

algorithm

2.时间复杂度与伪码

2.1 问题及实例

- 问题
 - 需要回答的一般性提问，通常含若干参数
- 问题描述
 - 定义问题参数(集合,变量,函数,序列等)
 - 说明每个参数的取值范围及参数间的关系
 - 定义问题的解
 - 说明解满足的条件(优化目标或约束条件)
- 问题实例
 - 参数的一组赋值可得到问题的一个实例

算法

- 算法
 - 有限条指令的序列
 - 这个指令序列确定了解决某个问题的一系列运算或操作
- 算法 A 解问题 P
 - 把问题 P 的任何实例作为算法 A 的输入
 - 每步计算是确定性的
 - A 能够在有限步停机
 - 输出该实例的正确的解

基本运算与输入规模

- 算法时间复杂度: 针对指定基本运算, 计数算法所做运算次数
- 基本运算: 比较, 加法, 乘法, 置指针, 交换...
- 输入规模: 输入串编码长度, 通常用下述参数度量:
 - 数组元素多少
 - 调度问题的任务个数
 - 图的顶点数与边数等
- 算法基本运算次数可表为输入规模的函数
- 给定问题和基本运算就决定了一个算法类

输入规模

- 排序：数组中元素个数 n
- 检索：被检索数组的元素个数 n
- 整数乘法：两个整数的位数 m, n
- 矩阵相乘：矩阵的行列数 i, j, k
- 图的遍历：图的顶点数 n , 边数 m

基本运算

- 排序: 元素之间的比较
- 检索: 被检索元素 x 与数组元素的比较
- 整数乘法:
 - 每位数字相乘(位乘) 1 次
 - m 位和 n 位整数相乘要做 mn 次位乘
- 矩阵相乘:
 - 每对元素乘 1 次
 - $i \times j$ 矩阵与 $j \times k$ 矩阵相乘要做 ijk 次乘法
- 图的遍历: 置指针

算法的两种时间复杂度

- 对于相同输入规模的不同实例，算法的基本运算次数也不一样，可定义两种时间复杂度
- **最坏情况下**的时间复杂度 $W(n)$
 - 算法求解输入规模为 n 的实例所需要的最长时间
- **平均情况下**的时间复杂度 $A(n)$
 - 在给定同样规模为 n 的输入实例的概率分布下，算法求解这些实例所需要的平均时间

A(n) 计算公式

- 平均情况下的时间复杂度 $A(n)$
- 设 S 是规模为 n 的实例集
- 实例 $I \in S$ 的概率是 P_i
- 算法对实例 I 执行的基本运算次数是 t_i

$$A(n) = \sum_{I \in S} P_I t_I$$

- 在某些情况下可以假定每个输入实例概率相等

例子：检索

- 检索问题
- 输入：
 - 非降顺序排列的数组 L
 - 元素数 n
 - 数 x
- 输出： j
 - 若 x 在 L 中, j 是 x 首次出现的下标
 - 否则 $j = 0$
- 基本运算：
 - x 与 L 中元素的比较

顺序检索算法

- $j=1$, 将 x 与 $L[j]$ 比较:
 - 如果 $x=L[j]$, 则算法停止, 输出 j ;
 - 如果不等, 则把 j 加1, 继续 x 与 $L[j]$ 的比较, 如果 $j>n$, 则停机并输出0

• 实例

1	2	3	4	5
---	---	---	---	---

- $x = 4$, 需要比较 次
- $x = 2.5$, 需要比较 次

最坏情况的时间估计

- 不同的输入有 个, 分别对应:

$$x = L[1], x = L[2], \dots, x = L[n]$$

$$x < L[1], L[1] < x < L[2], L[2] < x < L[3], \dots, L[n] < x$$

- 最坏情况下时间:

- $W(n) = n$

- 最坏的输入:

- x 不在 L 中或 $x = L[n]$ 要做 n 次比较

平均情况的时间估计

- 输入实例的概率分布：
- 假设 x 在 L 中概率是 p ，且每个位置概率相等

$$\begin{aligned} A(n) &= \sum_{i=1}^n i \frac{p}{n} + (1-p)n \\ &= \frac{p(n+1)}{2} + (1-p)n \end{aligned}$$

当 $p=1/2$ 时，

$$A(n) = \frac{n+1}{4} + \frac{n}{2} \approx \frac{3n}{4}$$

改进顺序检索算法

- $j=1$, 将 x 与 $L[j]$ 比较:
 - 如果 $x=L[j]$, 则算法停止, 输出 j ;
 - 如果 $x>L[j]$, 则把 j 加1, 继续 x 与 $L[j]$ 的比较;
 - 如果 $x<L[j]$, 则停机并输出0;
 - 如果 $j > n$, 则停机并输出 0.

- 实例

1	2	3	4	5
---	---	---	---	---

- $x = 4$, 需要比较 次
- $x = 2.5$, 需要比较 次
- 前提: 数组需要是排好序的

时间估计

- 最坏情况下: $W(n) = n$
- 平均情况下:
 - 输入实例的概率分布: 假设 x 在 L 中每个位置与空隙的概率都相等



改进检索算法平均时间复杂度是多少?

- 小结:
 - 算法最坏和平均情况下的时间复杂度定义
 - 如何计算上述时间复杂度

2.2 算法的伪码描述

- 赋值语句：←
- 分支语句：if ...then ... [else...]
- 循环语句：while, for, repeat until
- 转向语句：goto
- 输出语句：return
- 调用：直接写过程的名字
- 注释：//...

例：求最大公约数

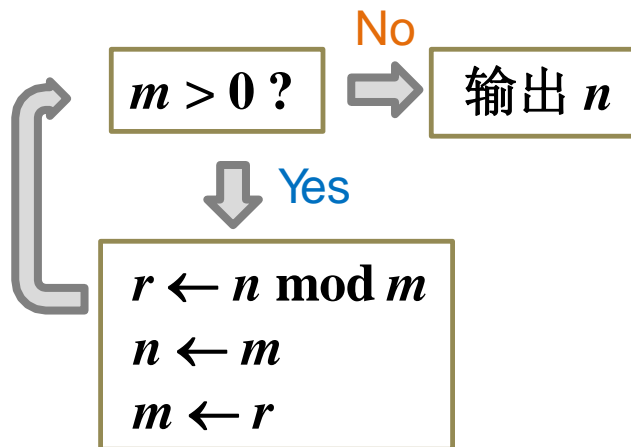
- 算法 $\text{Euclid}(m, n)$
- 输入：非负整数 m, n , 其中 m 与 n 不全为 0
- 输出： m 与 n 的最大公约数

1. while $m > 0$ do
2. $r \leftarrow n \bmod m$
3. $n \leftarrow m$
4. $m \leftarrow r$
5. return n

运行实例： $n=36, m=15$

while	n	m	r
第1次	36	15	6
第2次	15	6	3
第3次	6	3	0
	3	0	0

↓
输出3



例：改进的顺序检索

- 算法 **Search (L, x)**
- 输入：数组 $L[1..n]$, 元素从小到大排列, 数 x .
- 输出：
 - 若 x 在 L 中, 输出 x 的位置下标 j
 - 否则输出 0

1. $j \leftarrow 1$
2. **while** $j \leq n$ **and** $x > L[j]$ **do** $j \leftarrow j+1$
3. **if** $x < L[j]$ **or** $j > n$ **then** $j \leftarrow 0$
4. **return** j

例：插入排序

- 算法 Insert Sort (A, n)
- 输入： n 个数的数组 A
- 输出： 按照递增顺序排好序的数组 A

```
1. for  $j \leftarrow 2$  to  $n$  do
2.    $x \leftarrow A[j]$ 
3.    $i \leftarrow j-1$    //3-7 行把  $A[j]$  插入  $A[1..j-1]$ 
4.   while  $i > 0$  and  $x < A[i]$  do
5.      $A[i+1] \leftarrow A[i]$ 
6.      $i \leftarrow i-1$ 
7.    $A[i+1] \leftarrow x$ 
```

运行实例

2	4	1	5	3
---	---	---	---	---

$j = 3, x = A[3] = 1$

$i = 2, A[2] = 4$

$i > 0, x < A[2] \quad \checkmark$

2	4	4	5	3
---	---	---	---	---

$A[3] = 4, i = 1, x = 1$

$i > 0, x < A[1] \quad \checkmark$

2	2	4	5	3
---	---	---	---	---

$A[2] = 2, i = 0, x = 1$

$i > 0 \quad \times$

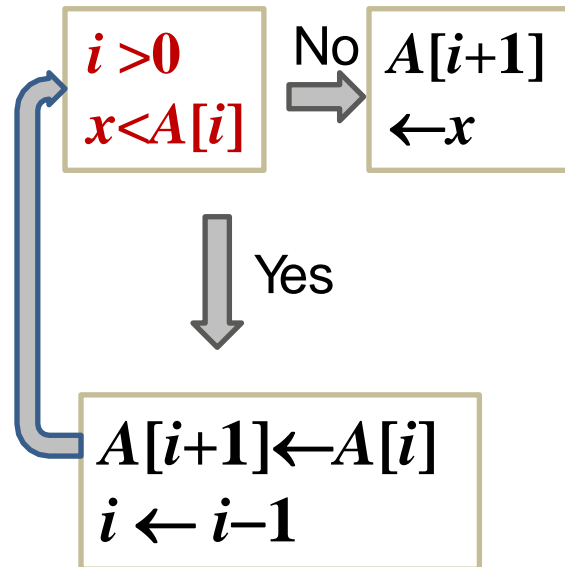
1	2	4	5	3
---	---	---	---	---

4. while $i > 0$ and $x < A[i]$ do

5. $A[i+1] \leftarrow A[i]$

6. $i \leftarrow i-1$

7. $A[i+1] \leftarrow x$



例：二分归并排序

- MergeSort (A, p, r)
 - 输入：数组 $A[p..r]$
 - 输出：按递增顺序排序的数组 A
-
1. if $p < r$
 2. then $q \leftarrow \lfloor (p+r)/2 \rfloor$
 3. MergeSort (A, p, q)
 4. MergeSort ($A, q+1, r$)
 5. Merge (A, p, q, r)
-
- MergeSort有递归调用,也调用Merge过程

例：算法A的伪码

- 算法 A
- 输入：实数的数组 $P[0..n]$, 实数 x
- 输出： y

1. $y \leftarrow P[0]; \text{ power} \leftarrow 1$
2. for $i \leftarrow 1$ to n do
3. $\text{power} \leftarrow \text{power} * x$
4. $y \leftarrow y + P[i] * \text{power}$
5. return y

例：算法A的伪码

对 $i = 1, 2, \dots, n \Rightarrow$

$$\begin{aligned} power &\leftarrow power * x \\ y &\leftarrow y + P[i] * power \end{aligned}$$

	i	$power$	y
初值		1	$P[0]$
循环	1	x	$P[0] + P[1]*x$
	2	x^2	$P[0] + P[1]*x + P[2]*x^2$
	3	x^3	$P[0] + P[1]*x + P[2]*x^2 + P[3]*x^3$
			...

输入 $P[0..n]$ 是 n 次多项式 $P(x)$ 的系数
算法 A 计算该多项式在 x 的值

小结

- 用伪码表示算法
- 伪码不是程序代码，只是给出算法的主要步骤
- 伪码中有哪些关键字？
- 伪码中允许过程调用

2.3 函数渐进界

- 大O符号
- 定义：设 f 和 g 是定义域为自然数集 N 上的函数
- 若存在正数 c 和 n_0 ，使得对一切 $n \geq n_0$ 有
 - $0 \leq f(n) \leq cg(n)$
- 成立，则称 $f(n)$ 的渐近的上界是 $g(n)$ ，记作
 - $f(n) = O(g(n))$

例子

- 设 $f(n) = n^2 + n$, 则
 - $f(n) = O(n^2)$, 取 $c = 2$, $n_0 = 1$ 即可
 - $f(n) = O(n^3)$, 取 $c = 1$, $n_0 = 2$ 即可
1. $f(n) = O(g(n))$, $f(n)$ 的阶不高于 $g(n)$ 的阶
 2. 可能存在多个正数 c , 只要指出一个即可
 3. 对前面有限个值可以不满足不等式
 4. 常函数可以写作 $O(1)$

大 Ω 符号

- 定义：设 f 和 g 是定义域为自然数集 N 上的函数
- 若存在正数 c 和 n_0 ，使得对一切 $n \geq n_0$ ，有

$$0 \leq cg(n) \leq f(n)$$

- 成立，则称 $f(n)$ 的渐近的下界是 $g(n)$
- 记作

$$f(n) = \Omega(g(n))$$

例子

• 设 $f(n) = n^2 + n$, 则

$f(n) = \Omega(n^2)$, 取 $c = 1, n_0 = 1$ 即可

$f(n) = \Omega(100n)$, 取 $c = 1/100, n_0 = 1$ 即可

1. $f(n) = \Omega(g(n))$, $f(n)$ 的阶不低于 $g(n)$ 的阶
2. 可能存在多个正数 c , 指出一个即可
3. 对前面有限个 n 值可以不满足上述不等式

小o符号

- 定义: 设 f 和 g 是定义域为自然数集 N 上的函数
- 若对于任意正数 c 都存在 n_0 , 使得对一切 $n \geq n_0$ 有

$$0 \leq f(n) < c g(n)$$

- 成立, 则记作

$$f(n) = o(g(n))$$

例子

- 例子: $f(n)=n^2+n$, 则

$$f(n)=o(n^3)$$

- $c \geq 1$ 显然成立, 因为 $n^2+n < cn^3$ ($n_0=2$)
- 任给 $1 > c > 0$, 取 $n_0 > \lceil 2/c \rceil$ 即可, 因为

$$cn \geq cn_0 > 2 \quad (\text{当 } n \geq n_0)$$

$$n^2+n < 2n^2 < cn^3$$

1. $f(n) = o(g(n))$, $f(n)$ 的阶低于 $g(n)$ 的阶
2. 对不同正数 c , n_0 不一样, c 越小 n_0 越大
3. 对前面有限个 n 值可以不满足不等式

小 ω 符号

- 定义：设 f 和 g 是定义域为自然数集 \mathbb{N} 上的函数
- 若对于任意正数 c 都存在 n_0 ，使得对一切 $n \geq n_0$ ，有
 - $0 \leq cg(n) < f(n)$
- 成立, 则记作
 - $f(n) = \omega(g(n))$

例子

- 设 $f(n) = n^2 + n$, 则

$$f(n) = \omega(n),$$

- 不能写 $f(n) = \omega(n^2)$, 因为取 $c = 2$, 不存在 n_0 使得对一切 $n \geq n_0$ 有下式成立

$$c n^2 = 2n^2 < n^2 + n$$

1. $f(n) = \omega(g(n))$, $f(n)$ 的阶高于 $g(n)$ 的阶
2. 对不同的正数 c , n_0 不等, c 越大 n_0 越大
3. 对前面有限个 n 值可以不满足不等式

Θ 符号

- 若 $f(n) = O(g(n))$ 且 $f(n) = \Omega(g(n))$,
- 则记作

$$f(n) = \Theta(g(n))$$

- 例子: $f(n) = n^2 + n$, $g(n) = 100n^2$, 那么有

$$f(n) = \Theta(g(n))$$

1. $f(n)$ 的阶与 $g(n)$ 的阶相等
2. 对前面有限个 n 值可以不满足条件

例子：素数测试

- 算法PrimalityTest(n)
 - 输入： n , 大于2的奇整数
 - 输出： true 或者 false
1. $s \leftarrow \lfloor n^{1/2} \rfloor$
 2. for $j \leftarrow 2$ to s
 3. if j 整除 n
 4. then return false
 5. return true

思考：

若 $n^{1/2}$ 可在 $O(1)$ 计算，
基本运算是整除，以下表示是否正确？

$$W(n) = O(n^{1/2})$$

$$W(n) = \theta(n^{1/2})$$

小结

- 五种表示函数的阶的符号
 - $O, \Omega, o, \omega, \Theta$
- 如何用定义证明函数的阶?

2.4 有关函数渐进界的定理

- 定理1：设 f 和 g 是定义域为自然数集合的函数
- (1) 如果 $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$ 存在, 并且等于某个常数 $c > 0$, 那么

$$f(n) = \theta(g(n))$$

- (2) 如果 $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$, 那么

$$f(n) = o(g(n))$$

- (3) 如果 $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = +\infty$, 那么

$$f(n) = \omega(g(n))$$

定理1证明

根据极限定义，对于给定正数 ε 存在某个 n_0 ，
只要 $n \geq n_0$ ，就有

$$|f(n)/g(n) - c| < \varepsilon$$

$$c - \varepsilon < f(n)/g(n) < c + \varepsilon$$

取 $\varepsilon = c/2$ ，

$$c/2 < f(n)/g(n) < 3c/2 < 2c$$

对所有 $n \geq n_0$ ， $f(n) \leq 2cg(n)$ ，于是 $f(n) = O(g(n))$ ；

对所有 $n \geq n_0$ ， $f(n) \geq (c/2)g(n)$ ，于是 $f(n) = \Omega(g(n))$ 。

从而 $f(n) = \Theta(g(n))$ 。

例：估计函数的阶

例1 设 $f(n) = \frac{1}{2}n^2 - 3n$, 证明 $f(n) = \Theta(n^2)$.

证 因为

$$\lim_{n \rightarrow \infty} \frac{f(n)}{n^2} = \lim_{n \rightarrow \infty} \frac{\frac{1}{2}n^2 - 3n}{n^2} = \frac{1}{2}$$

根据定理1, 有 $f(n) = \Theta(n^2)$

多项式函数的阶低于指数函数的阶

$$n^d = o(r^n), \quad r > 1, \quad d > 0$$

• 证：不妨设 d 为正整数，

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{n^d}{r^n} &= \lim_{n \rightarrow \infty} \frac{dn^{d-1}}{r^n \ln r} = \lim_{n \rightarrow \infty} \frac{d(d-1)n^{d-2}}{r^n (\ln r)^2} \\ &= \dots = \lim_{n \rightarrow \infty} \frac{d!}{r^n (\ln r)^d} = 0 \end{aligned}$$

对数函数的阶低于幂函数的阶

$$\ln n = o(n^d), \quad d > 0$$

• 证:

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{\ln n}{n^d} &= \lim_{n \rightarrow \infty} \frac{\frac{1}{n}}{d n^{d-1}} \\ &= \lim_{n \rightarrow \infty} \frac{1}{d n^d} = 0 \end{aligned}$$

定理 2

- 定理：设函数 f, g, h 的定义域为自然数集合，
 - (1) 如果 $f = O(g)$ 且 $g = O(h)$ ，那么 $f = O(h)$
 - (2) 如果 $f = \Omega(g)$ 且 $g = \Omega(h)$ ，那么 $f = \Omega(h)$
 - (3) 如果 $f = \Theta(g)$ 和 $g = \Theta(h)$ ，那么 $f = \Theta(h)$
- 函数的阶之间的关系具有传递性

定理2例子

- 按照阶从高到低排序以下函数：

$$f(n)=(n^2+n)/2, \quad g(n)=10n \quad h(n)=1.5^n, \quad t(n)=n^{1/2}$$

$$h(n) = \omega(f(n)),$$

$$f(n) = \omega(g(n)),$$

$$g(n) = \omega(t(n)),$$

- 排序 $h(n), f(n), g(n), t(n)$

定理3

- 定理：假设函数 f 和 g 的定义域为自然数集，若对某个其它函数 h ，有 $f=O(h)$ 和 $g=O(h)$ ，
- 那么 $f + g = O(h)$
- 该性质可以推广到有限个函数
- 算法由有限步骤构成，若每一步的时间复杂度函数的上界都是 $h(n)$ ，那么该算法的时间复杂度函数可以写作 $O(h(n))$

2.5 几类重要的函数

- 基本函数类
- 阶的高低
- 至少指数级: $2^n, 3^n, n!, \dots$
- 多项式级: $n, n^2, n \log n, n^{1/2}, \dots$
- 对数多项式级: $\log n, \log^2 n, \log \log n, \dots$

对数函数

- 符号:

- $\log n = \log_2 n$

- $\log^k n = (\log n)^k$ $\log \log n = \log(\log n)$

- 性质:

- (1) $\log_2 n = \Theta(\log_l n)$

- (2) $\log_b n = o(n^\alpha)$ $\alpha > 0$

- (3) $a^{\log_b n} = n^{\log_b a}$

性质(1)的证明

$$\log_k n = \frac{\log_l n}{\log_l k} \quad \log_l k \text{ 为常数}$$

$$\lim_{n \rightarrow \infty} \frac{\log_k n}{\log_l n} = \lim_{n \rightarrow \infty} \frac{\log_l n}{\log_l k \cdot \log_l n} = \frac{1}{\log_l k}$$

根据定理, $\log_k n = \Theta(\log_l n)$

性质(2)(3)的说明

$$\log_b n = \Theta(\ln n)$$

$$\ln n = o(n^\alpha) \quad \Rightarrow \quad \log_b n = o(n^\alpha) \quad \alpha > 0$$

$$a^{\log_b n} = n^{\log_b a}$$



$$\log_b n \log_b a = \log_b a \log_b n$$

指数函数与阶乘

Stirling公式
$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

$$n! = o(n^n)$$

$$n! = \omega(2^n)$$

$$\log(n!) = \Theta(n \log n)$$

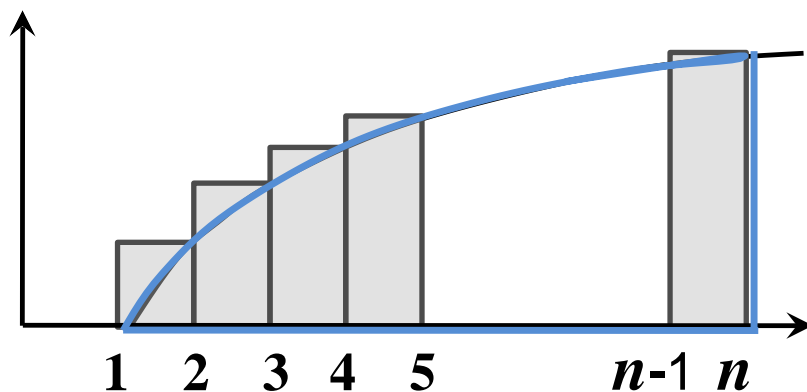
应用：估计搜索空间大小

$$C_{m+n-1}^m = \frac{(m+n-1)!}{m!(n-1)!}$$

$$= \frac{\sqrt{2\pi(m+n-1)}(m+n-1)^{m+n-1}(1+\Theta(\frac{1}{m+n-1}))}{\sqrt{2\pi m}m^m(1+\Theta(\frac{1}{m}))\sqrt{2\pi(n-1)}(n-1)^{n-1}(1+\Theta(\frac{1}{n-1}))}$$

$$= \Theta((1+\varepsilon)^{m+n-1})$$

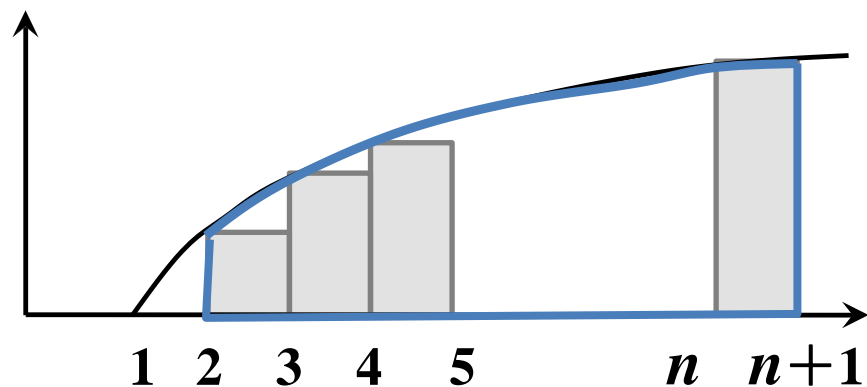
$\log(n!) = \Omega(n \log n)$ 的证明



$$\log(n!) = \sum_{k=1}^n \log k \geq \int_1^n \log x dx$$

$$= \log(n \ln n - n + 1) = \Omega(n \log n)$$

$\log(n!) = O(n \log n)$ 的证明



$$\log(n!) = \sum_{k=1}^n \log k \leq \int_2^{n+1} \log x dx = O(n \log n)$$

取整函数

- 取整函数的定义
 - $\lfloor x \rfloor$: 表示小于等于 x 的最大的整数
 - $\lceil x \rceil$: 表示大于等于 x 的最小的整数
- 实例
 - $\lfloor 2.6 \rfloor = 2$
 - $\lceil 2.6 \rceil = 3$
 - $\lfloor 2 \rfloor = \lceil 2 \rceil = 2$

取整函数的性质

$$(1) \ x-1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x+1$$

$$(2) \ \lfloor x+n \rfloor = \lfloor x \rfloor + n, \ \lceil x+n \rceil = \lceil x \rceil + n, \ n \text{ 为整数}$$

$$(3) \ \left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor = n$$

$$(4) \ \left\lceil \frac{\left\lceil \frac{n}{a} \right\rceil}{b} \right\rceil = \left\lceil \frac{n}{ab} \right\rceil, \quad \left\lfloor \frac{\left\lfloor \frac{n}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$$

证明(1)

如果 x 是整数 n , 根据定义

$$\lfloor x \rfloor = \lceil x \rceil = n,$$

$$x-1 < \lfloor x \rfloor = x = \lceil x \rceil < x+1$$

如果 $n < x < n+1$, n 为整数, 那么

$$\lfloor x \rfloor = n, \lceil x \rceil = n+1,$$

从而有

$$x-1 < n = \lfloor x \rfloor, \quad n < x < n+1 = \lceil x \rceil$$

$$\Rightarrow x-1 < n = \lfloor x \rfloor < x < \lceil x \rceil = n+1 < x+1$$

例：按照阶排序

$\log(n!), \log^2 n, 1, n!, n2^n, n^{1/\log n},$

$(3/2)^n, \sqrt{\log n}, (\log n)^{\log n}, 2^{2^n},$

$n^{\log \log n}, n^3, \log \log n, n \log n, n,$

$2^{\log n}, \log n$

排序结果

$$2^{2^n}, n!, n2^n, (3/2)^n, (\log n)^{\log n} = n^{\log \log n},$$

$$n^3, \log(n!) = \Theta(n \log n), n = 2^{\log n},$$

$$\log^2 n, \log n, \sqrt{\log n}, \log \log n,$$

$$n^{1/\log n} = 1$$

作业

- 1、请分析并写出插入排序、冒泡排序、快速排序在最坏情况和平均情况下的时间复杂度
- 2、请用伪码表示冒泡排序

- 2.1 算法的时间复杂度
 - 2.1.1 算法基本运算与输入规模
 - 2.1.2 算法的两种时间复杂度
- 2.2 算法的伪码表示
 - 2.2.1 数据集的线性可分性
 - 2.2.2 求最大公约数
 - 2.2.3 插入排序
- 2.3 函数的渐进的界
 - 2.3.1 大O和小o符号
 - 2.3.2 大 Ω 和小 ω 符号
 - 2.3.3 θ 符号
- 2.4 有关函数渐近的界的定理
 - 2.4.1 定理1
 - 2.4.2 定理2
 - 2.4.3 定理3
- 2.5 几类重要的函数
 - 2.5.1 基本函数
 - 2.5.2 对数函数
 - 2.5.3 指数函数
 - 2.5.4 取整函数