第一章 基本概念（二）

马嫄

2021/09/10

1.2 什么是算法

算法（Algorithm）

- 有限长的操作序列
- 接受一些输入（有些情况下不需要输入）
- 产生输出
- 一定在有穷步骤之后终止 - 有穷性
- 每一条指令必须
  - 有充分明确的目标，不可以有歧义 - 确定性
  - 每一条运算都应足够基本 - 可行性
  - 描述应不依赖于任何一种计算机语言以及具体的实现手段

算法设计原则

- 正确性：满足具体问题的需求
- 可读性：便于理解和修改
- 健壮性：当输入数据非法时，也能适当反应或进行处理
- 高效率：执行时间少
- 空间省：执行中需要的最大存储空间-存储量

时间复杂度 *T(n)*

- 衡量算法的效率，主要依据算法执行所需要的时间
  - 事后统计法：计算算法开始时间与完成时间差值

    缺点：1. 必须执行程序；

    2. 所得时间统计量依赖软硬件等环境因素，

    掩盖算法本身的优劣
  - 事前统计法：依据算法策略和问题规模***n***，是常用的方法

时间复杂度 *T(n)*

- 时间复杂度是问题规模*n*的函数

例：计算给定多项式在给定点*x*处的值

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n$$

```
double f( int n, double a[], double x )
{ int i;
  double p = a[0];
  for ( i=1; i<=n; i++ )
    p += (a[i] * pow(x, i));
  return p;
}
```

(1+2+.....+n)
=(n²+n)/2次乘法

$T(n) = C_1 n^2 + C_2 n$

时间复杂度 $T(n)$

- 在分析一般算法的效率时，我们经常关注下面两种复杂度：
  - 平均复杂度 $T_{avg}(n)$ – 算法的执行有多种可能的操作顺序
  - 最坏情况复杂度 $T_{worst}(n)$

$$T_{avg}(n) \leq T_{worst}(n)$$

时间复杂度*T(n)*的渐进表示法

- 随着问题规模n的增长，算法执行时间的增长率和$f(n)$的增长率相同，则可记作：

$$T(n) = \mathbf{O}(f(n))$$

- 渐进符号(O)的含义：存在常数$C > 0, n_0 > 0$, 使得当$n \geq n_0$时，有$T(n) \leq C \cdot f(n)$
- 一般地，时间复杂度用用算法执行基本操作的次数来度量

时间复杂度$T(n)$估算

- 算术运算和逻辑运算的时间为$O(1)$（即常数时间）

  例：+, −, ∗, /，赋值，调用等

- 语句的频度：该语句的重复执行的次数

例：a. $\{y *= x\}$ $O(1)$

b. for$(i=1; i<=n; i++)\{y *= x\}$ $O(n)$

c. for $(j=1; j<=n; j++)$

for$(i=1; i<=n; i++)\{y *= x\}$ $O(n^2)$

d. for$(i=1; i<=n; i*=2)\{y *= x\}$ $O(log_2 n)$

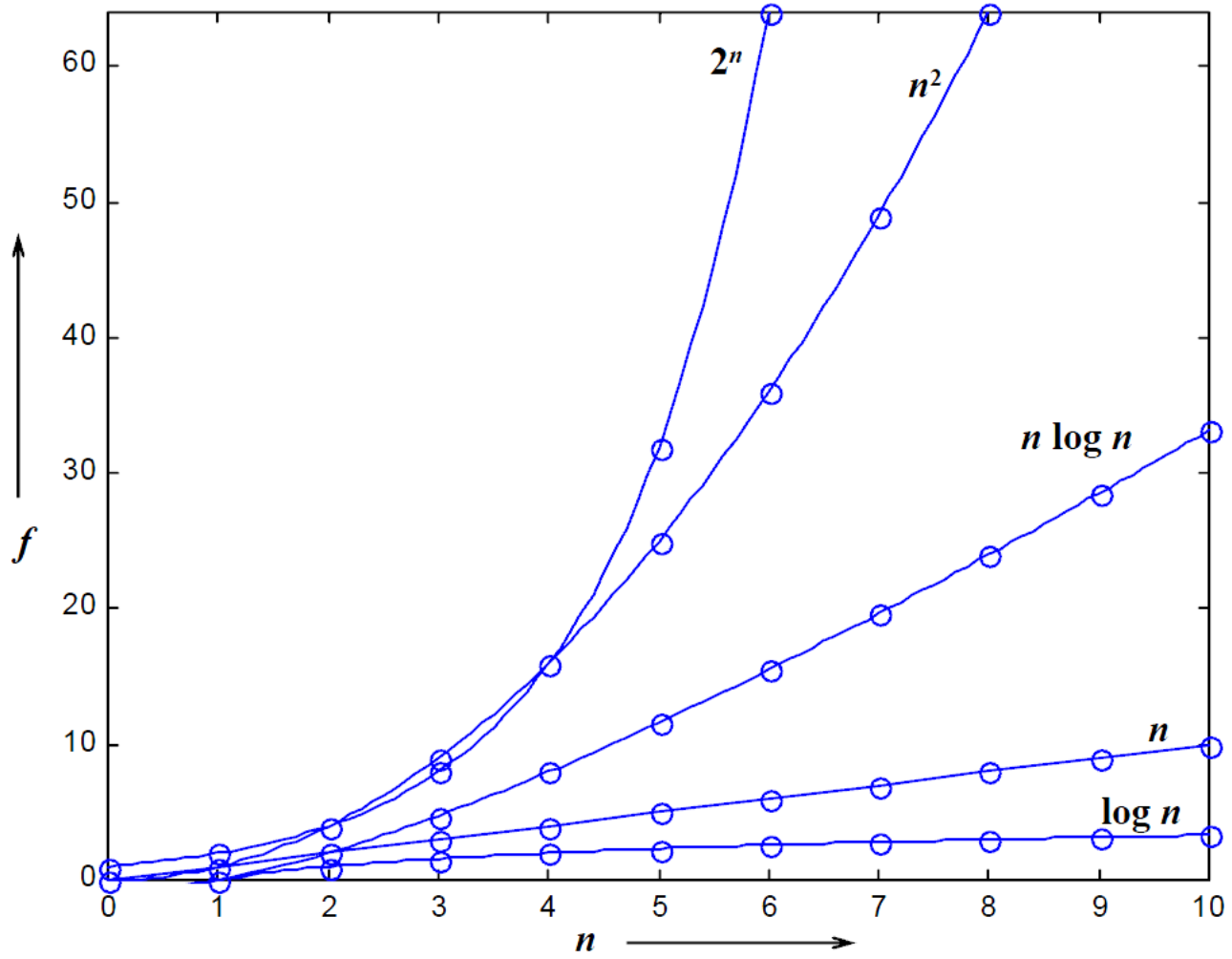
e. for$(i=n; i>0; i/=2)\{y *= x\}$ $O(log_2 n)$

时间复杂度*T(n)*分析

输入规模 n

函数	1	2	4	8	16	32
1	1	1	1	1	1	1
$\log n$	0	1	2	3	4	5
n	1	2	4	8	16	32
$n \log n$	0	2	8	24	64	160
n^2	1	4	16	64	256	1024
n^3	1	8	64	512	4096	32768
2^n	2	4	16	256	65536	4294967296
$n!$	1	2	24	40326	20922789880000	26313×10^{33}

时间复杂度*T(n)*分析



时间复杂度*T(n)*分析小窍门

- 若两段算法的复杂度分别为$T_1(n) = \mathbf{O}(f_1(n))$和$T_2(n) = \mathbf{O}(f_2(n))$，则
  - $T_1(n)+ T_1(n)=\max(\mathbf{O}(f_1(n)), \mathbf{O}(f_2(n)))$
  - $T_1(n) \times T_1(n)=\mathbf{O}(f_1(n) \times f_2(n))$
- 若$T(n)$是关于n的k阶多项式，那么$T(n) = \mathbf{O}(n^k)$
- 一个**for**循环的时间复杂度等于循环次数乘以循环体代码的复杂度
- **if-else**结构的复杂度取决于**if**的条件判断复杂度两个分枝部分的复杂度，总体复杂度取三者中最大

空间复杂度 *S(n)*

- 空间复杂度指算法执行时，所需要存储空间的量度，它也是问题规模***n***的函数，即

$$S(n) = \mathbf{O}(f(n))$$

习题

下列函数中，哪个函数具有最快的增长速度？

- A. $N(logN)^2$
- B. $(N^2)logN$
- C. $Nlog(N^2)$
- D. N^3

习题

求下列函数的渐进表达式：

- $3n^2+10n$；
- $n^2/10+2^n$；
- $\log n^3$；
- $10\log 3^n$；

习题

求下面一段代码的时间复杂度？

```
1.  if ( A > B ) {
2.      for ( i=0; i<N; i++ )
3.          for ( j=N*N; j>i; j-- )
4.              A += B;
5.  }
6.  else {
7.      for ( i=0; i<N*2; i++ )
8.          for ( j=N*2; j>i; j-- )
9.              A += B;
10. }
```