蘇州大學
SOOCHOW UNIVERSITY

Chap5 Induction and Recursion

Part II: Recursive Algorithms

Jin-Hui Wu

2026-04-03

大纲

- 数学归纳法
- 强归纳法
- 递归定义和结构归纳
- 递归算法 (5.4)

递归算法

- 递归算法 (recursively algorithm)
  - 把问题归约到输入规模更小的相同问题的算法
  - 当输入规模足够小时，可以直接给出解

递归算法

- 阶乘的计算
  - 递归：$n! = n \cdot (n-1)!$
  - 终止条件：$0! = 1$

```
procedure factorial(n: nonnegative integer)
if n = 0 then
   return 1
else
   return n·factorial (n − 1) {output is n!}
```

- 递归深度：递归函数嵌套调用的最大层数
- $O(n)$

递归算法

- 最大公约数的计算
  - 递归：$\gcd(a, b) = \gcd(b \bmod a, a)$
  - 终止条件：$\gcd(0, b) = b$

procedure *gcd*(*a*,*b*: nonnegative integers with $a < b$)
if $a = 0$ **then**
  return *b*
else
  return *gcd* (*b* **mod** *a*, *a*) {output is *gcd*(*a*, *b*)}

- 递归深度$O(\log b)$

递归算法

- 模幂算法
  - 递归：$b^n \bmod m = (b^{\lceil \frac{n}{2} \rceil} \bmod m)(b^{\lfloor \frac{n}{2} \rfloor} \bmod m) \bmod m$
  - 终止条件：$b^0 \bmod m = 1$

procedure *mpower*(b,m,n: integers with $b > 0$ and $m \geq 2$, $n \geq 0$)
if $n = 0$ **then**
  return 1
else if *n is even* **then**
  return $mpower(b,n/2,m)^2$ **mod** m
else
  return $(mpower(b,\lfloor n/2 \rfloor,m)^2$ **mod** $m \cdot b$ **mod** $m)$ **mod** m
{output is b^n **mod** m}

- 递归深度$O(\log n)$

递归算法

- 归并排序 (merge sort)
  - 将序列拆成两部分，各自排序后再合并

例：归并排序

- 排序：8, 2, 4, 6, 9, 7

递归算法

- 归并排序

拆分

procedure *mergesort*($L = a_1, a_2,\ldots,a_n$)
if $n > 1$ **then**
  $m := \lfloor n/2 \rfloor$
  $L_1 := a_1, a_2,\ldots,a_m$
  $L_2 := a_{m+1}, a_{m+2},\ldots,a_n$
  L := *merge*(*mergesort*(L_1), *mergesort*(L_2))
{L is now sorted into elements in increasing order}

- 时间复杂度$O(n)$

合并

procedure *merge*(L_1, L_2 :sorted lists)
L := empty list
while L_1 and L_2 are both nonempty
  remove smaller of first elements of L_1 and L_2 from its list;
  put at the left end of L
  if this removal makes one list empty
    then remove all elements from the other list and append them to L
return L {L is the merged list with the elements in increasing order}

- 时间复杂度$O(n \log n)$

递归算法和迭代算法

- 斐波那契数的递归算法

ALGORITHM 7 A Recursive Algorithm for Fibonacci Numbers.

procedure *fibonacci*(n: nonnegative integer)
if $n = 0$ **then return** 0
else if $n = 1$ **then return** 1
else return $fibonacci(n-1) + fibonacci(n-2)$
{output is $fibonacci(n)$}

- 时间复杂度$O(f_n)$

递归算法和迭代算法

- 斐波那契数的迭代算法 (iterative algorithm)
  - 用循环替代调用自身

ALGORITHM 8 An Iterative Algorithm for Computing Fibonacci Numbers.

```
procedure iterative fibonacci(n: nonnegative integer)
if n = 0 then return 0
else
    x := 0
    y := 1
    for i := 1 to n − 1
        z := x + y
        x := y
        y := z
    return y
{output is the nth Fibonacci number}
```

- 时间复杂度$O(n)$