课程回顾

- 算法是什么？
  - 良定义、合法输入、输出、有穷步骤
- 算法 vs. 程序
  - 算法：输入、输出、确定性、有限性（正确性前提）
  - 程序：算法的具体实现、可不满足有限性
- 问题求解步骤
  - 理解问题、精确解近似解/数据结构/算法设计策略、设计算法、证明正确性、分析算法、设计程序
- 算法描述及算法重要性
- 插入排序分析
  - 正确性分析：循环不变式
  - 算法执行时间：最好情况、最坏情况、平均情况

插入排序的正确性证明

```
INSERTION_SORT(A)
for j ← 2 to A.length do
    key ← A[j]
    // Insert A[j] into the sorted sequence A[1..j-1]
    i ← j - 1
    while i>0 and A[i]>key do
        A[i+1] ← A[i]
        i ← i - 1
    A[i+1] ← key
```

循环不变式：子数组$A[1..j\text{-}1]$有序

- 初始化：首先证明在第一次循环迭代之前 (当j=2时)，循环不变式成立。子数组$A[1..j\text{-}1]$仅由单个元素$A[1]$组成，实际上就是$A[1]$中原来的元素。而且该子数组是排序好的 (当然很平凡)。这表明第一次循环迭代之前循环不变式成立。

插入排序的正确性证明 (续)

```
INSERTION_SORT(A)
1  for j ← 2 to A.length do
2      key ← A[j]
3      // Insert A[j] into the sorted sequence A[1..j-1]
4      i ← j - 1
5      while i>0 and A[i]>key do
6          A[i+1] ← A[i]
7          i ← i - 1
8      A[i+1] ← key
```

循环不变式：
子数组$A[1..j\text{-}1]$有序

■ **保持：** 非形式化地，for循环体的第4~7行将$A[j\text{-}1]$、$A[j\text{-}2]$、$A[j\text{-}3]$等向右移动一个位置，直到找到$A[j]$的适当位置，第8行将$A[j]$的值插入该位置。这时子数组$A[1..j]$由原来在$A[1..j]$中的元素组成，但已按序排列。那么对for循环的下一次迭代增加j将保持循环不变式。

插入排序的正确性证明 (续)

```
INSERTION_SORT(A)
for j ← 2 to A.length do
    key ← A[j]
    // Insert A[j] into the sorted sequence A[1..j-1]
    i ← j - 1
    while i>0 and A[i]>key do
        A[i+1] ← A[i]
        i ← i - 1
    A[i+1] ← key
```

循环不变式：
子数组$A[1..j-1]$有序

- **终止：** 导致for循环终止的条件是$j>A.length=n$。因为每次循环迭代j增加1，那么必有$j=n+1$。在循环不变式的表述中将j用$n+1$代替，我们有：子数组$A[1..n]$由原来在$A[1..n]$中的元素组成，但已按序排列。注意到，子数组$A[1..n]$就是整个数组，我们推断出整个数组已排序。因此算法正确。

插入排序的正确性证明 (续)

```
INSERTION_SORT(A)
1 for j ← 2 to A.length do
2     key ← A[j]
3     // Insert A[j] into the sorted sequence A[1..j-1]
4     i ← j - 1
5     while i>0 and A[i]>key do
6         A[i+1] ← A[i]
7         i ← i - 1
8     A[i+1] ← key
```

5-7行循环不变式：子数组$A[i+2..j]$中的元素值都大于key值且有序

- 注：第二条性质的一种更形式化的处理要求我们对第5~7行的while循环给出并证明一个循环不变式
- 非形式化的分析来证明第二条性质对外层循环成立
- 思考：第5~7行while循环的循环不变式是什么？

插入排序算法分析

	代价	次数
INSERTION_SORT(A)		
1 **for** $j \leftarrow 2$ **to** $A.length$ **do**	c_1	n
2 $key \leftarrow A[j]$	c_2	n-1
3 // Insert $A[j]$ into the sorted sequence $A[1..j\text{-}1]$	0	n-1
4 $i \leftarrow j$ - 1	c_4	n-1
5 **while** i>0 and $A[i]$>key **do**	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i \leftarrow i$ - 1	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] \leftarrow key$	c_8	n-1

t_j：对于值j，第5行执行while循环测试的次数

插入排序算法分析 (续)

■算法执行时间

$$
\begin{aligned}
T(n) = & c_1 n + c_2(n-1) + c_4(n-1) \\
& + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)
\end{aligned}
$$

➢最好情况：顺序排列——$t_j = 1$

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\
&= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \\
&= \Theta(n)
\end{aligned}
$$

- 执行时间是*n*的线性函数

插入排序算法分析 (续)

■算法执行时间

➢最坏情况：逆序排列

$$\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

$$\sum_{j=2}^{n} (t_j - 1) = \sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

$$\begin{aligned}
T(n) =& c_1 n + c_2(n-1) + c_4(n-1) \\
& + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
=& \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n - (c_2 + c_4 + c_5 + c_8) \\
=& \Theta(n^2)
\end{aligned}$$

- 执行时间是n的二次函数

插入排序算法分析 (续)

■算法执行时间

- **平均情况**：在$A[1..j\text{-}1]$中一半大于$A[j]$、一半小于$A[j]$
- t_j约为$j/2$
- $T(n) = \Theta(n^2)$
  - 执行时间是n的二次函数

插入排序算法分析 (续)

■最坏情况：输入逆序

$$T(n) = \sum_{j=2..n} \Theta(j) = \Theta(n^2)$$

■平均情况：所有排列情况等可能出现

$$T(n) = \sum_{j=2..n} \Theta(j/2) = \Theta(n^2)$$

■插入排序算法快吗？

本章内容

- ■算法定义及基本概念（教材Chapter 1）
- ■算法描述（教材Chapter 2）
- ■函数增长及渐近记号表示（教材Chapter 3）
- ■标准记号与常用函数（教材Chapter 3）
- ■NP完全性理论（区分并理解P/ NP/ NPC/ NP-Hard 几类问题）（教材Chapter 34）

渐近增长

- 插入排序例子中可以看出：
  - 关注最坏情况的运行时间，是输入规模n的函数
  - 不关注常量
  - 不关注低阶项
- 宏观思想：
  - 忽略与机器相关的常量
  - 关注$n \to \infty$时$T(n)$的增长情况

渐近表示法

- 算法的渐近时间定义为一个函数，定义域为自然数集合$\mathbb{N}=\{0,1,2,...\}$(因为自变量n表示输入规模)。但有时也将其扩展到实数或限制到自然数的某子集上。

Θ记号

- **Θ记号**：给定一个函数$g(n)$，$\Theta(g(n))$表示以下函数的集合：

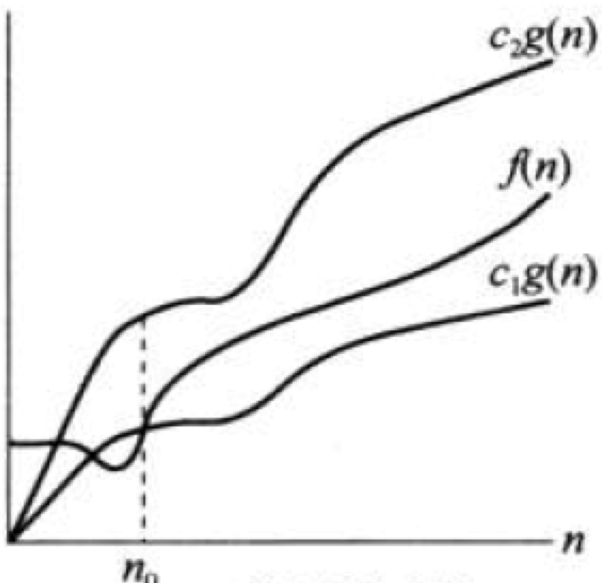
  $\boxed{\Theta(g(n)) = \{f(n)\text{: 存在正常量}c_1,c_2,n_0\text{, 使得对所有}n \geq n_0\text{有}0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}}$

  - 即$f(n) \in \Theta(g(n))$表示存在正常数c_1,c_2及足够大的n，使得$f(n)$夹在$c_1 g(n)$和$c_2 g(n)$之间
  - 通常$f(n) \in \Theta(g(n))$表示为$f(n)=\Theta(g(n))$
  - 称$g(n)$是$f(n)$的一个渐近紧确界 (asymptotically tight bound)

Θ记号 (续)

- **Θ记号**：给定一个函数$g(n)$，$\Theta(g(n))$表示以下函数的集合：

$\Theta(g(n)) = \{f(n)$: 存在正常量c_1, c_2, n_0, 使得对所有$n \geq n_0$有$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$



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

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = a, \quad a > 0$$

Θ记号 (续)

- 意义：对所有的$n \geq n_0$，函数$f(n)$在一个常数因子范围内等于$g(n)$，也就是说：$g(n)$是$f(n)$的渐近上界和渐近下界
  - $f(n)$——算法的计算时间
  - $g(n)$——算法计算时间的数量级
- 注：$\Theta(g(n))$定义中要求$f(n)$和$g(n)$是渐近非负的(n足够大时函数值非负)，否则$\Theta(g(n))$是空集

Θ记号 (续)

■例 (p27)：$T(n)=an^2+bn+c$ ($a>0$) 则$T(n)=\Theta(n^2)$

取

$$c_1 = \frac{a}{4}, c_2 = \frac{7a}{4}, n_0 = 2\max(\frac{|b|}{a}, \sqrt{\frac{|c|}{a}})$$

则对所有$n \geq n_0$, $0 \leq c_1 n^2 \leq an^2+bn+c \leq c_2 n^2$成立

■一般来说，对任意多项式

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

其中a_i为常量且$a_d>0$，我们有$p(n)=\Theta(n^d)$

Θ记号 (续)

- 记号$\Theta(1)$
  - 算法运行时间与输入规模n无关
  - 也可理解为常值函数$\Theta(n^0)$
- 渐近符号在公式中替换低阶项用来简化表达
  - 例：$4n^2+2n+1= 4n^2+2n+\Theta(1) = 4n^2+\Theta(n) = \Theta(n^2)$

O记号 (大O表示法)

■ O记号 (渐近上界)：给定一个函数$g(n)$，$O(g(n))$表示以下函数的集合：

$O(g(n)) = \{f(n)$: 存在正常量c, n_0, 使得对所有$n \geq n_0$有$0 \leq f(n) \leq cg(n)\}$

➢ 即在一个常数因子范围内$g(n)$是$f(n)$的渐近上界

➢ Θ记号强于O记号

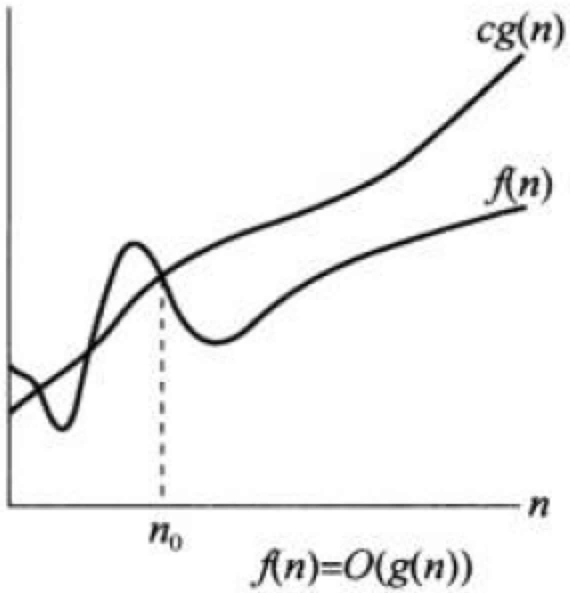
$$\Theta(g(n)) \subseteq O(g(n))$$

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

O记号 (大O表示法)

- O记号 (渐近上界)：给定一个函数$g(n)$，$O(g(n))$表示以下函数的集合：

$O(g(n))$ = {$f(n)$: 存在正常量c, n_0, 使得对所有$n \geq n_0$有$0 \leq f(n) \leq cg(n)$}



- $g(n)$是$f(n)$的一个渐近上界(asymptotically upper bound)，限制算法最坏情况运行时间

O记号 (大O表示法) (续)

- 注：
  - O记号描述上界，当用于限制算法最坏运行时间时，蕴涵着该算法在任意输入上的运行时间都限制于此界
  - Θ记号则不然，一个算法的最坏运行时间是$\Theta(g(n))$，并非蕴涵着该算法对每个输入实例的运行时间渐近紧确界均为$\Theta(g(n))$
  - 当说算法的运行时间上界是$O(g(n))$往往是指其最坏运行时间，无须修饰语，对那些最好、最坏、平均时间数量级不同者均成立，而Θ则要分开表达、加修饰语
- 例：插入排序最坏情况运行时间的界$O(n^2)$适用于该算法对每个输入的运行时间，但$\Theta(n^2)$并不适用于所有输入，当输入已排好序时为$\Theta(n)$

O记号 (大O表示法) (续)

- 例1：$T(n)=k_1n^2+k_2n+k_3=O(n^2)$, $k_1>0$
  - 由于前面的例子已经说明$T(n)=\Theta(n^2)$，因为$\Theta(n^2)$集合包含于$O(n^2)$，所以$T(n)=O(n^2)$
  - $k_1n^2+k_2n+k_3 \le (k_1+|k_2|+|k_3|)n^2$，因此当$c=k_1+|k_2|+|k_3|$且$n\geq 1$时满足$0 \le k_1n^2+k_2n+k_3 \le cn^2$
- 例2：$T(n)= k_1n^2+k_2n+k_3=O(n^3)$, $k_1>0$
  - $k_1n^2+k_2n+k_3\le(k_1+|k_2|+|k_3|)n^3$

O记号 (大O表示法) (续)

■注：

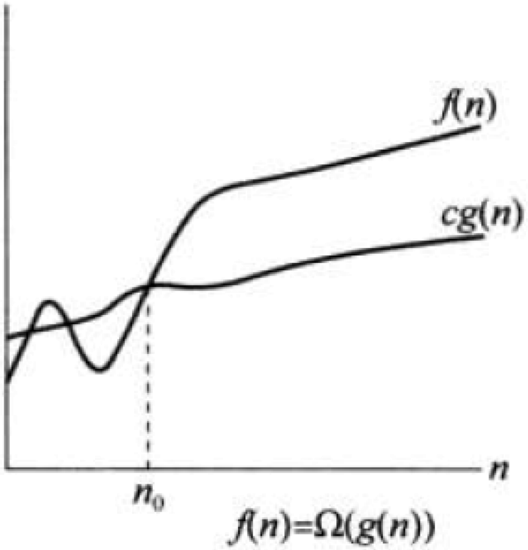
- 当我们说“运行时间是$O(n^2)$”时，通常表示最坏情况运行时间是$O(n^2)$，最好情况通常更好
- O记号使得分析算法变得更加简单
- 记号说明：
  - 通常将$f(n)\in O(g(n))$写作$f(n)=O(g(n))$
  - 公式中通常也会使用O记号，例如$2n^2+3n+1 = 2n^2+O(n)$ (也就是说$2n^2+3n+1 = 2n^2+f(n)$，其中$f(n)=O(n)$)
  - $O(1)$表示常数时间，与输入规模无关

Ω记号 (大Ω表示法)

■Ω记号 (渐近下界)：给定一个函数$g(n)$，$\Omega(g(n))$表示以下函数的集合：

$\Omega(g(n)) = \{f(n)$: 存在正常量c, n_0, 使得对所有$n \geq n_0$有$0 \leq cg(n) \leq f(n)\}$

➢即在一个常数因子范围内$g(n)$是$f(n)$的渐近下界



渐近表示法 (续)

- **定理 3.1** (p28) 对任意两个函数$f(n)$和$g(n)$，我们有$f(n)=\Theta(g(n))$，当且仅当$f(n)=O(g(n))$且$f(n)=\Omega(g(n))$。
  - 即$g(n)$是$f(n)$的渐近确界当且仅当$g(n)$是$f(n)$的渐近上界和渐近下界
- 当Ω用来界定一个算法的最好情况下的运行时间时，蕴涵着该算法在任意输入上的运行时间都限制于此界
  - 例：插入排序的下界是$\Omega(n)$，其对任何实例成立(即插入排序的最好运行时间是$\Omega(n)$)

    $n = \Omega(n), n^2 = \Omega(n)$

o记号 (小o表示法)

■o记号 (渐近非紧上界)：给定一个函数$g(n)$，$o(g(n))$表示以下函数的集合：

$o(g(n))$ = {$f(n)$: 对任意常数$c>0$, 存在常数$n_0 > 0$, 使得对所有$n \geq n_0$有$0 \leq f(n) < cg(n)$}

➤O记号表示的渐近上界可以是渐近紧致的，也可以是渐近非紧界

- 例如：$5n^2=O(n^2)$, $5n=O(n^2)$

➤o记号表示渐近非紧上界 $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

- 例如：$5n=o(n^2)$, $5n^2 \neq o(n^2)$

➤$f(n)$和$g(n)$数量级不同

ω记号 (小ω表示法)

■ω记号 (渐近非紧下界)：给定一个函数$g(n)$，$\omega(g(n))$表示以下函数的集合：

$\omega(g(n))$ = {$f(n)$: 对$\boxed{任意常数c>0}$, 存在常数$n_0 > 0$, 使得对所有$n \geq n_0$有$0 \leq cg(n) \boxed{<} f(n)$}

➢Ω记号表示的渐近下界可以是渐近紧致的，也可以是渐近非紧界

- 例如：$5n^2=\Omega(n^2)$, $5n^3=\Omega(n^2)$

➢ω记号表示渐近非紧下界 $\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty$

- 例如：$5n^3=\omega(n^2)$, $5n^2\neq\omega(n^2)$

➢$f(n)$和$g(n)$数量级不同， $f(n)$数量级更大

渐近表示法总结

- $f(n)=\Theta(g(n))$ 渐近紧确界

  $\Theta(g(n)) = \{f(n)$: 存在正常量c_1,c_2,n_0, 使得对所有$n \geq n_0$有$0 \leq c_1g(n) \leq f(n) \leq c_2g(n)\}$

- $f(n)=O(g(n))$ 渐近上界

  $O(g(n)) = \{f(n)$: 存在正常量c,n_0, 使得对所有$n \geq n_0$有$0 \leq f(n) \leq cg(n)\}$

- $f(n)=\Omega(g(n))$ 渐近下界

  $\Omega(g(n)) = \{f(n)$: 存在正常量c,n_0, 使得对所有$n \geq n_0$有$0 \leq cg(n) \leq f(n)\}$

渐近表示法总结 (续)

- $f(n)=o(g(n))$ 渐近非紧上界

$o(g(n))$ = {$f(n)$: 对任意常数$c>0$, 存在常数$n_0 > 0$, 使得对所有$n \geq n_0$有$0 \leq f(n) < cg(n)$}

- $f(n)=\omega(g(n))$ 渐近非紧下界

$\omega(g(n))$ = {$f(n)$: 对任意常数$c>0$, 存在常数$n_0 > 0$, 使得对所有$n \geq n_0$有$0 \leq cg(n) < f(n)$}

- $$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \begin{cases} 0 & f(n)=o(g(n)), f(n)=O(g(n)) \\ a, a>0 & f(n)=\Theta(g(n)), f(n)=O(g(n)), f(n)=\Omega(g(n)) \\ \infty & f(n)=\omega(g(n)), f(n)=\Omega(g(n)) \end{cases}$$

函数比较

■函数比较：实数的许多关系性质也适用于渐近比较。假定$f(n)$和$g(n)$是渐近正的：

■传递性 (5种渐近记号均适用)

$$f(n)=\Theta(g(n)) \text{ and } g(n)=\Theta(h(n)) \quad\Rightarrow\quad f(n)=\Theta(h(n))$$
$$f(n)=O(g(n)) \text{ and } g(n)=O(h(n)) \quad\Rightarrow\quad f(n)=O(h(n))$$
$$f(n)=\Omega(g(n)) \text{ and } g(n)=\Omega(h(n)) \quad\Rightarrow\quad f(n)=\Omega(h(n))$$
$$f(n)=o(g(n)) \text{ and } g(n)=o(h(n)) \quad\Rightarrow\quad f(n)=o(h(n))$$
$$f(n)=\omega(g(n)) \text{ and } g(n)=\omega(h(n)) \quad\Rightarrow\quad f(n)=\omega(h(n))$$

函数比较 (续)

- **自反性** (渐近非紧界无该性质)

$$f(n) = \Theta(f(n)), f(n) = O(f(n)), f(n) = \Omega(f(n))$$

- **对称性** (仅紧确界有该性质)

$$f(n) = \Theta(g(n)) \text{ iff } g(n) = \Theta(f(n))$$

- **转置对称性** (O与Ω、o与ω)

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \text{ iff } g(n) = \omega(f(n))$$

函数比较 (续)

- 可将两函数间的渐近比较类比于两个实数间的比较：

$$f(n) = O(g(n)) \text{ is like } a \leq b$$
$$f(n) = \Omega(g(n)) \text{ is like } a \geq b$$
$$f(n) = \Theta(g(n)) \text{ is like } a = b$$
$$f(n) = o(g(n)) \text{ is like } a < b$$
$$f(n) = \omega(g(n)) \text{ is like } a > b$$

  - 若$f(n)=o(g(n))$，称$f(n)$渐近小于$g(n)$
  - 若$f(n)=\omega(g(n))$，称$f(n)$渐近大于$g(n)$

函数比较 (续)

- 但实数的三岐性（三分性）不可类比到渐近表示中
- 三岐性：对任意两个实数a和b，下列三种情况必有一个成立：$a<b$, $a=b$, $a>b$
- 并非所有函数都是渐近可比较的，即存在$f(n)$和$g(n)$，$f(n)\neq O(g(n))$且$f(n)\neq\Omega(g(n))$
  - 例：函数n和$n^{1+\sin n}$无法渐近比较
    - $1+\sin n$在0到2之间波动

本章内容

- ■算法定义及基本概念（教材Chapter 1）
- ■算法描述（教材Chapter 2）
- ■函数增长及渐近记号表示（教材Chapter 3）
- ■标准记号与常用函数（教材Chapter 3）
- ■NP完全性理论（区分并理解P/ NP/ NPC/ NP-Hard 几类问题）（教材Chapter 34）

标准记号

■单调性

- 若$m \le n \rightarrow f(m) \le f(n)$，则函数$f(n)$是单调递增的
- 若$m \le n \rightarrow f(m) \ge f(n)$，则函数$f(n)$是单调递减的
- 若$m < n \rightarrow f(m) < f(n)$，则函数$f(n)$是严格递增的
- 若$m < n \rightarrow f(m) > f(n)$，则函数$f(n)$是严格递减的

标准记号 (续)

■向下取整与向上取整

- $\lfloor x \rfloor$：小于或等于x的最大整数 (x是任意实数)
- $\lceil x \rceil$：大于或等于x的最小整数 (x是任意实数)

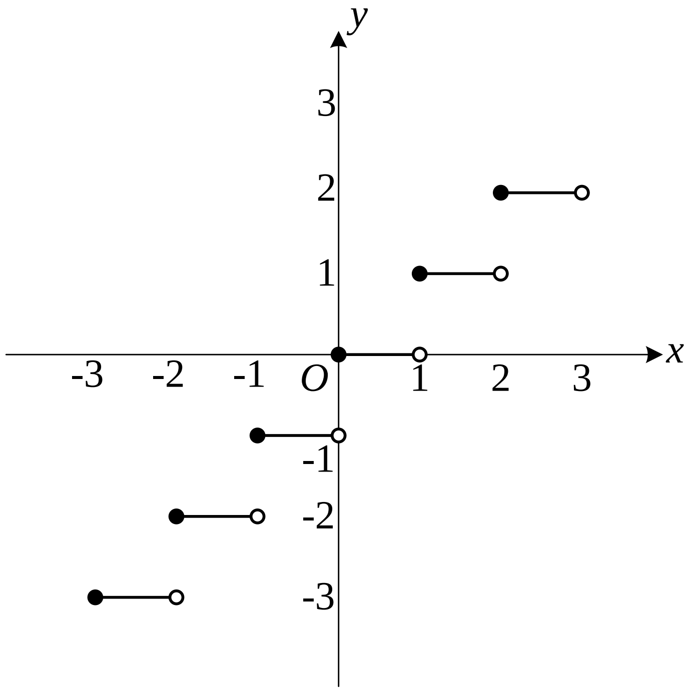
一些性质：

- $x-1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x+1, \quad \forall x \in \mathbb{R}$
- $\lceil n/2 \rceil + \lfloor n/2 \rfloor = n, \quad \forall n \in \mathbb{Z}$
- $$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil, \left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor,$$
  $$\left\lceil \frac{a}{b} \right\rceil \le \frac{a+(b-1)}{b}, \left\lfloor \frac{a}{b} \right\rfloor \ge \frac{a-(b-1)}{b}, \quad \forall x \in \mathbb{R} \wedge x \ge 0$$
  $$\forall a, b \in \mathbb{Z} \wedge a, b > 0$$

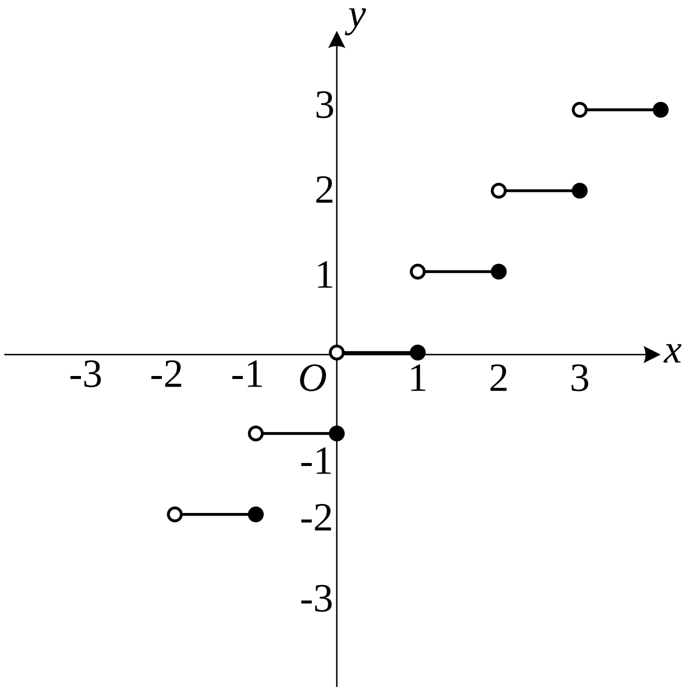
标准记号 (续)

■向下取整与向上取整

- $f(x) = \lfloor x \rfloor$ 和 $f(x) = \lceil x \rceil$ 是单调递增的



$y = \lfloor x \rfloor$



$y = \lceil x \rceil$

标准记号 (续)

■模运算

- 对任意整数a和任意正整数n，$a \bmod n$的值就是商a/n的余数：

$$a \bmod n = a - n\lfloor a/n \rfloor$$

- $0 \le (a \bmod n) < n$
- 若$(a \bmod n) = (b \bmod n)$，则记$a \equiv b \pmod{n}$，称模$n$时$a$等价于$b$ (表示余数相等)

$$a \equiv b \pmod{n} \quad \text{iff} \quad n\text{是}b\text{-}a\text{的一个因子}$$

- 若模n时a不等价于b，则记$a \not\equiv b \pmod{n}$

常用函数

■多项式

➤n的d次多项式：

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

其中，d是非负整数，常量$a_0, a_1, \dots, a_d$是多项式的系数且$a_d \neq 0$

常用函数 (续)

■多项式

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

一些性质：

- 一个多项式渐近为正，当且仅当$a_d > 0$
- 对于一个d次渐近正的多项式$p(n)$，有$p(n)=\Theta(n^d)$
- 对任意实常量$a \geq 0$，函数n^a单调递增；对任意实常量$a \leq 0$，函数n^a单调递减
- 若对某个常量k有$f(n)=O(n^k)$，则称函数$f(n)$是多项式有界的

常用函数 (续)

■指数

➢对所有正实数a、实数m和n，有

$$a^0 = 1$$

$$a^1 = a$$

$$a^{-1} = 1/a$$

$$(a^m)^n = a^{mn}$$

$$(a^m)^n = (a^n)^m$$

$$a^m a^n = a^{m+n}$$

对所有n和$a \geq 1$，函数a^n关于n单调递增

为了方便假定$0^0=1$

常用函数 (续)

■指数

一些性质：

➢对所有实常量a和b，其中$a>1$，有

$$\lim_{n\to\infty}\frac{n^b}{a^n}=0$$

因此$n^b=o(a^n)$，即任意底大于1的指数函数比任意多项式函数增长得快

常用函数 (续)

■指数

一些性质：

- 对所有实数x，有

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- 对所有实数x，有 $e^x \geq 1+x$，等号当且仅当$x=0$时成立
- 当$|x|<1$时，有 $1+x \leq e^x \leq 1+x+x^2$
- 当$x\rightarrow 0$时，$e^x = 1+x+\Theta(x^2)$
- $\lim_{n\rightarrow\infty}\left(1+\frac{x}{n}\right)^n = e^x$

常用函数 (续)

■对数

$$\lg n = \log_2 n$$

$$\ln n = \log_e n$$

$$\lg^k n = (\lg n)^k$$

$$\lg \lg n = \lg(\lg n)$$

- 对数函数仅作用于下一项，例如$\lg n+k=(\lg n)+k$，而不是$\lg(n+k)$

常用函数 (续)

■对数

一些性质：

- ➢若常量b>1，则函数$\log_b n$是严格递增的
- ➢对所有实数a>0, b>0, c>0, n，有：

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

$$\log_b(1/a) = -\log_b a$$

$$\log_c(ab) = \log_c a + \log_c b$$

$$\log_b a = \frac{1}{\log_a b}$$

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

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

$$\boxed{\log_b a = \frac{\log_c a}{\log_c b}}$$

对数的底更换时仅相差一个常量因子

其中对数的底均不为1

常用函数 (续)

■对数

一些性质：

➢当$|x|<1$时，存在一种简单的级数展开：

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \ldots = \lim_{n\to\infty}\sum_{i=1}^{n}\frac{(-1)^{i+1}}{i}\cdot x^i$$

➢若x>-1，则$\frac{x}{1+x} \le \ln(1+x) \le x$，当且仅当$x$=0时等号成立

常用函数 (续)

■对数

- ➢若对某个常量k，$f(n)=O(\lg^k n)$，则称函数$f(n)$是多对数有界的
- ➢$\lim\limits_{n\to\infty}\dfrac{n^b}{a^n}=0\ (a>1)$ 式中用$\lg n$代替n、2^a代替a，得到：

$$\lim_{n\to\infty}\frac{\lg^k n}{(2^a)^{\lg n}}=\lim_{n\to\infty}\frac{\lg^k n}{n^a}=0\ (a>0)$$

即对任意常量$a>0$，$\lg^k n=o(n^a)$

任意正的多项式函数都比任意多对数函数增长得快

常用函数 (续)

■阶乘

- $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n = \prod_{i=1}^{n} i$，并定义0!=1
- $n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$
- 斯特林(Stirling)近似公式

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

常用函数 (续)

■阶乘

➢斯特林(Stirling)近似公式

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

➢对所有$n \geq 1$，有 $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}$

其中 $\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$

➢ $n! = o(n^n) \quad n! = \omega(2^n) \quad \lg(n!) = \Theta(n \lg n)$

常用函数 (续)

■多重函数

➢使用记号$f^{(i)}(n)$表示函数$f(n)$重复i次作用于n上：

$$f^{(i)}(n) = \underbrace{f(f(f...f}_{i}(n)))$$

其中，定义$f^{(0)}(n)=n$

➢ $$f^{(i)}(n) = \begin{cases} n, & i = 0 \\ f(f^{(i-1)}(n)), & i > 0 \end{cases} \quad (i \in \mathbb{Z})$$