

# Discrete

## Mathematics - P101

**有根树** (rooted tree). 由一个顶点集合和连接这些顶点的边组成。顶点集合包含一个特殊顶点，树根。

有根树的集合可以递归地定义

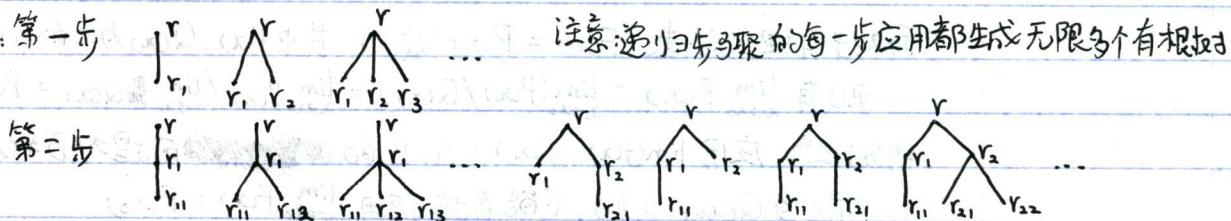
**基础步骤聚**: 单个顶点,  $r$  是有根树

**递归步骤聚**: 假设  $T_1, T_2, \dots, T_n$  是有根树, 并分别有树根  $r_1, r_2, \dots, r_n$ , 其中  $n \in \mathbb{Z}^+$

则指定一个新的树根  $r$ ,  $r$  不属于  $T_1, T_2, \dots, T_n$  加入边连接  $r$  与  $r_1, r_2, \dots, r_n$  形成的树  $T$  也是有根树

**基础步骤聚**:

**递归步骤聚**: 第一步 注意: 递归步骤聚的每一步应用都生成无限多个有根树



**满二叉树** (full binary tree), 指在一个二叉树中, 对每一个结点, 或者是叶结点, 或者有两个子结点

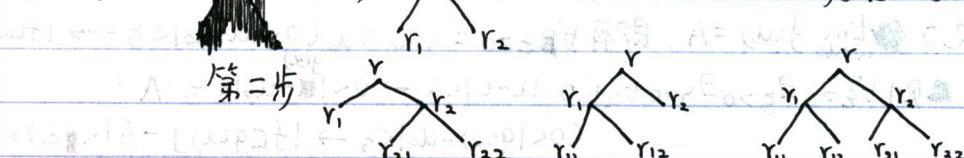
满二叉树的集合可以递归地定义

**基础步骤聚**: 单个顶点,  $r$  是满二叉树

**递归步骤聚**: 如果  $T_1$  和  $T_2$  都是满二叉树, 则存在一个表示为  $T_1, T_2$  的满二叉树  $T$  包含树根  $r$ , 连接  $r$  与  $T_1, T_2$  树根  $r_1, r_2$  的边

**基础步骤聚**:

**递归步骤聚**: 第一步 注意: 递归步骤聚的每一步应用都生成  $2^i - 1$  个满二叉树



特别注意: 满二叉树的国内外定义是不同的, 会产生歧义

国内对满二叉树的定义为, 如果一个二叉树有  $k$  层, 且总结点数为  $2^k - 1$ , 则其为满二叉树

**基础步骤聚**: 单个顶点,  $r$  是满二叉树

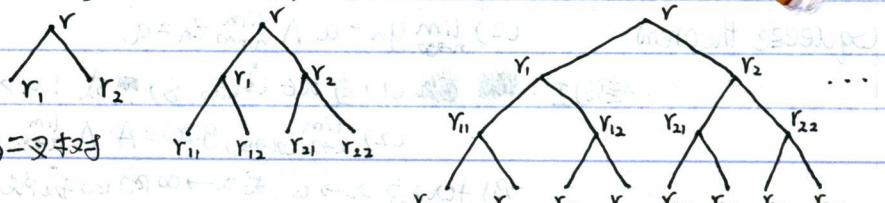
**递归步骤聚**: 在递归步骤聚的第一步, 如果  $T_1, T_2$  是第  $k-1$  步生成的满二叉树, 其中  $k \in \mathbb{Z}^+$

则存在一个表示为  $T_1, T_2$  的满二叉树  $T$

$T$  包含树根  $r$ , 和连接  $r$  与  $T_1, T_2$  树根  $r_1, r_2$  的边

**基础步骤聚** ( $k=0$ )

**递归步骤聚**: 第一步 第二步 第3步 ...



注意: 递归步骤聚第一步应用生成 1 个满二叉树

# Discrete

## Mathematics - P102

扩展二叉树 (Extended binary tree) 指允许左子树或右子树为空的二叉树

扩展二叉树可以通过在空的子树上增加空的叶结点，使二叉树成为国际定义的满二叉树

扩展二叉树的集合可以递归地定义

基础步骤：空集是扩展二叉树。

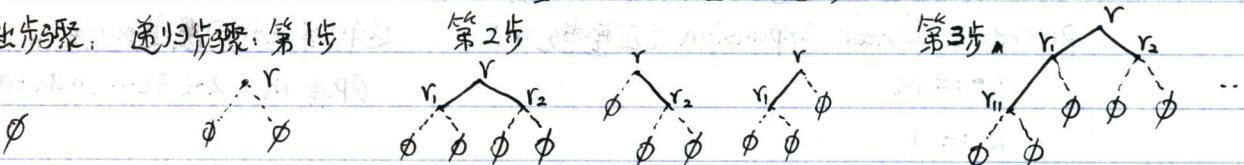
递归步骤：如果  $T_1, T_2$  为扩展二叉树，则存在表示为  $T_1 \cdot T_2$  的扩展二叉树  $T$

$T$  包括根结点  $r$  和当  $T_1$  或  $T_2$  非空时，连接  $r$  与  $r_1$  或  $r_2$  的边

基础步骤：递归步骤：第1步

第2步

第3步



结构归纳法 (Structural induction)，指证明关于递归定义的集合的结果的技术

不直接使用数学归纳法也可以证明关于递归定义的集合的结果

基础步骤：对于递归定义的基础步骤所规定的属于该集合的所有元素

证明命题是对这些元素为真

递归步骤：如果对于递归定义的递归步骤所使用的每个已知元素命题为真

则根据规则生成的新元素也有命题为真

结构归纳法可以用于证明关于树或特殊类型的树 (如满二叉树) 的命题

基础步骤：对于只含有单个顶点的满二叉树，证明命题为真

递归步骤：如果对于满二叉树  $T_1, T_2$  命题为真

则对于包含根结点  $r$  和以  $T_1, T_2$  为左右子树的树  $T_1 \cdot T_2$ ，命题为真

$h(T)$

记为定义在满二叉树集合上的函数，返回满二叉树的高度  $h(T)$

基础步骤：对于只含有单个顶点的满二叉树  $T$ ,  $h(T)=0$

递归步骤：如果满二叉树  $T_1, T_2$  有高度  $h(T_1), h(T_2)$

则满二叉树  $T_1 \cdot T_2$  有高度  $h(T_1 \cdot T_2) = 1 + \max(h(T_1), h(T_2))$

$n(T)$

记为定义在满二叉树集合上的函数，返回满二叉树的结点数  $n(T)$

基础步骤：对于只含有单个顶点的满二叉树  $T$ ,  $n(T)=1$

递归步骤：如果满二叉树  $T_1, T_2$  有结点数  $n(T_1), n(T_2)$

则满二叉树  $T_1 \cdot T_2$  有结点数  $n(T) = 1 + n(T_1) + n(T_2)$

注意以上高度和结点数的定义对于国内定义和国际定义都是适用的

# Discrete

## Mathematics - P103

对于满二叉树  $T$ ,  $n(T) \leq 2^{h(T)+1} - 1$

基础步骤聚: 对于只含有树根  $r$  的满二叉树  $T$ ,  $h(T) = 0$

$$n(T) = 1 \leq 2^{0+1} - 1, \text{ 即 } P(T) \text{ 为真}$$

递归步骤聚: 假设对于满二叉树  $T_1, T_2$ , 有  $n(T_1) \leq 2^{h(T_1)+1} - 1$  且  $n(T_2) \leq 2^{h(T_2)+1} - 1$

则对于满二叉树  $T = T_1 \cdot T_2$ ,  $h(T) = \max(h(T_1), h(T_2)) + 1$

$$n(T) = 1 + n(T_1) + n(T_2)$$

$$\leq 1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1 \quad \text{根据归纳假设}$$

$$\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1$$

$$= 2 \cdot 2^{\max(h(T_1), h(T_2)) + 1} - 1 = 2^{h(T)+1} - 1, \text{ 即 } P(T_1 \cdot T_2) \text{ 为真}$$

于是依据结构归纳法, 对于满二叉树  $T$ , 有  $n(T) \leq 2^{h(T)+1} - 1$

注意对于国内定义的满二叉树, 有  $n(T) = 2^{h(T)+1} - 1$

对于  $x, y \in \Sigma^*$ ,  $l(xy) = l(x) + l(y)$ , 其中  $\Sigma^*$  为定义在字母表  $\Sigma$  上的字符串集合

取定义在  $\Sigma^*$  上的命题  $P(y)$  为, 对任意  $x \in \Sigma^*$ ,  $l(xy) = l(x) + l(y)$

基础步骤聚: 当  $y = \lambda$  时, 对于任意  $x \in \Sigma^*$ ,  $l(x\lambda) = l(x) + 0 = l(x) + l(\lambda)$

于是有  $P(\lambda)$  为真

递归步骤聚: 假设对于任意  $y \in \Sigma^*$ , 有  $P(y)$  为真, 则考虑  $P(ya)$ , 其中  $a \in \Sigma$

根据  $l(w)$  函数的递归定义,  $l(xy) = l(xy) + 1$ ,  $l(ya) = l(y) + 1$

又根据归纳假设,  $l(xy) = l(x) + l(y) + 1$

则  $l(xy) = l(xy) + 1 \stackrel{(II)}{=} l(x) + l(y) + 1 = l(x) + l(ya)$ , 即  $P(ya)$  为真

于是依据结构归纳法, 对于任意  $x, y \in \Sigma^*$ , 有  $l(xy) = l(x) + l(y)$

$W^R$

记为定义在字符串集合  $\Sigma^*$  上的运算, 结果为字符串  $w$  的倒置 (inverse), 或称反转

基础步骤聚: 对于空串  $\lambda$ ,  $\lambda^R = \lambda$

递归步骤聚: 如果有  $w \in \Sigma^*$  且  $a \in \Sigma$ , 则  $(wx)^R = a w^R$

对于任意  $w_1, w_2 \in \Sigma^*$ , 有  $(w_1 w_2)^R = w_2^R w_1^R$

取定义在  $\Sigma^*$  上的命题  $P(x)$  为, 对任意  $w \in \Sigma^*$ ,  $(wx)^R = x^R w^R$

基础步骤聚: 当  $x = \lambda$  时, 对于任意  $w \in \Sigma^*$ ,  $(w\lambda)^R = w^R = \lambda^R w^R$ , 即有  $P(\lambda)$  为真

递归步骤聚: 假设对于任意  $x \in \Sigma^*$ , 有  $P(x)$  为真, 则考虑  $P(xa)$ , 其中  $a \in \Sigma$

根据  $w^R$  的递归定义, 有  $(wx)a^R = a(wx)^R$ ,  $(xa)^R = a x^R$

又根据归纳假设,  $(wx)^R = x^R w^R$

则  $(wx)a^R = a(wx)^R \stackrel{(II)}{=} a x^R w^R = (xa)^R w^R$ , 即  $P(xa)$  为真

于是依据结构归纳法, 对于任意  $w_1, w_2 \in \Sigma^*$ , 有  $(w_1 w_2)^R = w_2^R w_1^R$

# Discrete

## Mathematics - P104

$w^i \in \Sigma^i$  表示为定义在  $\Sigma^*$  上的运算，结果为  $w$  的重复 (replicate)，其中  $\Sigma^*$  为字母表上的字符串的集合

基础步骤：当  $i=0$  时， $w^0 = \lambda$

递归步骤：对于任意  $i \in \mathbb{Z}^+$ ， $w^i = w^{i-1}w$

对于任意  $w \in \Sigma^*$  和  $i \in N$ ，有  $|cw^i| = i \cdot |cw|$

基础步骤：当  $i=0$  时， $|cw^0| = |c\lambda| = 0 = 0 \cdot |cw|$

递归步骤：假设对任意  $i \in N$ ，有  $|cw^i| = i \cdot |cw|$ ，则考虑  $|cw^{i+1}|$

已知  $w^{i+1} = ww^i$ ，且  $w, w^i \in \Sigma^*$ ，则有  $|cw^{i+1}| = |cw w^i| = |cw| + |w^i|$

根据归纳假设， $|cw^i| = i \cdot |cw|$

于是根据结构归纳法，对于任意  $w \in \Sigma^*$  和  $i \in N$ ，有  $|cw^i| = i \cdot |cw|$

对于任意  $w \in \Sigma^*$  和  $i \in N$ ，有  $(wi)^R = (w^R)^i$

基础步骤：当  $i=0$  时， $(w^0)^R = (\lambda)^R = \lambda = (w^R)^0$

递归步骤：假设对任意  $i \in N$ ，有  $(w^i)^R = (w^R)^i$ ，则考虑  $(w^{i+1})^R$

已知  $w^{i+1} = ww^i$ ，且  $w, w^i \in \Sigma^*$ ，则有  $(ww^i)^R = (w^i)^R w^R$

又根据归纳假设  $(w^i)^R = (w^R)^i$

则有  $(w^{i+1})^R = (ww^i)^R = (w^i)^R w^R \stackrel{(IH)}{=} (w^R)^i w^R = (w^R)^{i+1}$

于是根据结构归纳法，对于任意  $w \in \Sigma^*$  和  $i \in N$ ，有  $(wi)^R = (w^R)^i$

对于正整数  $n$ ，令  $f_n$  为第  $n$  个斐波那契数，另外对于  $n=0$ ， $f_0 = 0$

对于  $n \in \mathbb{Z}^+$ ，有  $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$

基础步骤：当  $n=1$  时， $f_1^2 = 1^2 = f_1 \cdot f_2$

递归步骤：假设对任意  $n \in \mathbb{Z}^+$ ，有  $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$ ，则考虑  $P(n+1)$

$f_1^2 + f_2^2 + \dots + f_n^2 + f_{n+1}^2 \stackrel{(IH)}{=} f_n f_{n+1} + f_{n+1}^2 = (f_n + f_{n+1}) f_{n+1} = f_{n+1} f_{n+2}$

于是依据数学归纳法，对于  $n \in \mathbb{Z}^+$ ，有  $f_1^2 + f_2^2 + \dots + f_n^2 = f_n f_{n+1}$

对于  $n \in \mathbb{Z}^+$ ，有  $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$

基础步骤：当  $n=1$  时， $f_1 = 1 = f_2$

递归步骤：假设对任意  $n \in \mathbb{Z}^+$ ，有  $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$  为真，则考虑  $P(n+1)$

$f_1 + f_3 + \dots + f_{2n-1} + f_{2n+1} \stackrel{(IH)}{=} f_{2n} + f_{2n+1} = f_{2n+2}$

于是依据数学归纳法，对于  $n \in \mathbb{Z}^+$ ，有  $f_1 + f_3 + \dots + f_{2n-1} = f_{2n}$

# Discrete Mathematics - P105

例题 - 例题 3

对于  $n \in \mathbb{Z}^+$ , 有  $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$

基础步骤: 当  $n=1$  时,  $f_2 \cdot f_0 - f_1^2 = 1 \times 0 - 1^2 = -1 = (-1)^1$

递归步骤: 假设对任意正整数  $n$ ,  $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$  为真, 则考虑  $P(n+1)$

$$\begin{aligned} f_{n+2}f_n - f_{n+1}^2 &= (f_n + f_{n+1})f_n - (f_{n-1} + f_n)f_{n-1} \\ &= f_n^2 + f_{n+1}f_n - f_{n-1}^2 - 2f_{n-1}f_n - f_n^2 \\ &= (f_{n+1} - f_{n-1})f_n - (f_{n-1} + f_n)f_{n-1} \\ &= f_n^2 - f_{n+1}f_{n-1} = -(f_{n+1}f_{n-1} - f_n^2) \stackrel{(IH)}{=} -(-1) \cdot (-1)^n = (-1)^{n+1} \end{aligned}$$

于是依据数学归纳法, 对于任意  $n \in \mathbb{Z}^+$ , 有  $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$

对于  $n \in \mathbb{Z}^+$ , 有  $f_0 + f_1 + f_2 + \dots + f_{2n-1} + f_{2n} = f_{2n}^2$

基础步骤: 当  $n=1$  时,  $f_0 + f_1 + f_2 = 0 \times 1 + 1 \times 1 = 1 = f_2^2$

递归步骤: 假设对任意正整数  $n$ ,  $f_0 + f_1 + f_2 + \dots + f_{2n-1} + f_{2n} = f_{2n}^2$  为真, 则考虑  $P(n+1)$

$$\begin{aligned} f_0 + f_1 + f_2 + \dots + f_{2n-1} + f_{2n} + f_{2n}f_{2n+1} + f_{2n+1}f_{2n+2} &\stackrel{(IH)}{=} f_{2n}^2 + f_{2n}f_{2n+1} + f_{2n+1}f_{2n+2} = f_{2n}(f_{2n} + f_{2n+1}) + f_{2n+1}f_{2n+2} \\ &= f_{2n}f_{2n+2} + f_{2n+1}f_{2n+2} = f_{2n+2}^2 \end{aligned}$$

于是依据数学归纳法, 对于任意  $n \in \mathbb{Z}^+$ , 有  $f_0 + f_1 + f_2 + \dots + f_{2n-1} + f_{2n} = f_{2n}^2$

对于  $n \in \mathbb{Z}^+$ , 有  $f_0 - f_1 + f_2 - \dots - f_{2n-1} + f_{2n} = f_{2n-1} - 1$

基础步骤: 当  $n=1$  时,  $f_0 - f_1 + f_2 = 0 - 1 + 1 = 0 = f_{2 \times 1 - 1} - 1$

递归步骤: 假设对任意正整数  $n$ ,  $f_0 - f_1 + f_2 - \dots - f_{2n-1} + f_{2n} = f_{2n-1} - 1$  为真, 则考虑  $P(n+1)$

$$\begin{aligned} f_0 - f_1 + f_2 - \dots - f_{2n-1} + f_{2n} - f_{2n+1} + f_{2n+2} &\stackrel{(IH)}{=} f_{2n-1} - 1 - f_{2n+1} + f_{2n+2} = f_{2n+2} - (f_{2n+1} - f_{2n-1}) - 1 \\ &= f_{2n+2} - f_{2n} - 1 = f_{2n+1} - 1 \end{aligned}$$

于是依据数学归纳法, 对于任意  $n \in \mathbb{Z}^+$ , 有  $f_0 - f_1 + f_2 - \dots - f_{2n-1} + f_{2n} = f_{2n-1} - 1$

如果  $A$  为  $2 \times 2$  矩阵  $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ , 则对于  $n \in \mathbb{Z}^+$ ,  $A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$

基础步骤: 当  $n=1$  时,  $A^1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} f_2 & f_1 \\ f_1 & f_0 \end{bmatrix}$

递归步骤: 假设对任意正整数  $n$ ,  $A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$  为真, 则考虑  $P(n+1)$

$$A^{n+1} = A^n A \stackrel{(IH)}{=} \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} f_{n+1} + f_n & f_{n+1} \\ f_n + f_{n-1} & f_n \end{bmatrix} = \begin{bmatrix} f_{n+2} & f_{n+1} \\ f_{n+1} & f_n \end{bmatrix}$$

于是依据数学归纳法, 对于任意  $n \in \mathbb{Z}^+$ ,  $A^n = \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$

注意, 对于矩阵的行列式, 有  $|A^n| = |A|^n$ , 其中  $n \in \mathbb{Z}^+$

则有  $f_{n+1}f_{n-1} - f_n^2 = \begin{vmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}^n = (-1)^n$

# Discrete Mathematics - P106

对于满二叉树  $T$ , 其树叶数为  $l(T)$ , 内点数为  $i(T)$ , 则有  $l(T) = i(T) + 1$

基础步骤: 对于只含有对根的树  $T$ ,  $l(T) = 1 = 0 + 1 = i(T) + 1$

递归步骤: 假设对于满二叉树  $T_1, T_2$ , 有  $l(T_1) = i(T_1) + 1$ ,  $l(T_2) = i(T_2) + 1$  为真

则对于满二叉树  $T = T_1 \cdot T_2$ ,  $T$  包含树根  $r$  和左右子树分别为  $T_1, T_2$

于是有  $l(T) = l(T_1) + l(T_2)$ ,  $i(T) = i(T_1) + i(T_2) + 1$

$$\text{则 } l(T) = l(T_1) + l(T_2) \stackrel{(IH)}{=} i(T_1) + 1 + i(T_2) + 1 \\ = [i(T_1) + i(T_2) + 1] + 1 = i(T) + 1, \text{ 即 } PP(T) \text{ 为真}$$

于是依据结构归纳法, 对于满二叉树  $T$ , 有  $l(T) = i(T) + 1$

注意这个命题对于满二叉树的国内定义与国际定义均成立

字典序 (lexicographical order), 指一般化地将单词依据其构成字母在字母表的排序进行排序的方法  
a generalization of the way words are alphabetically ordered  
based on the alphabetical order of their component letters

如可以定义  $N \times N$  (即非负整数的有序对) 上的字典序为

如果对于  $(x_1, y_1), (x_2, y_2) \in N \times N$ , 有  $x_1 < x_2$  或  $x_1 = x_2$  且  $y_1 < y_2$ ,

则有  $(x_1, y_1) < (x_2, y_2)$

于是可知这个集合的非空子集具有最小元素, 即这个集合是良序的

即所有有序对的  $x$  值最小的有序对中  $y$  值最小的有序对

广义归纳法, 指对数学归纳法进行扩展, 用于证明除整数集合外其他具有良序性的集合的命题

如对于递归地定义  $\{a_{m,n}\}$  的集合, 其中  $m \in N$ ,  $n \in N$ .

则可以利用定义在  $N \times N$  上的字典序和广义归纳法证明定义在  $\{a_{m,n}\}$  上的命题

对于  $(m, n) \in N \times N$ , 递归地定义  $a_{m,n}$ , 递归步聚:  $a_{m,n} = \begin{cases} a_{m-1,n} + 1 & n=0 \text{ 且 } m>0 \\ a_{m,n-1} + n & n>0 \end{cases}$

基础步聚:  $a_{0,0} = 0$

则有对于任意  $(m, n) \in N \times N$ , 有  $a_{m,n} = m + \frac{n(n+1)}{2}$

基础步聚: 当  $(m, n) = (0, 0)$  时,  $a_{m,n} = 0 = 0 + \frac{0(0+1)}{2}$

递归步聚: 假设对于任意  $(m, n) \neq (0, 0)$ , 对任意字典序小于  $(m, n)$  的  $(m', n')$ ,  $a_{m',n'} = m' + \frac{n'(n'+1)}{2}$

则当  $n=0$  且  $m>0$  时,  $a_{m,n} = a_{m-1,n} + 1$ , 又  $(m-1, n) < (m, n)$

根据归纳假设设  $a_{m-1,n} = m-1 + \frac{n(n+1)}{2} = m-1$

则有  $a_{m,n} = a_{m-1,n} + 1 = m = m + \frac{n(n+1)}{2}$

则当  $n>0$  时,  $a_{m,n} = a_{m,n-1} + n$ , 根据归纳假设设  $a_{m,n-1} = m + \frac{(n-1)n}{2}$

又  $a_{m,n} = a_{m,n-1} + n = m + \frac{(n-1)n}{2} + n = m + \frac{n(n+1)}{2}$

于是依据广义归纳法, 对于  $(m, n) \in N \times N$ , 递归定义的  $a_{m,n} = m + \frac{n(n+1)}{2}$

# Discrete

## Mathematics - P107

正整数的分拆 (integer partition) 指把正整数  $n$  写为正整数之和的方式，  
令  $P_m$  为对正整数  $m$  的不同分拆的数目，其中和式里整数的顺序无关紧要  
并令  $P_{m,n}$  为用不超过  $n$  的正整数之和来表示  $m$  的不同分拆的数目

首先可知  $P_{m,m} = P_m$  即对正整数的不同分拆的数目等于用不超过其本身的正整数之和的分拆数目

证明过程为，取辅助函数  $Q_{m,n}$  为用最大项为  $n$  的正整数之和来表示  $m$  的不同分拆数目

则可知  $Q_{m,n} = P_{m,n} - P_{m,n-1}$ ,  $Q_{m,1} = P_{m,1}$

且有  $P_m = Q_{m,1} + Q_{m,2} + Q_{m,3} + \dots + Q_{m,m} + Q_{m,m+1} + \dots$

可知  $\forall n \in \mathbb{Z}^+$  ( $n > m \rightarrow Q_{m,n} = 0$ ) 即对  $m$  的分拆中不存在比  $m$  大的正整数

于是有  $P_m = \sum_{k=1}^m Q_{m,k} = P_{m,1} + (P_{m,2} - P_{m,1}) + \dots + (P_{m,m-1} - P_{m,m-2}) + (P_{m,m} - P_{m,m-1})$

又  $P_{m,n}$  有递归定义，  
 $\left\{ \begin{array}{l} P_{1,1} = 1, \\ P_{m,1} = 1, \text{ 对于 } 1 \text{ 的分拆只有 } 1 = 1 \end{array} \right.$

对于  $m \in \mathbb{Z}^+, n \in \mathbb{Z}^+$   
 $P_{m,n} = \begin{cases} P_{m,m}, & m < n, \text{ 由于 } m \text{ 只可能用不超过其自身的正整数之和表示} \\ 1 + P_{m,m-1}, & m = n > 1, \text{ 由于 } P_{m,m} - P_{m,m-1} = Q_{m,m} = 1 \\ P_{m,n-1} + P_{m-n,n}, & m > n > 1 \end{cases}$

由于  $Q_{m,n}$  中对  $m$  的分拆必定包含至少一项  $n$ ，去掉此项则是一个  $P_{m-n,n}$  的分拆

所以  $Q_{m,n}$  中的分拆与  $P_{m-n,n}$  中的分拆是一一对应的

$Q_{m,n} = P_{m-n,n} = P_{m,n} - P_{m,n-1}$

于是有  $P_{m,n} = P_{m,n-1} + P_{m-n,n}$

integerPartition :: Int → Int

integerPartition m = part m m

let part m n

|  $m == 1 \text{ or } n == 1$  : return 1

|  $m < n$  : return part m n

|  $m == n$  : return 1 + (part m (m-1))

|  $m > n$  : return (part m (n-1)) + (part (m-n) n)

def integerPartition(m):

def part(m, n): if m == 1 or n == 1: return 1

elif m < n: return part(m, m)

elif m == n: return 1 + part(m, m-1)

else: m > n: return part(m, n-1) + part(m-n, n)

return part(m, m)

# Discrete

## Mathematics - P108

原始递归函数 (primitive recursive function), 在可计算理论中, 使用复合与原始递归作为中心运算来定义一类函数  
in computability theory, a class of functions defined

using composition and primitive recursion as central operation

对计算的完全的形式化而言, 原始递归函数形成重要的构造模块

form an important building block on the way to a full formalization of computability

原始递归函数皆是可计算函数, 即可以用总是停机的图灵机计算

但可计算函数并不都是原始递归函数

原始递归函数接受自然数或自然数的  $n$  元组 ( $n \in \mathbb{Z}^+$ ) 作为参数并返回自然数

定义 基本原始递归函数由公理定义

常数函数 (constant function), 0元常数函数  $C_0$  (0-ary constant function) ( $C_0 = 0$ ) 是原始递归函数  
后继函数 (successor function), 1元后继函数  $S(x) = x + 1$  是原始递归函数

返回传入参数根据皮亚诺公理 (Peano postulate) 得到的后继数

投影函数 (projection function), 对于任意  $n \geq 1$ , 和  $1 \leq i \leq n$ , 则有  $n$  元投射函数  $P_i^n$  返回  $n$  个参数中的第  $i$  个参数  
即  $P_i^n(x_1, x_2, \dots, x_n) = x_i$ ,  $P_i^n$  是原始递归函数

更复杂的原始递归函数可以通过应用公理中的运算 (operation) 获得

复合 (composition), 对于给定的  $k$  元原始递归函数  $f$ , 和  $k+m$  元原始递归函数  $g_1, g_2, \dots, g_k$ , 其中  $k, m \in \mathbb{Z}^+$

定义  $m$  元函数  $h(x_1, \dots, x_m) = f[g_1(x_1, \dots, x_m), g_2(x_1, \dots, x_m), \dots, g_k(x_1, \dots, x_m)]$

即函数  $h$  是函数  $f, g_1, g_2, \dots, g_k$  的复合, 且有函数  $h(x_1, \dots, x_m)$  是原始递归函数

原始递归 (primitive recursion), 对于给定的  $k$  元原始递归函数  $f$ , 和  $k+2$  元原始递归函数  $g$ , 其中  $k \in \mathbb{N}$

定义  $k+1$  元函数  $h(0, x_1, x_2, \dots, x_k) = f(x_1, x_2, \dots, x_k)$

如果  $h(S(y), x_1, x_2, \dots, x_k) = g[y, h(y, x_1, x_2, \dots, x_k), x_1, x_2, \dots, x_k]$

注意投影函数的作用, 投影函数可以用于避免明显刻板的函数元数方式 (apparent rigidity in terms)

即可以把一个函数的参数子集传入另一个函数, 如对  $f(a, b, c) = g[h(a, c), h(a, b)]$

可形式化地定义为  $f(a, b, c) = g[h[P_1^3(a, b, c), P_3^3(a, b, c)], h[P_1^3(a, b, c), P_2^3(a, b, c)]]$

常数函数 Constant :: Int

constant = 0

则可以以原始递归函数形式化地定义加法函数

后继函数 Successor :: Int → Int

pri\_rec\_add :: Int → Int → Int

Successor x = x + 1

pri\_rec\_add x 0 = projection 1 | | [x]

投影函数 projection :: Int → Int → [Int] → Int

pri\_rec\_add x y =

projection n i args = args !! (i-1)

let y' = predecessor y

in Successor (projection 3 2

(y': (pri\_rec\_add x y'): x: []))

后继函数 predecessor :: Int → Int

in Successor (projection 3 2

(y': (pri\_rec\_add x y'): x: []))

的反函数 predecessor x = x - 1

# Discrete

## Mathematics - P109

阿克曼函数 (Ackermann function), 输入两个自然数而输出一个自然数, 输出值增长速度非常快  
 是可计算函数, 但是非原始递归函数的最近发现的最简单的实例之一  
 one of the simplest and earliest-discovered examples of total computable function that is not primitive recursive function

通常阿克曼函数指一类有不同变种, 但具有类似性质的函数  
 即传入自然数或自然数的n元组为参数并返回自然数, 且返回值随输入增长的增长速度非常快  
 都是可计算函数, 也是递归函数, 但是非原始递归函数

### 原始版本

由希耳伯特 (Hilbert) 的学生 Sudan 和 Ackermann 在 1920 年代提出.

递归地定义函数  $\varphi(m, n, p)$ , 其中  $m, n, p \in \mathbb{N}$

$$\varphi(m, n, 0) = m + n$$

$$\varphi(m, 0, 1) = 0$$

$$\varphi(m, 0, 2) = 1$$

$$\varphi(m, 0, p) = m$$

$$\varphi(m, n, p) = \varphi(m, \varphi(m, n-1, p), p-1)$$

$$\text{origin\_ack} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$$

$$\text{origin\_ack } m \ n \ 0 = m + n$$

$$\text{origin\_ack } m \ 0 \ 1 = 0$$

$$\text{origin\_ack } m \ 0 \ 2 = 1$$

$$\text{origin\_ack } m \ 0 \ p = m$$

$$\text{origin\_ack } m \ n \ p =$$

希耳伯特在 On the Infinite 中猜想此为非原始递归函数

而阿克曼在 On Hilbert's Construction of the Real Numbers 中证明了这个猜想

### 常用版本

由 Rózsa Péter 和 Raphael Robinson 定义, 为讨论阿克曼函数时较常用的版本

递归地定义函数  $A(m, n)$ , 其中  $m, n \in \mathbb{N}$

$$A(0, n) = n + 1$$

$$\text{ack } 0 \ n = n + 1$$

$$A(m, 0) = A(m-1, 1), \quad m > 0$$

$$\text{ack } m \ 0 = \text{ack } (m-1) \ 1$$

$$A(m, n) = A(m-1, A(m, n-1)), \quad n > 0$$

$$\text{ack } m \ n = \text{ack } (m-1) (\text{ack } m \ (n-1))$$

### $A(m, n)$ 的值

$m \backslash n$	0	1	2	3	4	$n$
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2 = 2 + (n+3) - 3$
2	3	5	7	9	11	$2n+3 = 2(n+3) - 3$
3	5	13	29	61	125	$2^{n+3} - 3 = 2^{\uparrow (n+3)} - 3$
4	$13 = 2^2 - 3$	$65533 = 2^{2^2} - 3$	$2^{65536} - 3$	$2^{2^{65536}} - 3$	$2^{2^{2^{65536}}} - 3$	$2^{\uparrow \uparrow \uparrow (n+3)} - 3$
5	$65533 = 2^{\uparrow \uparrow \uparrow 3} - 3$	$2^{\uparrow \uparrow \uparrow 4} - 3$	$2^{\uparrow \uparrow \uparrow 5} - 3$	$2^{\uparrow \uparrow \uparrow 6} - 3$	$2^{\uparrow \uparrow \uparrow 7} - 3$	$2^{\uparrow \uparrow \uparrow (n+3)} - 3$
6	$2^{\uparrow \uparrow \uparrow 3} - 3$	$2^{\uparrow \uparrow \uparrow 4} - 3$	$2^{\uparrow \uparrow \uparrow 5} - 3$	$2^{\uparrow \uparrow \uparrow 6} - 3$	$2^{\uparrow \uparrow \uparrow 7} - 3$	$2^{\uparrow \uparrow \uparrow \uparrow (n+3)} - 3$
$m$	$(2 \rightarrow 3) \rightarrow (m-2) - 3$	$(2 \rightarrow 4) \rightarrow (m-2) - 3$	$(2 \rightarrow 5) \rightarrow (m-2) - 3$	$(2 \rightarrow 6) \rightarrow (m-2) - 3$	$(2 \rightarrow 7) \rightarrow (m-2) - 3$	$(2 \rightarrow (n+3) \rightarrow (m-2)) - 3 = 2^{\uparrow \uparrow \uparrow \uparrow (n+3)} - 3$

# Discrete

## Mathematics - P110

89 - Juttigog A

考虑阿克曼函数(Ackermann function)的一个版本。

递归地定义函数  $A(m, n)$ , 其中  $m, n \in \mathbb{N}$

$$A(0, n) = 1$$

$$A(1, 0) = 2$$

$$A(m, 0) = m+2, m \geq 2$$

$$A(m, n) = A(A(m-1, n), n-1), n \geq 1$$

$\text{ackermann} :: \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

$$\text{ackermann } 0, n = 1$$

$$\text{ackermann } 1, 0 = 2$$

$$\text{ackermann } m, 0 = m+2$$

$$\text{ackermann } m, n = \text{ackermann}(\text{ackermann}(m-1), n)$$

对任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 1) = 2m$

基础步骤: 当  $m=1$  时,  $A(1, 1) = A(A(0, 1), 0) = A(1, 0) = 2$

递归步骤: 假设对任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 1) = 2m$  为真, 则考虑  $A(m+1, 1)$

$$\text{则 } A(m+1, 1) = A(A(m+1-1, 1), 0) = A(m, 1) + 2 \stackrel{(IH)}{=} 2m + 2 = 2(m+1)$$

于是根据数学归纳法, 对于任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 1) = 2m$

对任意  $n \in \mathbb{N}$ , 有  $A(1, n) = 2^n$

基础步骤: 当  $n=0$  时,  $A(1, 0) = 2$  平凡地为真

递归步骤: 假设对任意  $n \in \mathbb{N}$  时, 有  $A(1, n) = 2^n$  为真, 则考虑  $A(1, n+1)$

$$\begin{aligned} \text{则 } A(1, n+1) &= A(A(1-1, n+1), n+1-1) \\ &= A(A(0, n+1), n) = A(1, n) \stackrel{(IH)}{=} 2^n \end{aligned}$$

于是根据数学归纳法, 对于任意  $n \in \mathbb{N}$ , 有  $A(1, n) = 2^n$

对任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 2) = 2^m$

基础步骤: 当  $m=1$  时,  $A(1, 2) = A(A(1-1, 2), 2-1) = A(1, 1) = 2$

递归步骤: 假设对任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 2) = 2^m$  为真, 则考虑  $A(m+1, 2)$

$$\text{则 } A(m+1, 2) = A(A(m, 2), 2-1) = A(A(m, 2), 1)$$

又 对任意  $m \in \mathbb{Z}^+$ ,  $A(m, 2) = 2^m > 1$

$$\begin{aligned} \text{则 } A(m+1, 2) &= A(A(m, 2), 1) = 2 \cdot A(m, 2) = 2 \cdot 2^m \\ &\stackrel{(IH)}{=} 2 \cdot 2^m = 2^{m+1} \end{aligned}$$

于是根据数学归纳法, 对于任意  $m \in \mathbb{Z}^+$ , 有  $A(m, 2) = 2^m$

又根据克努特箭头( $\uparrow$ )的定义, 有  $A(m, 2) = 2^m = 2 \uparrow m$

则可以猜测这个阿克曼函数在  $n \geq 2$  时, 对任意  $m \in \mathbb{Z}^+$ ,

$$\text{有 } A(m, n) = 2 \uparrow^{n-1} m = 2 \rightarrow (m) \rightarrow (n-1)$$

# Discrete Mathematics - P111

迭代幂次 (tetration) 又称~~迭代乘方~~，幂塔运算 超幂运算，指幂的下一个超运算级别，用以表示极大数字。  
由皮亚诺公理定义的后继函数  $a' = a + 1$ ，通常认为是第0级超运算，即超-0运算。

则有加法  $a + n = \underbrace{a + a + \dots + a}_{n\text{t}}$ ，即加法运算的  $a + n$  可理解为  $a$  的  $n$  次后继运算。

乘法  $a * n = \underbrace{a * a * \dots * a}_{n\text{t}}$ ，即乘法运算的  $a * n$  可理解为  $n$  个  $a$  连续~~相加~~ 加上  $a$  的运算。

幂  $a^n = \underbrace{a * a * \dots * a}_{n\text{t}}$ ，即幂运算的  $a^n$  可理解为  $n$  个  $a$  连续乘以  $a$  的运算。

于是可以得到~~迭代幂次~~的定义，记为  ${}^n a$ ，~~其~~中  $a$  为底数， $n$  为高阶值， $a \in \mathbb{Z}^+, n \in \mathbb{N}$ 。

即  ${}^n a = \underbrace{a * a * \dots * a}_{n\text{t}}$ ，即迭代幂次  ${}^n a$  可理解为  $a$  连续取幂于自己  $n$  次。

但是注意这个记法中~~其~~实只有  $(n-1)$  次幂运算。

而幂运算  $a^{a^a}$  可记为  $a^a * a^a$  (Haskell) 或  $a ** a ** a$  (Python)

且幂运算符是右结合的，如  $i (^) \rightarrow \text{index } 8$

即  $a^a * a^a$  等价于  $(a^a * a^a)$ ， $a ** a ** a$  等价于  $(a ** (a ** a))$

所以可以记为  ${}^n a = \underbrace{a * a * \dots * a}_{n\text{t}} = \underbrace{a * a * \dots * a}_{n\text{t}} * 1 = (a * a * \dots * (a * a * 1) \dots)$

递归定义运算  ${}^n a$ ，其中  $a \in \mathbb{Z}^+, n \in \mathbb{N}$ 。

tetration:  $\text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$

$${}^n a = \begin{cases} 1 & n=0 \\ a [{}^{n-1} a] & n>0 \end{cases}$$

tetration 0 = 1

tetration  $a n = a ^ (\text{tetration } a (n-1))$

另外特别地有， ${}^n a$  对于  $a$  和  $n$  都是单调递增的。当  $n$  也是正整数时，则有  ${}^n a$  是严格单调递增的。

即对于任意  $a \in \mathbb{Z}^+, n \in \mathbb{N}$ ，有  ${}^n (a+1) > {}^n a$ ， ${}^{(n+1)} a > {}^n a$ 。

基础步骤：当  $a=1$  时， $\forall n \in \mathbb{N}$ ， ${}^n a = 1$ ，于是有  $\forall n \in \mathbb{N}$ ， ${}^n a = 1 > {}^{n-1} a$

当  $n=0$  时， $\forall a \in \mathbb{Z}^+$ ， ${}^n a = 1$ ，于是有  $\forall a \in \mathbb{Z}^+$ ， ${}^n (a+1) = 1 > {}^n a$

递归步骤：假设  $\forall a \in \mathbb{Z}^+, n \in \mathbb{N}$ ， ${}^n a > {}^{n-1} a$  为真，对于字典序小于  $(a, n)$  的有序对  $(a', n')$ ， ${}^{n'} a' \geq {}^{n-1} (a-1) \wedge {}^{n'} a' \geq {}^{(n'-1)} a'$  为真。

则考虑  $(a, n)$ ， ${}^n a = a [{}^{n-1} a]$ ， ${}^{n-1} a > {}^{n-2} a$ ，

注意有对于大于 1 的正整数  $a$ ， $a$  当  $n \geq 0$  时，有  $a^n > n$ 。于是有  ${}^n a = a [{}^{n-1} a] > {}^{n-1} a$

注意对于大于 0 的整数  $n$ ，当  $a \geq 1$  时， $(a+1)^n > a^n$ ，

于是有  ${}^n a = a [{}^{n-1} a] > (a-1) [{}^{n-1} a] \stackrel{(IH)}{\geq} (a-1) [{}^{n-1} (a-1)] = {}^n (a-1)$

于是依据扩展归纳法，有对于任意  $a \in \mathbb{Z}^+, n \in \mathbb{N}$ ，有  ${}^n (a+1) > {}^n a$ ， ${}^{(n+1)} a > {}^n a$ 。

基于前述的阿克曼函数定义

对于任意  $m \in \mathbb{Z}^+$ ，有  $A(m, 3) = {}^m 2$

基础步骤：当  $m=1$  时， $A(1, 3) = 2 = {}^1 2$

递归步骤：假设对任意  $m \in \mathbb{Z}^+$ ，有  $A(m, 3) = {}^m 2$  为真，则考虑  $A(m+1, 3)$

$$A(m+1, 3) = A(A(m+1-1, 3), 3-1) = A(A(m, 3), 2)$$

$$= {}^{A(m, 3)} 2 \stackrel{(IH)}{=} {}^{{}^m 2} 2 = {}^{(m+1)} 2$$

于是依据数学归纳法，对于任意  $m \in \mathbb{Z}^+$ ，有  $A(m, 3) = {}^m 2$