归纳与递归

离散数学—归纳与递归

内容提要

- 归纳
  - 数学归纳法
  - 强数学归纳法
  - 运用良序公理来证明
- 递归
  - 递归定义
  - 结构归纳法
  - 递归算法

数学归纳法

- 证明目标
  - $\forall n\, P(n)$　//n的论域为正整数集合
- 证明框架
  - 基础步骤：$P(1)$为真
  - 归纳步骤：证明$\forall k\,(P(k)\rightarrow P(k+1))$
    - //对任意正整数k，给出 $P(k) \vdash P(k+1)$ 的论证步骤.
    - …
  - 因此，对任意正整数n, $P(n)$ 成立. // 即：$\forall n\, P(n)$

数学归纳法（有效性）

- 良序公理
  - 正整数集合的非空子集都有一个最小元素
- 数学归纳法的有效性（归谬法）
  - 假设$\forall n\ P(n)$不成立，则 $\exists n\ (\neg P(n))$成立.
  - 令$S=\{ n\in Z^+ \mid \neg P(n)\}$，S是非空子集.
  - 根据良序公理，S有最小元素，记为m, $m\neq 1$
  - $(m-1)\notin S$, 即$P(m-1)$成立.
  - 根据归纳步骤，$P(m)$成立，即$m\notin S$，矛盾.
  - **因此**，$\forall n\ P(n)$成立.

数学归纳法（举例）

- $H_k=1+1/2+\ldots+1/k$ （k为正整数）
- 证明：$H_{2^n} \geq 1+n/2$ （n为正整数）
  - 基础步骤：$P(1)$为真， $H_2=1+1/2$
  - 归纳步骤：对任意正整数k, $P(k) \Rightarrow P(k+1)$.

$$H_{2^{k+1}} = H_{2^k} +1/(2^k+1)+\ldots+1/2^{k+1}$$

$$\geq(1+k/2)+2^k(1/2^{k+1}) =1+(1+k)/2$$

  - **因此**，对任意正整数n, $P(n)$ 成立.

数学归纳法（举例）

- 猜测前n个奇数的求和公式，并证明之。
  - 1=1
  - 1+3=4
  - 1+3+5=9
  - 1+3+5+7=16
  - …
  - $1+3+\ldots+(2n-1)=n^2$（n为正整数）
  - 运用数学归纳法证明（练习）

数学归纳法证明时常见错误

- 例1：任意n个人，他们一定全部在同一天出生.
- 错误证明：
  - Basis： 当$n=1$时，只有一个人，命题显然成立；
  - I.H.：假设任意k个人，他们全部在同一天出生，则：
  - I.S.： 当有$k+1$个人时（编号为$1,2,\cdots,k,k+1$），根据归纳假设，第1人至第k人（共k个人）一定在同一天出生；第2至第 $k+1$ 人（共 k 个人）也一定在同一天出生。因此，这$k+1$人全部在同一天出生。根据数学归纳法，命题成立. □
  - 归纳基础错误：$P(1) \nrightarrow P(2)$!

数学归纳法证明时常见错误

- 例2：证明 $\sum_{i=1}^{n} 2i-1=n^2$
- 错误证明：
  - Basis：当$n=1$时，$\sum_{i=1}^{1} 2i-1=1^2$命题成立；
  - I.H.：假设当$n=k$时$\sum_{i=1}^{k} 2i-1=k^2$ 成立，则：
  - I.S.：根据等差数列的求和公式，$\sum_{i=1}^{k+1} 2i-1=1+3+5+\cdots+2(k+1)-1=\frac{[1+2(k+1)-1](k+1)}{2}=(k+1)^2$。

    根据数学归纳法，命题成立. □
  - 归纳过程错误：未证明$P(k)\rightarrow P(k+1)$!

强数学归纳法

- 证明目标
  - $\forall n\, P(n)$　//n的论域为正整数集合
- 证明框架
  - 基础步骤：$P(1)$为真
  - 归纳步骤：证明$\forall k\, (P(1)\wedge\ldots\wedge P(k)\rightarrow P(k+1))$
    - //对任意正整数k，给出 $P(1), \ldots, P(k) \vdash P(k+1)$ 的论证步骤.
    - …
  - 因此，对任意正整数n, $P(n)$ 成立. // 即：$\forall n\, P(n)$

强数学归纳法（一般形式）

- 设$P(n)$是与整数n有关的陈述， a和b是两个给定的整数，且$a \le b$.
- 如果能够证明下列陈述
  - $P(a), P(a+1), \ldots, P(b)$.
  - 对任意$k \ge b$, $P(a) \wedge \ldots \wedge P(k) \rightarrow P(k+1)$
- 则下列陈述成立
  - 对任意$n \ge a$, $P(n)$.

强数学归纳法（有效性）

- $\{ n \in Z \mid n \geq a \}$是良序的
  - 良序集：该集合的非空子集都有一个最小元素
- 数学归纳法的有效性（归谬法）
  - 假设$\forall n\ P(n)$不成立，则 $\exists n\ (\neg P(n))$成立.
  - 令$S=\{ n \in Z \mid (n \geq a) \wedge \neg P(n) \}$，S是非空子集.
  - 根据良序公理，S有最小元素，记为m, $m>a$
  - $a, \ldots, (m-1) \notin S$, 即$P(a), \ldots, P(m-1)$成立.
  - 根据归纳步骤，$P(m)$成立，即$m \notin S$，矛盾.
  - **因此**，$\forall n\ P(n)$成立.

强数学归纳法（举例）

- 任意整数n(n ≥2)可分解为（若干个）素数的乘积
  - n = 2.
  - 考察 n+1.
- 用4分和5分就可以组成12分及以上的每种邮资.
  - $P(12)$, $P(13)$, $P(14)$, $P(15)$.
  - 对任意$k \geq 15$, $P(12) \wedge \ldots \wedge P(k) \rightarrow P(k+1)$

（强）数学归纳法（举例）

- 对每个正整数$n \geq 4$，$n! > 2^n$
  - 基础步骤：$P(4)$为真，$24 > 16$
  - 归纳步骤：对任意正整数$k \geq 4$, $P(k) \Rightarrow P(k+1)$.

    $(k+1)! = (k+1)\, k! > (k+1)\, 2^k > 2^{k+1}$
  - **因此**，对任意正整数$n \geq 4$, $P(n)$ 成立.

运用良序公理来证明（举例）

- 设a是整数, d是正整数, 则存在唯一的整数q和r满足
  - $0 \le r < d$
  - $a = dq + r$
- 证明
  - 令$S=\{a-dq \mid 0 \le a-dq，q \in Z\}$，S非空.
  - 非负整数集合具有良序性
  - S有最小元，记为$r_0 = a-dq_0$.
  - 可证 $0 \le r_0 < d$
  - 唯一性证明， $0 \le r_1 - r_0 = d(q_0-q_1) < d$，因此，$q_1=q_0$

运用良序公理来证明（举例）

- 在循环赛胜果图中，若存在长度为m（$m \geq 3$）的回路，则必定存在长度为3的回路。

  备注：$a_i \rightarrow a_j$表示a_i赢了a_j

证明

- 设最短回路的长度为k //良序公理的保证
- 假设 $k>3$
- $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \ldots \rightarrow a_k \rightarrow a_1$
- 若$a_3 \rightarrow a_1$, 存在长度为3的回路，矛盾。
- 若$a_1 \rightarrow a_3$, 存在长度为$(k-1)$的回路，矛盾。

递归定义（N上的函数）

- 递归地定义自然数集合N上的函数。
  - 基础步骤：指定这个函数在0处的值；
  - 递归步骤：给出从较小处的值来求出当前的值之规则。
- 举例，阶乘函数 $F(n)=n!$ 的递归定义
  - $F(0)=1$
  - $F(n)=n\cdot F(n-1)$ for $n>0$

Fibonacci 序列

- Fibonacci 序列 $\{f_n\}$ 定义如下
  - $f_0 = 0$,
  - $f_1 = 1$,
  - $f_n = f_{n-1} + f_{n-2}$, 对任意$n \geq 2$.
- 其前几个数
  - 0, 1, 1, 2, 3, 5, 8, …
- 证明：对对任意$n \geq 0$，$f_n = \dfrac{\alpha^n - \beta^n}{\alpha - \beta}$

其中，$\alpha = \dfrac{1+\sqrt{5}}{2}, \beta = \dfrac{1-\sqrt{5}}{2}.$

归纳证明: Fibonacci 序列

- 验证：当n=0,1时，陈述正确。
- 对于k+1，$f_{k+1} = f_k + f_{k-1}$

$$= \frac{\alpha^k - \beta^k}{\alpha - \beta} + \frac{\alpha^{k-1} - \beta^{k-1}}{\alpha - \beta}$$

$$= \frac{\left(\alpha^k + \alpha^{k-1}\right) - \left(\beta^k + \beta^{k-1}\right)}{\alpha - \beta}$$

$$= \frac{\alpha^{k+1} - \beta^{k+1}}{\alpha - \beta}.$$

注意：$\alpha^2 = \alpha + 1$，且$\alpha^{n+1} = \alpha^n + \alpha^{n-1}$对任意$n \geq 1$.

递归定义（集合）

- 递归地定义集合。
  - 基础步骤：指定一些初始元素；
  - 递归步骤：给出从集合中的元素来构造新元素之规则；
  - 排斥规则（只包含上述步骤生成的那些元素）默认成立
- 举例，正整数集合的子集S
  - $x \in S$
  - 若$x \in S$且$y \in S$，则 $x+y \in S$。

递归定义（举例）

- 字母表Σ上的字符串集合Σ^*。
  - 基础步骤：$\lambda \in \Sigma^*$（λ表示空串）；
  - 递归步骤：若$\omega \in \Sigma^*$且$x \in \Sigma$，则$\omega x \in \Sigma^*$。
- 字符串的长度（Σ^*上的函数l）。
  - 基础步骤：$l(\lambda)=0$;
  - 递归步骤：$l(\omega x) = l(\omega) + 1$, 若$\omega \in \Sigma^*$且$x \in \Sigma$

递归定义（举例）

- Σ^*上的字符串连接运算。(Concatenation)
  - 基础步骤：若$\omega \in \Sigma^*$，则 $\omega \cdot \lambda = \omega$;
  - 递归步骤：若$\omega_1 \in \Sigma^*$ 且$\omega_2 \in \Sigma^*$ 以及$x \in \Sigma$，

    则 $\omega_1 \cdot (\omega_2\, x) = (\omega_1 \cdot \omega_2)\, x$ 。

    // $\omega_1 \cdot \omega_2$通常也写成$\omega_1\, \omega_2$

递归定义（举例）

- 复合命题的合式公式。
  - 基础步骤：T, F, s都是合式公式，其中s是命题变元;
  - 递归步骤：若E和F是合式公式，则 $(\neg E)$、$(E\wedge F)$、 $(E\vee F)$、 $(E\rightarrow F)$和$(E\leftrightarrow F)$都是合式公式。

结构归纳法

- 关于递归定义的集合的命题，进行结构归纳证明。
  - 基础步骤：证明对于初始元素来说，命题成立;
  - 递归步骤：针对生产新元素的规则，若相关元素满足命题，则新元素也满足命题
- 结构归纳法的有效性源于自然数上的数学归纳法
  - 第0步（基础步骤），…

结构归纳法（举例）

- $l(xy) = l(x) + l(y)$, x和y属于 Σ^* 。
- 证明
  - 设$P(y)$表示：每当x属于 Σ^*，就有$l(xy) = l(x) + l(y)$ 。
  - 基础步骤：每当x属于 Σ^*，就有$l(x\lambda) = l(x) + l(\lambda)$ 。
  - 递归步骤：假设$P(y)$为真，a属于 Σ, 要证$P(ya)$为真。
    - 即：每当x属于 Σ^*，就有$l(xya) = l(x) + l(ya)$
    - $P(y)$为真，$l(xy) = l(x) + l(y)$
    - $l(xya)= l(xy)+1= l(x) + l(y)+1=l(x) + l(ya)$

广义结构归纳法（举例）

- N×N是良序的（字典序）
- 递归定义$a_{m,n}$
  - $a_{0,0} = 0$
  - $a_{m,n} = a_{m-1,n} + 1 \quad (n=0,\ m>0)$
  - $a_{m,n} = a_{m,n-1} + n \quad (n>0)$
- 归纳证明 $a_{m,n} = m+n(n+1)/2$

0	**1**	**3**
1	**2**	**4**
2	**3**	**5**

递归算法

- 举例：欧几里德算法

```
function gcd(a, b)  // a≥b≥0, a>0
  if b=0
        return a
 else
        return gcd(b, a mod b)
```

- 递归算法的正确性
- 递归算法的复杂性（时间、空间）

递归与迭代

- $n!$
- $fibonacci(n)$

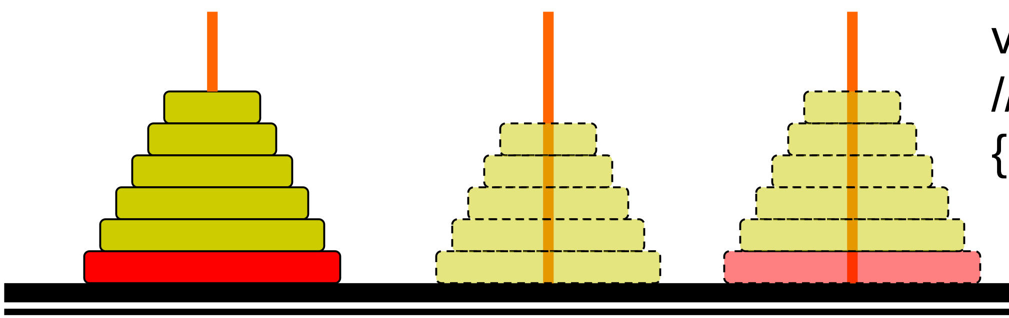
- 正确性如何保证？

作业

- 见课程QQ群

递归思维：例 1

- 汉诺塔问题: How many moves are need to move all the disks to the third peg by moving only one at a time and never placing a disk on top of a smaller one.



$$T(1) = 1$$
$$T(n) = 2T(n-1) + 1$$

```
void hanoi(int n,char one, two, three)
// 将n个盘从one座借助two座,移到three座
{
   void move(char x,char y);
   if(n==1)  then move(one,three);
      else  {
          hanoi(n-1,one,three,two);
          move(one,three);
          hanoi(n-1,two,one,three);
       }
}
```

汉诺塔问题的解

$$T(n) = 2T(n-1) + 1$$

$$2T(n-1) = 4T(n-2) + 2$$

$$4T(n-2) = 8T(n-3) + 4$$

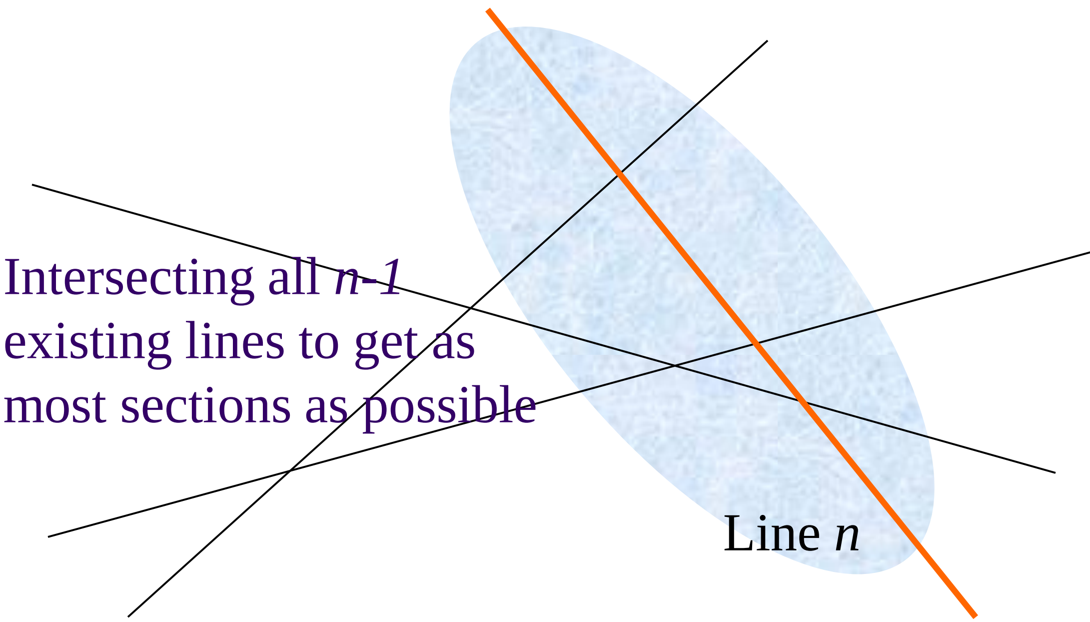
$$\ldots\ldots$$

$$2^{n-2}T(2) = 2^{n-1}T(1) + 2^{n-2}$$

$$\Rightarrow \quad \boldsymbol{T(n) = 2^n - 1}$$

递归思维：例 2

- Cutting the plane
  - How many sections can be generated at most by *n* straight lines with infinite length？



$L(0) = 1$

$L(n) = L(n-1) + n$

Solution of Cutting the Plane

$$
\begin{aligned}
L(n) &= L(n-1)+n \\
&= L(n-2)+(n-1)+n \\
&= L(n-3)+(n-2)+(n-1)+n \\
&= \ldots\ldots \\
&= L(0)+1+2+\ldots\ldots+(n-2)+(n-1)+n
\end{aligned}
$$

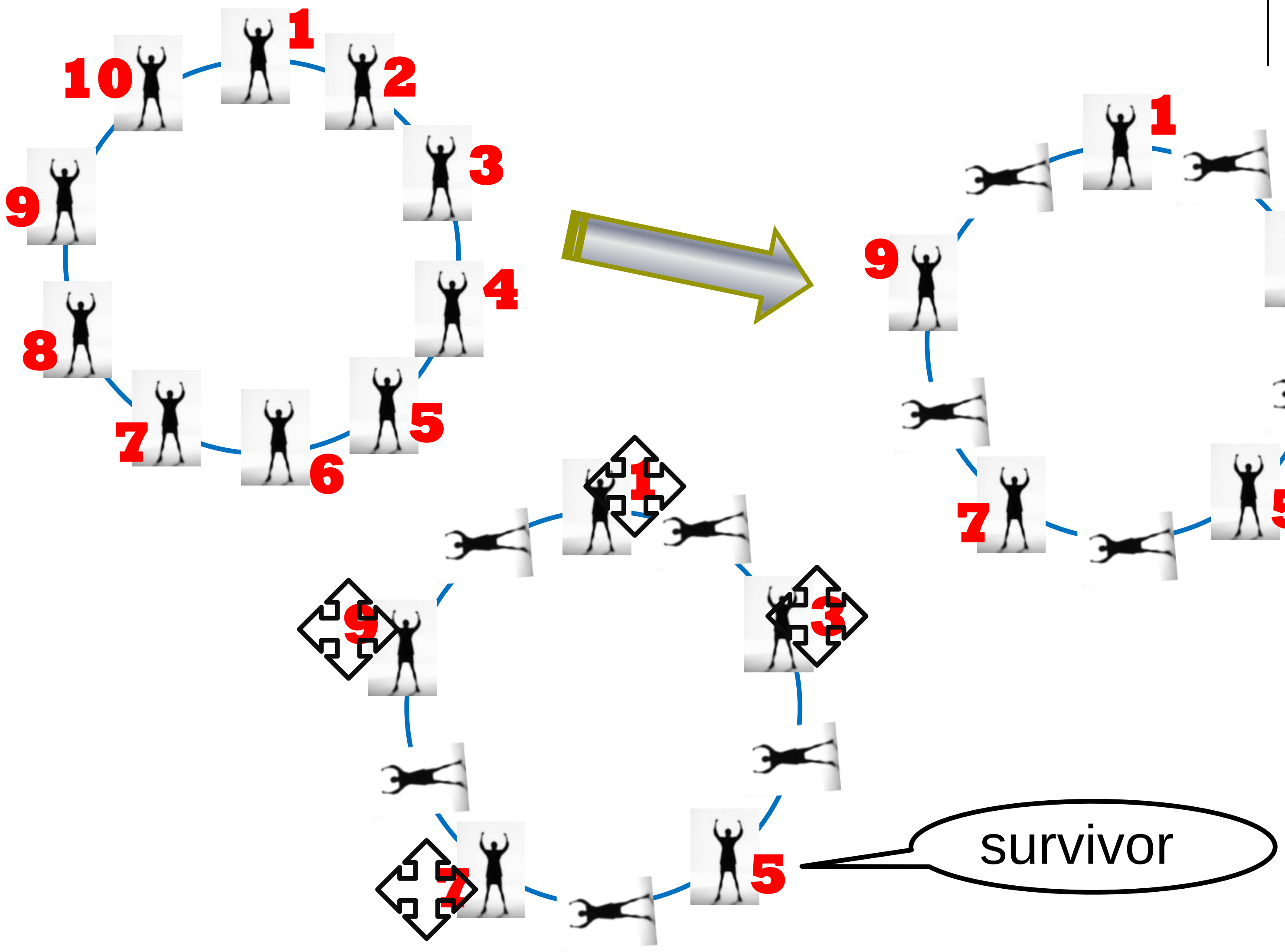
$$\boxed{L(n) = n(n+1)/2 + 1}$$

递归思维：例 3
Josephus Problem

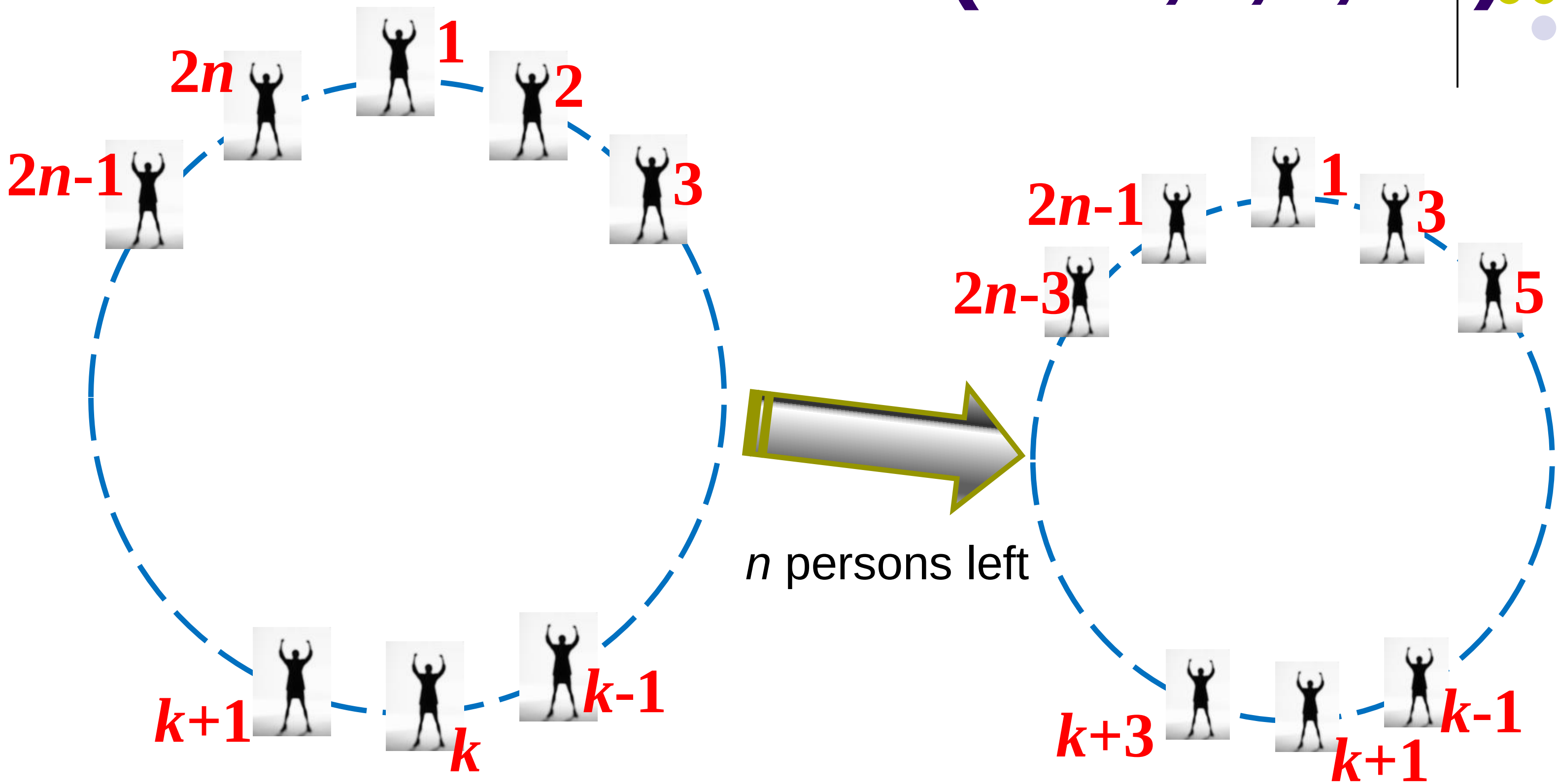
- Live or die, it’s a problem!
- Legend has it that Josephus wouldn't have lived to become famous without his mathematical talents. During the Jewish Roman war, he was among a band of 41 Jewish rebels trapped in a cave by the Romans. Preferring suicide to capture, the rebels decided to form a circle and, proceeding around it, to kill every third remaining person until no one was left. But Josephus, along with an unindicted co-conspirator wanted none of this suicide nonsense; so he quickly cal ated where he and his friend should stand in the vicio cle.

> We use a simpler version: “every second...”

Make a Try: for *n*=10



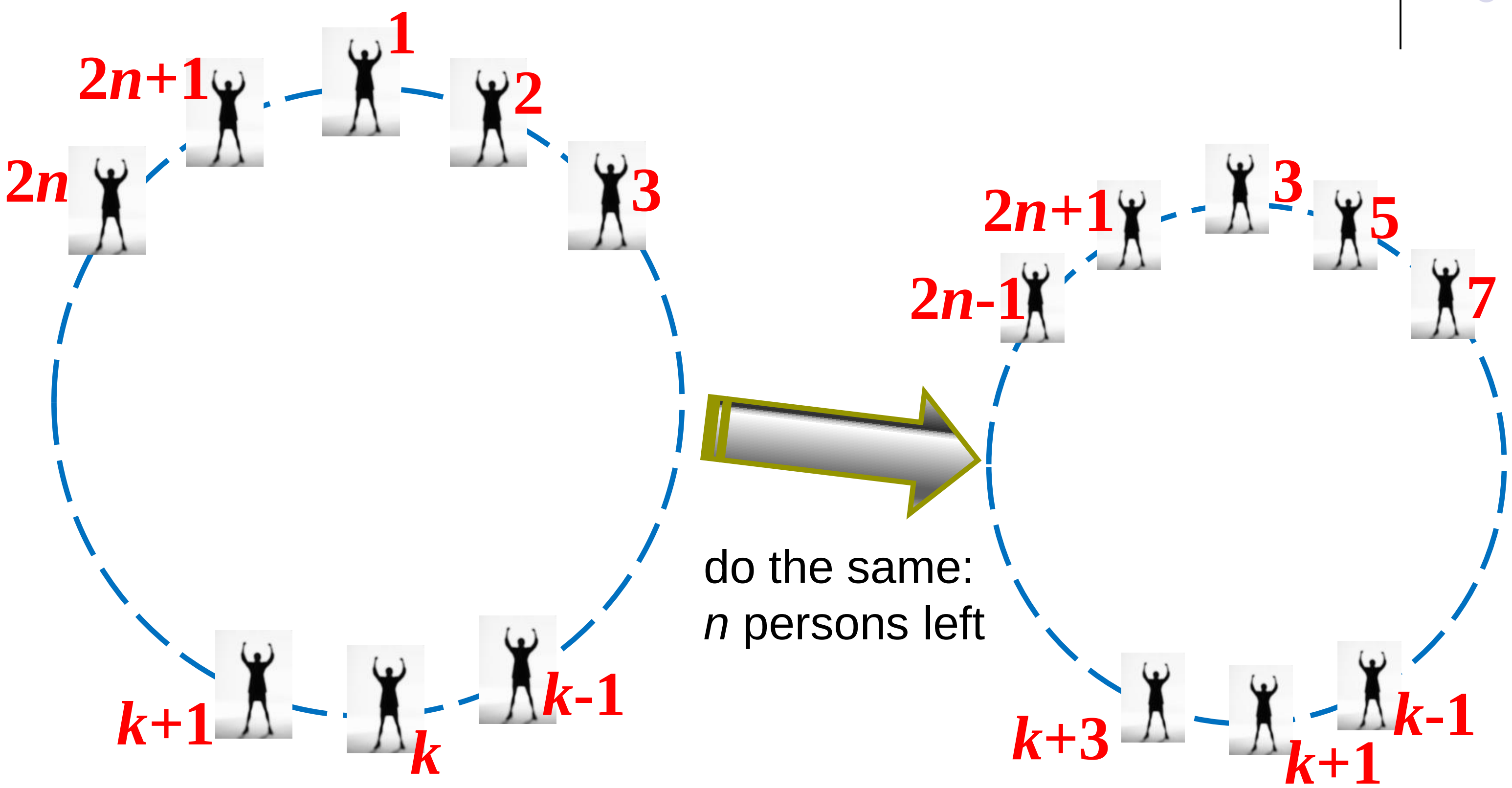
For 2*n* Persons (*n*=1,2,3,…)



The solution is: newnumber (J(n))

And the newnumber(k) is **$2k-1$**

And What about 2*n*+1 Persons (n=1,2,3,...)



The solution is: newnumber (J(n))

And for the time, the newnumber(k) is **$2k+1$**

Solution in Recursive Equations

$$J(1) = 1;$$

$$J(2n) = 2J(n) - 1, \qquad \text{for } n \geqslant 1;$$

$$J(2n+1) = 2J(n) + 1, \qquad \text{for } n \geqslant 1.$$

Explicit Solution for small *n*’s

n	1	2 3	4 5 6 7	8 9 10 11 12 13 14 15	16
$J(n)$	1	1 3	1 3 5 7	1 3 5 7 9 11 13 15	1

Look carefully ...

and, find the pattern...

and, prove it!

Eureka!

If we write n in the form $n = 2^m + l$,

(where 2^m is the largest power of 2 not exceeding n and where l is what's left),

the solution to our recurrence seems to be:

$$J(2^m + l) = 2l + 1, \qquad \text{for } m \geq 0 \text{ and } 0 \leq l < 2^m.$$

As an example: J(100) = J(64+36) = 36*2+1 = 73

Binary Representation

- Suppose *n*'s binary expansion is :

$$n = (b_m\, b_{m-1} \ldots b_1\, b_0)_2$$

- then:

$$\begin{aligned} n &= (1\, b_{m-1}\, b_{m-2} \ldots b_1\, b_0)_2\,, \\ l &= (0\, b_{m-1}\, b_{m-2} \ldots b_1\, b_0)_2\,, \\ 2l &= (b_{m-1}\, b_{m-2} \ldots b_1\, b_0\, 0)_2\,, \\ 2l+1 &= (b_{m-1}\, b_{m-2} \ldots b_1\, b_0\, 1)_2\,, \\ J(n) &= (b_{m-1}\, b_{m-2} \ldots b_1\, b_0\, b_m)_2 \end{aligned}$$