§4.3 递归定义和结构归纳法

*有时很难给出一个对象的显式定义，但很容易用对象自己来定义该对象，这种方法称为递归(recursion)。

一．递归定义的函数

我们用两步定义一个以非负整数为定义域的函数:

基础步：说明函数在 0 处的值。

递归步：根据函数在较小的整数的值定义函数在更大的整数的值。

这种定义称为递归的(recursive)或归纳定义(inductive definition)。

例 1：给出以下递归定义的函数

f(0)=3

f(n+1)=2f(n)+3

求：f(1), f(2), f(3), f(4)。

解：$f(1)=2f(0)+3=2\cdot 3+3=9$

$f(2)=2f(1)+3=2\cdot 9+3=21$

$f(3)=2f(2)+3=2\cdot 21+3=45$

$f(4)=2f(3)+3=2\cdot 45+3=93$

例 2：给出阶乘函数 F(n)=n!的归纳定义。

解：F(0)=0!=1.

F(n+1)=(n+1)F(n)

例如：F(5)=5F(4)$=5\cdot 4F(3)=5\cdot 4\cdot 3F(2)=5\cdot 4\cdot 3\cdot 2F(1)$

$= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1F(0) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 = 120$ 。

*一个递归定义的函数称为是良好定义的(well defined)，是说：对每一个正整数 n，函数在 n 处的值以无二义的方式确定。

例 4：递归定义以下和式：

$$S_n = \sum_{i=0}^{n} a_i$$

解：$S_0 = \sum_{i=0}^{0} a_i = a_0$

$S_{n+1} = \sum_{i=0}^{n+1} a_i = \sum_{i=0}^{n} a_i + a_{n+1} = S_n + a_{n+1}$

定义 1：Fibonacci 序列$f_0, f_1, f_2, \cdots$递归定义为：$f_0 = 0, f_1 = 1$，且 $f_n = f_{n-1} + f_{n-2}, n = 2,3,4, \cdots$。

例 5：求 Fibonacci 数f_2, f_3, f_4, f_5和f_6。

解：$f_2 = f_1 + f_0 = 1 + 0 = 1$

$f_3 = f_2 + f_1 = 1 + 1 = 2$

$f_4 = f_3 + f_2 = 2 + 1 = 3$

$f_5 = f_4 + f_3 = 3 + 2 = 5$

$f_6 = f_5 + f_4 = 5 + 3 = 8$ 。

例 6：证明：当$n \geq 3$，有$f_n > \alpha^{n-2}$，其中$\alpha = (1 + \sqrt{5})/2$。

解：我们用强归纳法证明这个不等式。设 P(n)表示$f_n > \alpha^{n-2}$。

归纳基础：n=3, 4 时，

$\alpha < 2 = f_3$，$\alpha^2 = \frac{3+\sqrt{5}}{2} < 3 = f_4$

故 P(3), P(4)成立。

归纳步：假设 P(j)成立，即$f_j > \alpha^{j-2}$对所有 j(3≤ j $\leq k$且$k \geq 4$)

成立。欲证 P(k+1)成立，即$f_{k+1} > \alpha^{k-1}$。因为α是方程 $x^2 - x - 1 = 0$的一个解，故有$\alpha^2 = \alpha + 1$。故

$\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1)\alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$

由归纳假设，如果$k \geq 4$，有

$$f_{k-1} > \alpha^{k-3},\quad f_k > \alpha^{k-2}$$

从而有$f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$。

定理 1：(LAME 定理) 设 a, b 是正整数且$a \geq b$。那么用欧几里德算法求 gcd(a, b)所用的除法的次数小于或等于 5 乘以 b 的十进制数的位数。

二．递归定义的集合与结构

*递归定义中的基础步与递归步，排斥规则(见书 P349)。

例 7：考虑整数集合的子集 S，递归定义如下：

基础步：$3 \in S$

递归步：如果$x \in S$且$y \in S$，那么$x + y \in S$。

那么$3 + 3 = 6 \in S, 3 + 6 = 9 \in S, 6 + 6 = 12 \in S$，等等。

例 10：复合命题的合式公式：我们可以用 T, F，命题变量和联结词：¬, ∧, ∨, →, ↔定义复合命题的合式公式：

基础步：T, F 和 s 都是合式公式，其中 s 是一个命题变量。

递归步：如果 E 和 F 是合式公式，那么(¬E), (E∧ F), $(E \vee F), (E \to F), (E \leftrightarrow F)$都是合式公式。

*排斥规则。

由定义知：$(p\vee q),(p\rightarrow F),(q\wedge F),\big((p\vee q)\rightarrow(q\wedge F)\big),$
$\big(q\vee(p\wedge q)\big),((p\rightarrow F)\rightarrow T)$ 都是合式公式，而$p\neg\wedge q, pq\wedge,$
$\neg\wedge pq$都不是合式公式。

定义 4：有根树 (rooted tree)

基础步：一个单独的顶点 r 是一个有根树，r 是该树的树根。

递归步：假设$T_1,T_2,\cdots,T_n$是互不相交的有根树，它们的树根分别为$r_1,r_2,\cdots,r_n$。由一个新顶点 r 和$T_1,T_2,\cdots,T_n$构成一个新的有根树 T，其中 r 与$r_1,r_2,\cdots,r_n$分别连一条边，r 称为 T 的树根。

*见书 P351，图 2

定义 5：一个扩展二叉树(extended binary tree)定义如下：

基础步：一个空集是一个扩展二叉树。

递归步：如果T_1和T_2是两个不相交的扩展二叉树，由新顶点 r 和T_1,T_2构成一个新的扩展二叉树$T=T_1\cdot T_2$，r 与T_1,T_2的树根各连一条边(当T_1,T_2不为空时)。r 称为 T 的根，T_1称为 r 的左子树(left subtree)，T_2称为 r 的右子树(right subtree)。

*见书 P352，图 3.

定义 6：满二叉树(full binary tree)

基础步：一个单独的顶点 r 是一个满二叉树，r 是该树的根。

归纳步：如果T_1和T_2是不相交的满二叉树，由新顶点 r 和T_1,T_2构成一个新的满二叉树$T=T_1\cdot T_2$，其中 r 与左子树T_1的树根及右子树T_2的树根各连一条边。

*见书 P353，图 4

*满二叉树的每个顶点或者有两个孩子，或者没有孩子。

三．结构归纳法

例 12：证明例 7 中的集合 S 的元素是 3 的所有正整数倍的数。

解：设$A=\{3n|n\in Z^{+}\}$。证明：A=S。

首先证：$A\subseteq S$。设 P(n)表示$3n\in S$。

归纳基础：n=1 时，由 S 的定义，$3=3\cdot 1\in S$。

归纳步：假设对正整数 k，有 P(k)成立，即$3k\in S$，欲证$P(k+1)$成立。由于$3\in S$，又由假设有$3k\in S$，由 S 的定义，$3k+3=3(k+1)\in S$。故 P(k+1)成立。

再证：$S\subseteq A$。由 S 的定义，

首先有$3=3\cdot 1\in S$，故$3\in A$。再由归纳步，若$x,y\in S$，有$x+y\in S$，由于$x\in S$和$y\in S$，由归纳假设，x$\in A$且$y\in A$，即$3|x$且$3|y$，由 3.4 节定理 1 的(i)，有$3|(x+y)$，故$x+y\in A$。

*结构归纳法(structural induction)

归纳基础：证明递归定义的基础步中属于集合的所有元素结论都成立。

递归步：证明：如果在定义的递归步中用于构造新元素的元素都满足结论，那么在定义的递归步中所构造出来的新元素也满足结论。

例 13：证明例 10 中定义的合式公式左、右括号的数目相等。

解：

基础步：T, F 和 s 不含括号，故左、右括号数目相等。

递归步：假设 p 和 q 是合式公式，其中左、右括号数目相等，令l_p和l_q分别表示 p 和 q 的左括号数，r_p和r_q分别表示 p 和 q 的右括号数。已知$l_p = r_p$，$l_q = r_q$。那么在(¬p)中，左括号数$= l_p + 1 = r_p + 1 =$右括号数,而在$(p \vee q), (p \wedge q)$, $(p \rightarrow q)$和$(p \leftrightarrow q)$中，左括号数$= l_p + l_q + 1 = r_p + r_q + 1 =$右括号数，故由结构归纳法知，所有合式公式的左、右括号数目相等。

定义 7：定义满二叉树 T 的高度 h(T)如下：

基础步：只有一个根结点 r 的满二叉树 T 的高度 h(T)=0。

递归步：如果T_1, T_2是满二叉树，$T = T_1 \cdot T_2$，那么 T 的高度$h(T) = 1 + \max(h(T_1), h(T_2))$。

设 n(T)表示满二叉树 T 的顶点数，递归定义如下：

基础步：只含一个根结点 r 的满二叉树 T 的顶点数 n(T)=1。

递归步：如果T_1和T_2是满二叉树，且$T = T_1 \cdot T_2$，那么 T 的顶点数$n(T) = n(T_1) + n(T_2) + 1$。

定理 2：如果 T 是满二叉树，那么$n(T) \leq 2^{h(T)+1} - 1$。

证明：我们用结构归纳法证明这个不等式。

基础步：如果满二叉树 T 只含一个根结点 r，那么 n(T)=1，h(T)=0。满足$n(T) = 1 \leq 2^{0+1} - 1 = 1$。

归纳步：假设$T = T_1 \cdot T_2$，T_1和T_2是满二叉树，满足:$n(T_1) \leq 2^{h(T_1)+1} - 1$以及$n(T_2) \leq 2^{h(T_2)+1} - 1$，由 n(T)和 h(T)的递归

公式：

$n(T)=1+n(T_1)+n(T_2), h(T)=1+\max(h(T_1),h(T_2))$,有

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

$\leq 1+\left(2^{h(T_1)+1}-1\right)+(2^{h(T_2)+1}-1)$

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

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

$=2\cdot 2^{h(T)}-1$

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

证毕。

§4.4　递归算法 (Recursive algorithms)

一． 定义：一个算法是递归的，是说，它可以通过求解具有更小的输入的同一问题来解决问题。

二． 例子

例 1：给出计算 n!的递归算法，其中 n 是非负整数。

解：因为$n!=n\cdot(n-1)!$且 0!=1，故设计以下递归算法。

算法 1：计算 n!的递归算法。

PROCEDURE factorial (n: nonnegative integer);

IF n=0 THEN factorial(n) := 1

ELSE factorial(n) $:=n\cdot \mathrm{factorial}(n-1)$;

*计算 4！的步骤：$4!=4\cdot 3!, 3!=3\cdot 2!, 2!=2\cdot 1!, 1!=1\cdot 0!$,

由 0!=1，得 1!=1$\cdot 1=1, 2!=2\cdot 1=2, 3!=3\cdot 2=6, 4!=$

$4 \cdot 6=24$ 。

例 2：给出计算a^n的递归算法，其中 a 是非 0 实数，n 是非负整数。

解：由于$a^n = a \cdot a^{n-1}$，且$a^0 = 1$，得到递归算法。

算法 2：计算a^n的递归算法。

PROCEDURE power (a: nonzero real number, n: nonnegative integer);

IF n=0 THEN power(a, n) := 1

ELSE $\mathrm{power}(a, n) := a \cdot \mathrm{power}(a, n-1)$;

例 3：设计递归算法计算$b^n \bmod m$，其中 b, n 和 m 都是整数，满足: $m \geq 2, n \geq 0$和 $1 \leq b < m$。

解：由递归式 $b^n \bmod m = \left(b \cdot (b^{n-1} \bmod m)\right) \bmod m$,

和 $b^0 \bmod m = 1$, 我们可以得到递归算法。然而，我们有更有效的计算方法。根据递归式：

$b^n \bmod m = (b^{n/2} \bmod m)^2 \bmod m$, 其中 n 是偶数，和

$b^n \bmod m = \left(\left(b^{\lfloor n/2 \rfloor} \bmod m\right)^2 \bmod m \cdot b \bmod m\right) \bmod m$ ，其中 n 是奇数。

算法 3：指数求模运算的递归算法。

PROCEDURE mpower (b, n, m: integers with $m \geq 2, n \geq 0$);

IF n=0 THEN

mpower(b, n, m) := 1

ELSE IF n is even THEN

mpower(b, n, m) := $\text{mpower}(b, \frac{n}{2}, m)^2 \bmod m$

ELSE

mpower(b,n,m):=(mpower$\left(b, \left\lfloor \frac{n}{2} \right\rfloor, m\right)^2 \bmod m \cdot b \bmod m)$

$\bmod m$;

*例如：输入 b=2, n=5, m=3

mpower(2,5,3)=(mpower$(2,2,3)^2 \bmod 3 \cdot 2 \bmod 3) \bmod 3$,

mpower(2,2,3)=mpower$(2,1,3)^2 \bmod 3$, 再算

mpower(2,1,3)=(mpower$(2,0,3)^2 \bmod 3 \cdot 2 \bmod 3) \bmod 3$,

mpower(2,0,3)=1.再往回算,

mpower(2,1,3)=($1^2 \bmod 3 \cdot 2 \bmod 3) \bmod 3$= 2，故

mpower(2,2,3)= $2^2 \bmod 3$ = 1，最后，

mpower(2,5,3)= $(1^2 \bmod 3 \cdot 2 \bmod 3) \bmod 3 = 2$.

例 4：给出递归算法，求两个非负整数 a 和 b 的最大公因子，其中$a < b$。

解：由于我们有递归式：gcd(a, b)=gcd(b mod a, a)以及 gcd(0,b) =b 。 我们得到递归算法：

算法 4：求 gcd(a, b)的递归算法。

PROCEDURE gcd (a, b: nonnegative integers with $a < b$);

IF a = 0 THEN gcd(a, b) := b

ELSE gcd(a, b) := gcd(b mod a, a);

例 5：用递归算法表示线性查找算法。

解：在序列$a_1, a_2, \cdots, a_n$中查找元素 x，在第 i 步，x 和a_i比较，

如果$x = a_i$，则返回 x 的位置 i，如果 x 不等于a_i，则在剩下的序列$a_{i+1}, a_{i+2}, \cdots, a_n$中查找，我们设立递归过程 Search(i,j,x)在$a_i, a_{i+1}, \cdots, a_j$中查找 x。初始输入为：(1,n,x)。递归算法如下：

算法 5：递归的线性查找算法。

```
PROCEDURE Search (i, j, x: integers, 1 ≤ i ≤ j ≤ n);
IF  a_i = x  THEN
   location := i
ELSE IF i = j THEN
   location := 0
ELSE
   Search(i+1, j, x);
```

例 6：构造递归的二分查找算法。

解：在递增序列$a_1, a_2, \cdots, a_n$中查找元素 x。

算法 6：递归的二分查找算法：

```
PROCEDURE binary search(i, j, x: integers, 1 ≤ i ≤ j ≤ n);
BEGIN
   m := ⌊(i + j)/2⌋;
   IF x = a_m  THEN
      location := m
   ELSE IF (x < a_m and i < m) THEN
      binary search(i, m − 1, x)
```

ELSE IF $(x > a_m \text{ and } j > m)$ THEN

binary search $(m+1, j, x)$

ELSE location := 0;

END

三．证明算法的正确性

例 7：证明：计算实数 a 的幂的算法 2 是正确的。

解：我们对指数 n 用数学归纳法证明。

当 n=0 时，算法第一步求得 power(a, 0)=1，因为$a^0 = 1\ (a \neq 0)$，故算法正确。

设当 n=k 时，算法计算正确，即 power(a, k)=a^k。

当 n=k+1 时，算法求得 power(a,k+1)= $a \cdot \text{power}(a,k)$，因为 $a^{k+1} = a \cdot a^k$，又由归纳假设，power(a,k+1)= $a \cdot \text{power}(a,k)$ $= a \cdot a^k = a^{k+1}$，故算法正确。

例 8：证明：指数取模运算的算法 3 正确。

解：我们对指数 n 用强归纳法证明。

设 b 和 m 是整数且$m \geq 2$。

当 n=0 时，算法求得 mpower(b,n,m)=1，因为$b^0 \bmod m = 1$，这一步正确。

设当$0 \leq j < k$时，mpower(b,j,m) = $b^j \bmod m$。

现在设 n=k。当 k 是偶数时，根据归纳假设，有

mpower(b,k,m)=mpower$\left(b, \frac{k}{2}, m\right)^2 \bmod m = \left(b^{\frac{k}{2}} \bmod m\right)^2 \bmod m = b^k \bmod m$.

当 k 是奇数时，由归纳假设，有

mpower(b,k,m)= $\left(\mathrm{mpower}\left(b,\left\lfloor\frac{k}{2}\right\rfloor,m\right)^2 \bmod m\cdot b \bmod m\right)$

$\bmod m \;=\left(\left(b^{\left\lfloor\frac{k}{2}\right\rfloor} \bmod m\right)^2 \bmod m\cdot b \bmod m\right)\bmod m$

$=b^{2\lfloor k/2\rfloor+1}\bmod m=b^k\bmod m$.

故算法正确。

四．递归和重复(循环)(recursion and iteration)

算法 7：求 Fibonacci 数的递归算法

```
PROCEDURE fibonacci (n: nonnegative integer);
IF n = 0 THEN fibonacci(0) := 0
ELSE IF n = 1 THEN fibonacci(1) := 1
ELSE fibonacci(n) := fibonacci(n − 1) + fibonacci(n − 2);
```

*在递归的计算中，由于$f_n=f_{n-1}+f_{n-2}$，某些f_i的计算有重复。例如：计算f_4，其中f_2被重复计算(见书 P366，图 1).

*用非递归的循环计算，有时可以消除重复计算。

算法 8：用循环计算 Fibonacci 数。

```
PROCEDURE iterative fibonacci (n: nonnegative integer);
IF n = 0 THEN y := 0
ELSE BEGIN
   x := 0;
   y := 1;
   FOR i := 1 TO  n − 1  DO
```

```
    BEGIN
        z := x+y;
        x := y;
        y := z;
    END
END;
```

{ y is the nth Fibonacci number}.

作业：

1. 给出以下序列$\{a_n\}$的递归定义：

(a)$a_n = 4n - 2$; (b)$a_n = n(n+1)$; (c)$a_n = 1 + (-1)^n$;

2. 用结构归纳法证明：n(T)≥ $2h(T) + 1$，其中 T 是一棵满二叉树，n(T)是 T 的顶点数，并且 h(T)是 T 的高度。

3. 设计一个递归算法计算n^2，其中 n 是非负整数. 然后证明此算法是正确的。

4. 设计递归算法计算以下序列中第 k 项(k 是非负整数)：

$a_0 = 1, a_1 = 2$，且$a_n = a_{n-1} \cdot a_{n-2} (n = 2,3,\cdots)$。然后设计一个循环算法，计算该序列的第 k 项。