2i + 1$，$i = 0,1,\cdots,2^p - 1$，$p = 0,1,2,\cdots$
- 归纳法证明
  - 基本步骤：$p = 0$，此时$i = 0$，$J(1) = 2 \times 0 + 1 = 1$
  - 归纳假设：假设对于$0,1,\cdots,p$和任意$i = 0,1,\cdots,2^p - 1$命题成立
  - 归纳步骤：考虑命题对于$p+1$是否成立
    - 如果$i = 2j$，其中$0 \le j < 2^p$

$$J(2^{p+1} + i) = J(2^{p+1} + 2j) = J(2(2^p + j)) = 2J(2^p + j) - 1 = 2(2j + 1) - 1 = 2i + 1$$

    - 如果$i = 2j + 1$，其中$0 \le j < 2^p$

$$J(2^{p+1} + i) = J(2^{p+1} + 2j + 1) = J(2(2^p + j) + 1) = 2J(2^p + j) + 1 = 2(2j + 1) + 1 = 2i + 1$$

九连环

Chinese Ring Puzzle

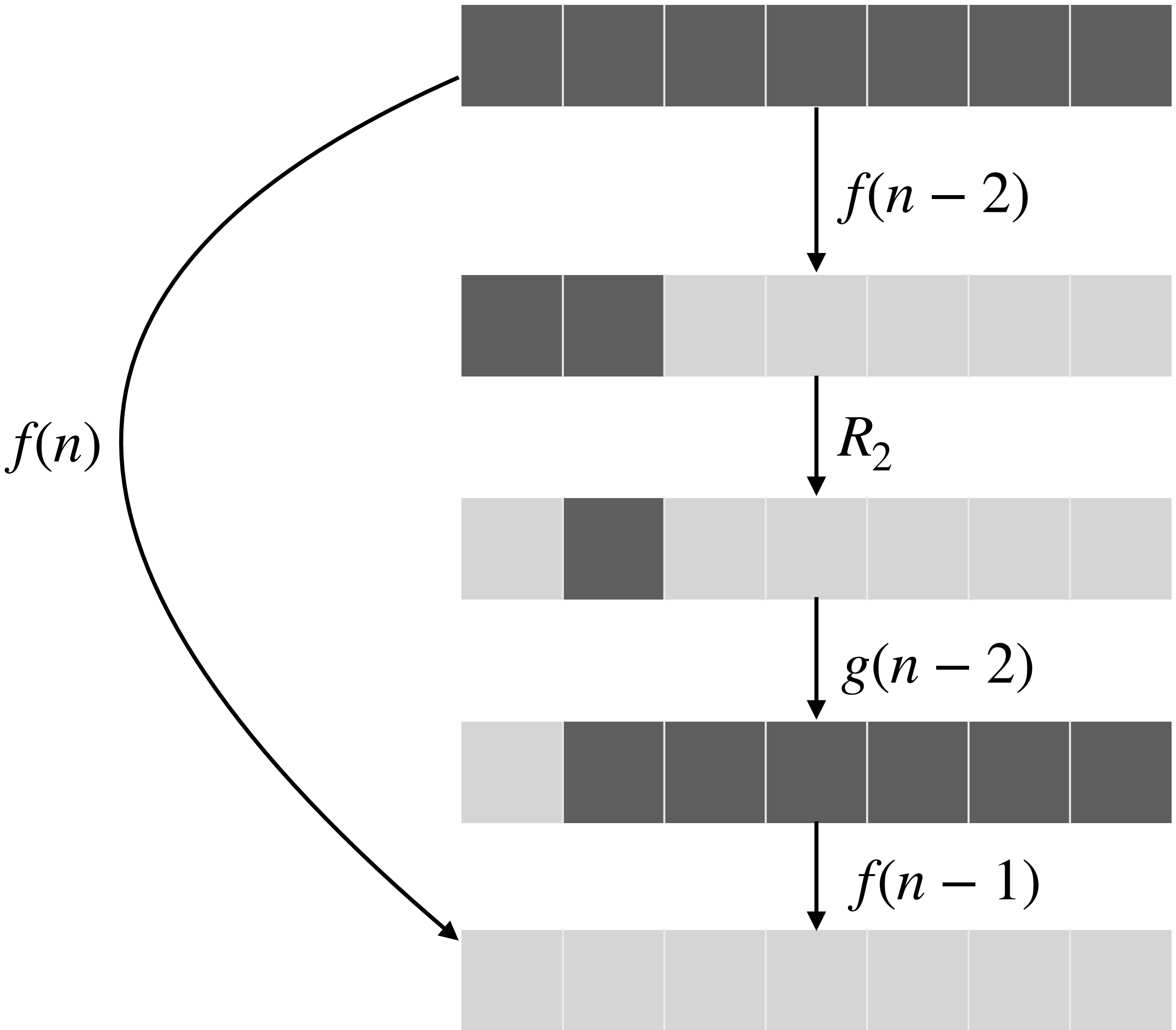
九连环

- 九连环是中国传统民间智力玩具，九个圆环相连成串套在架上，解开为胜
- 抽象模型：将每个环看成一个比特（架上为1，架下为0）
    - 问题：将一串n个1转换成一串n个0
    - R_1：始终可以翻转第1个位（即最右边的位）
    - R_2：如果比特串恰好以k个0结尾，那么可以翻转第$k+2$个位
- 当$n=4$时，转换过程如下

$$1111 \xrightarrow{R_2} 1101 \xrightarrow{R_1} 1100 \xrightarrow{R_2} 0100 \xrightarrow{R_1} 0101 \xrightarrow{R_2} 0111 \xrightarrow{R_1} 0110 \xrightarrow{R_2} 0010 \xrightarrow{R_1} 0011 \xrightarrow{R_2} 0001 \xrightarrow{R_1} 0000$$

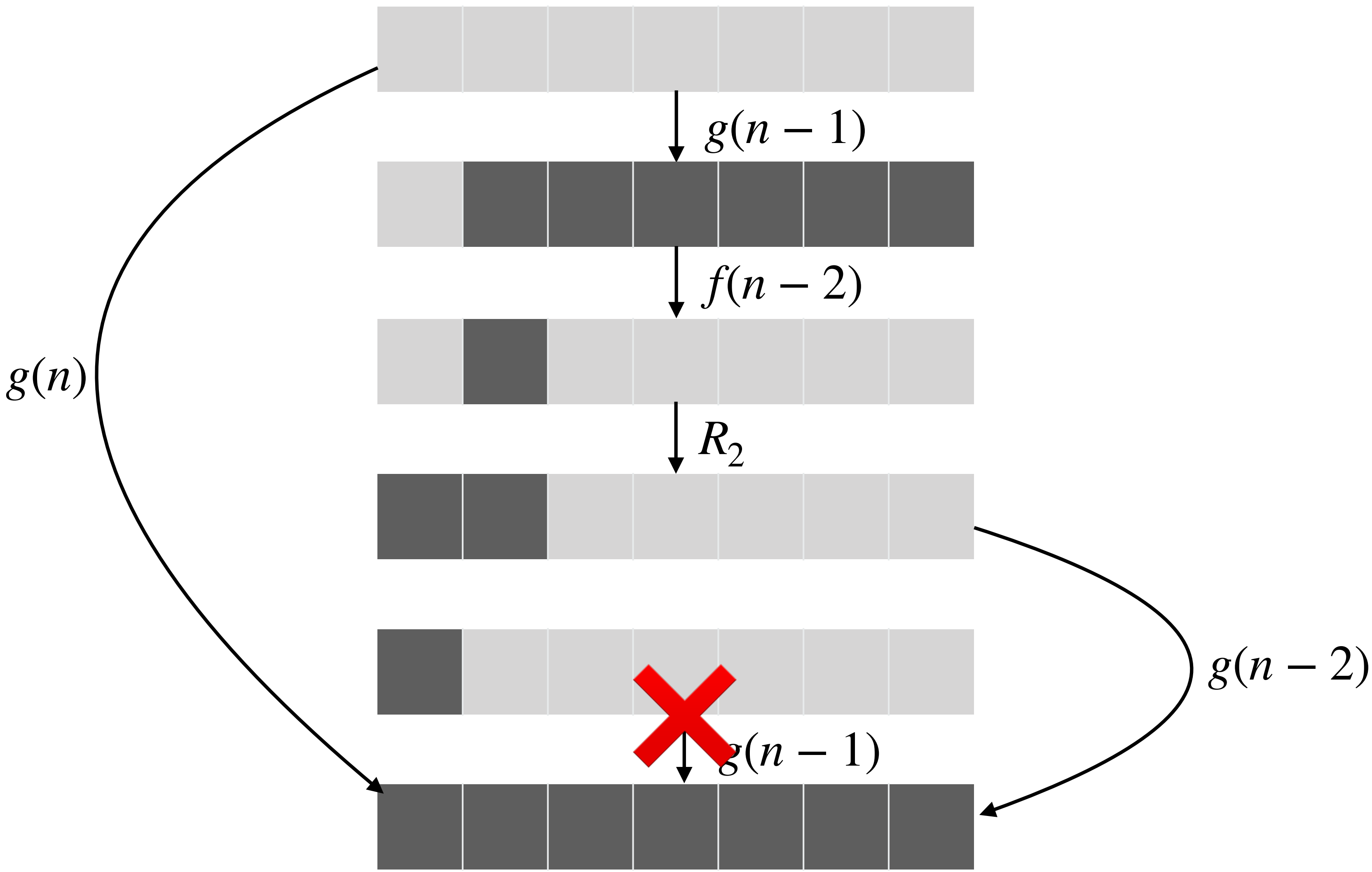
九连环的递归解法

- $f(n)$：将长度为n的串$11\cdots1$转换成长度为n的串$00\cdots0$
- $g(n)$：将长度为n的串$00\cdots0$转换成长度为n的串$11\cdots1$



九连环的递归解法

- $f(n)$：将长度为n的串$11\cdots1$转换成长度为n的串$00\cdots0$
- $g(n)$：将长度为n的串$00\cdots0$转换成长度为n的串$11\cdots1$



递归算法的时间复杂度

- 令$F(n)$，$G(n)$分别是算法$f(n)$，$g(n)$的运行时间

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ F(n-1) + F(n-2) + G(n-2) + 1 & \text{if } n > 2 \end{cases}$$

$$G(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ G(n-1) + G(n-2) + F(n-2) + 1 & \text{if } n > 2 \end{cases}$$

- $f(n)$翻转序列的逆序正是$g(n)$的翻转序列，所以$F(n) = G(n)$
- $F(n) = F(n-1) + 2F(n-2) + 1$，$n > 2$

$$\Rightarrow F(n) = \frac{2}{3}2^n - \frac{1}{2} - \frac{1}{6}(-1)^n$$