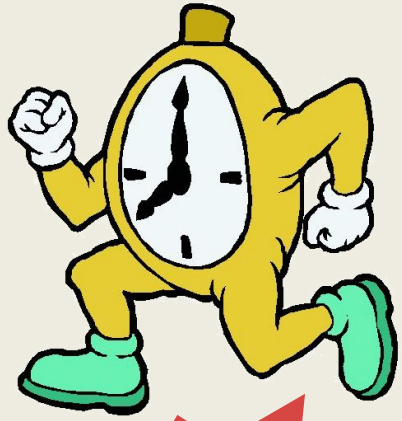
本节主题：

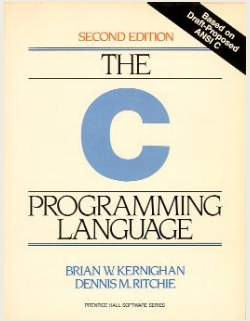
算法复杂度概念

关注效率

- 效率
  - 单位时间完成的工作量
  - 指最有效地使用社会资源以满足人类的愿望和需要
- 算法的效率：有效地使用计算资源满足需求
  - 时：用时短（CPU的计算资源）
  - 空：耗费的内存少（存储资源）
- 何为“用时短”的算法？



用什么比较？



不能用程序执行时间比较效率

- 执行的环境不同
- 实现的语言不同
- 其他因素

用基本运算的次数度量算法的时间复杂度！

- 算法的构成
  - 一个算法是由控制结构（顺序、分支和循环）和原操作（固有数据类型的操作，如比较、加减乘除等）构成的
  - 算法时间取决于两者的综合效果

控制语句1
原操作

…

控制语句n
原操作

```
void fun(int a[],int n)
{
    int i;
    for (i=0; i<n; i++)
        a[i]=2*i;
    for (i=0; i<n; i++)
        printf("%d ",a[i]);
    printf("\n");
}
```

为比较同一问题的不同算法：

- 从算法中选取一种对于所研究的问题来说是基本运算的原操作（简称为基本运算）。
- 算法执行时间大致为基本运算所需的时间与其运算次数的乘积。
- 基本运算的一般是最深层循环内的语句。

算法的时间复杂度

- 原理
  - 在一个算法中，进行基本运算的次数越少，其运行时间也就相对地越少；基本运算次数越多，，其运行时间也就相对地越多。
- 定义
  - 把算法中包含基本运算次数的多少称为算法的时间复杂度，也就是说，一个算法的时间复杂度是指该算法的基本运算次数。
- 度量
  - 算法中基本运算次数T(n)是问题规模n的某个函数f(n)，记作：T(n)=O(f(n))

```
void fun(int a[],int n)
{
   int i;
   for (i=0; i<n; i++)
      a[i]=2*i;
   for (i=0; i<n; i++)
      printf("%d ",a[i]);
   printf("\n");
}
```

乘法的次数：
T(n)=n
T(n)=O(n)

算法的时间复杂度的表示法

算法中基本运算次数T(n)是问题规模n的某个函数f(n)，记作：T(n)=O(f(n))

- “大O”
  - 表示时间复杂度的“量级”
  - 随问题规模n的增大，算法执行时间的增长率和f(n)的增长率相同。
  - 例如，$T(n)=3n^2-5n+10000=O(n^2)$
- “O”的形式定义：
  - 若f(n)是正整数n的一个函数，
  - 存在一个正常数C和n_0
  - 使得当$n \geq n_0$时
  - 有$T(n) \leq C \cdot f(n)$
  - 则记T(n)=O(f(n))

指定正常数C=3，
当n>2000时，有$3n^2-5n+10000 < 3n^2$

$T(n)=O(f(n))$的实质

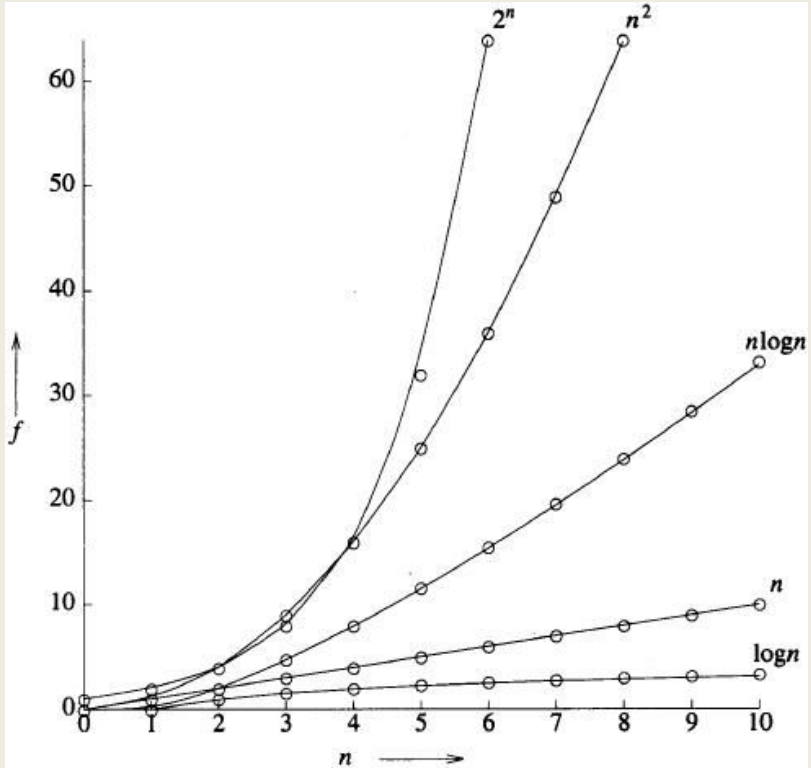
$$\lim_{n\to\infty}\frac{T(n)}{f(n)}=C$$

指导意义：

- 只求出T(n)的最高阶，忽略其低阶项和常系数，这样既可简化T(n)的计算，又能比较客观地反映出当n很大时算法的时间性能。
- 算法分析只关心n很大时的表现。

复杂度的量级

- O(1)，常数阶
  - 复杂度与问题规模n无关
  - 例：没有循环的算法
- $O(\log_2 n)$，对数阶
  - 同 O(lgn)，O(lbn)
- O(n)，线性阶
  - 与问题规模n的增长呈线性增大关系
  - 例：有执行n次的一重循环的算法
- $O(n\log_2 n)$，线性对数阶
- $O(n^2)$，平方阶
- $O(n^3)$，立方阶
- $O(2^n)$，指数阶



$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<O(2^n)<O(n!)$

运行时间随问题规模n的变化.....

n	$\lg n$	n	$n\lg n$	n^2	n^3	2^n
8	3nsec	0.01μ	0.02μ	0.06μ	0.51μ	0.26μ
16	4nsec	0.02μ	0.06μ	0.26μ	4.10μ	65.5μ
32	5nsec	0.03μ	0.16μ	1.02μ	32.7μ	4.29sec
64	6nsec	0.06μ	0.38μ	4.01μ	262μ	5.85cent
128	7nsec	0.13μ	0.90μ	16.38μ	0.002sec	10^{20}cent
256	8nsec	0.26μ	2.05μ	65.54μ	0.02sec	10^{58}cent
512	9nsec	0.51μ	4.61μ	262.14μ	0.13sec	10^{135}cent
2048	0.01μ	2.05μ	22.53μ	0.01sec	1.07sec	10^{598}cent
8192	0.01μ	8.19μ	106.50μ	0.07sec	1.15min	10^{2447}cent
32768	0.01μ	32.77μ	491.52μ	1.07sec	1.22hrs	10^{9845}cent
131072	0.02μ	131.07μ	2228.2μ	0.29min	3.3days	10^{39438}cent
524288	0.02μ	524.29μ	9961.5μ	4.58min	4.6years	10^{157808}cent

注 1：nsec 为纳秒，μ为微秒，sec 为秒，min 为分钟，hrs 为小时，days 为天，years 为年，cent 为世纪

注 2：假定每次执行一个操作所需时间的 1ns

三种复杂度

- 最佳情况复杂度
  - 可望但不要指望……
- 平均情况复杂度
  - 这个指标有价值
- 最坏情况复杂度
  - 如果不能承受极端情况的损失，这个要在意

思考题

- 为什么要进行算法的时间复杂度分析？