Ch2 算法分析

1. 计算可解性

1.1 暴力算法 Brute force

对于许多非平凡的问题，都有一种强力的搜索算法，即检验所有可能的解

- 对于大小为 n 的输入，通常需要 2^n 时间或更糟
- 在实践中是不可接受的

1.2 多项式时间算法 poly-time

我们希望，当输入的规模加倍时，算法的运行时间只增加常数倍 C

存在 $c > 0$ 且 $d > 0$，使得对于每个大小为 n 的输入，它的运行时间为 cn^d，如果算法满足以上性质，我们称它为多项式时间算法 poly-time

1.3 最坏情形运行时间

在给定大小 n 的输入条件下，求出算法最大可能运行时间的界，我们通常分析的是最坏运行时间，因为

- 它通常在实践中抓住算法的运行效率
- 很难找到有效的替代方案

1.4 有效算法的定义 efficient

1. 当实现一个算法时，如果它在真实的输入实例上运行的快，那么这个算法是有效的
2. 如果一个算法与暴力算法相比较，在最坏情况下达到质量上更好的性能，那么它是有效的
3. 如果一个算法的运行时间是多项式的，那么它是有效的

它在实践中确实有效

- 在实践中，人们开发的多时间算法几乎总是有低常数和低指数
- 突破暴力破解的指数障碍，通常暴露出问题的一些关键结构

存在例外

- 一些多时间算法确实有高常量和 / 或指数，在实践中是无用的
- 一些指数时间算法被广泛使用，因为最坏运行情况几乎没有

在常规情况下，多项式时间算法相比于指数时间算法的增长要慢得多

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

我们把效率定义的如此详细是因为它能够表达下面的概念

- 对某个特定问题不存在有效的算法

2. 增长的渐近阶

2.1 渐近阶

渐近上界 Upper bounds

如果存在常数 $c > 0$ 和 $n_0 \geq 0$ 使得对于所有 $n \geq n_0$ 我们都有$T(n) \leq c \cdot f(n)$，那么我们说 $T(n)$ 是 $O(f(n))$ 的

$O(f(n))$ 是一个函数集合，但我们经常写成 $T(n) = O(f(n))$ 而不是$T(n) \in O(f(n))$

渐近下界 Lower bounds

如果存在常数 $c > 0$ 和 $n_0 \geq 0$ 使得对于所有 $n \geq n_0$ 我们都有 $T(n) \geq c \cdot f(n)$，那么我们说 $T(n)$ 是 $\Omega(f(n))$ 的

渐近紧界 Tight bounds

如果 $T(n)$ 既是 $O(f(n))$ 又是 $\Omega(f(n))$ 的，我们说 $T(n)$ 是 $\Theta(f(n))$

直接计算渐近紧界：

当 $n \to \infty$ 时，如果函数 $f(n)$ 和 $g(n)$ 之比趋向一个正常数，那么 $f(n) = \Theta(g(n))$

命题：如果 f 和 g 是两个函数，

$$lim_{n\to\infty} \frac{f(n)}{g(n)}$$

存在，并且等于某个常数 $c > 0$，那么 $f(n) = \Theta(g(n))$

2.2 渐近增长率的性质

传递性 Transitivity

如果 $f = O(g)$ 且 $g = O(h)$ 那么 $f = O(h)$

如果 $f = \Omega(g)$ 且$g = \Omega(h)$ 那么 $f = \Omega(h)$

如果 $f = \Theta(g)$ 且 $g = \Theta(h)$ 那么 $f = \Theta(h)$

可加性 Additivity

如果 $f = O(g)$ 且 $g = O(h)$ 那么 $f + g = O(h)$

如果 $f = \Omega(g)$ 且$g = \Omega(h)$ 那么 $f + g = \Omega(h)$

如果 $f = \Theta(g)$ 且 $g = \Theta(h)$ 那么 $f + g = \Theta(h)$

2.3 常见函数的渐近界

多项式 Polynomials

$a_0 + a_1 n + ... + a_d n^d$ 的渐近界是 $\Theta(n^d)$（$a_d > 0$）

多项式时间

如果存在常数 d，对于任意输入规模 n，运行时间是 $O(n^d)$

对数

对于任意的常数 $a, b > 1$，$O(log_a n) = O(log_b n) = O(logn)$

对于任何 $x > 0$，$logn = O(n^x)$

即对数函数比任何的多项式函数都增长的慢

指数

对于任何 $r > 1$ 和 $d > 0$，$n^d = O(r^n)$

即任意的指数函数比任意的多项式函数都增长的快

3. 一般运行时间的概述

3.1 线性时间 Linear time $O(n)$

运行时间至多是输入规模的常数倍

计算最大值问题

计算 n 个数 $a_1, a_2, ..., a_n$ 的最大值

```
max ← a1
for i = 2 to n do
    if ai > max then
        max ← ai
    end if
end for
return max
```

算法的运行时间为 $O(n)$

归并问题

给定两个排好序的序列 $A = a_1, a_2, ..., a_n$ 和 $B = b_1, b_2, ..., b_n$，把它们合并为一个有序的整体

```
i ← 1, j ← 1
while(both lists are non-empty){
    if(ai ≤ bj){
        append ai to output list and increment i
    }else{
        append bj to output list and increment ij
    }
}
append remainder of nonempty list to output list
return output list
```

算法的运行时间为 $O(n)$

3.2 $O(n\ logn)$时间

经常出现在分治算法中

排序问题

归并排序和堆排序是执行 $O(n\ logn)$ 比较的排序算法

最大时间间隔问题

给定 n 个到达服务器时间为 $x_1, x_2, \ldots, x_n$ 的文本副本，最大的，没有文本副本到达的时间区间是多少？

$O(n\ logn)$ 解：对到达时间进行排序，按顺序扫描已排序列表，确定连续时间戳之间的最大间隙

3.3 二次时间 Quadratic time $O(n^2)$

枚举所有元素对

最近点问题

给定平面上 n 个点 $(x_1, y_1), \ldots, (x_n, y_n)$，找到最接近的一对点

$O(n^2)$ 解：计算所有点对，找到最小值

```
min ← (x1-x2)² + (y1-y2)²
for(i = 1 to n){
    for(j = 1 to n){
        d ← (xi-xj)² + (yi-yj)²
        if(d < min){
            min ← d
        }
    }
}
```

3.4 立方时间 Cubic time $O(n^3)$

可以枚举所有三元组

集合不相交问题

给定 n 个集合 $S_1, \ldots, S_n$，每个都是从 1 到 n 的子集，是否有集合是不相交的？

$O(n^3)$ 解：对于每一对集合，确定它们是否不相交

```
for(each set Si){
    for(each other set Sj){
        for(element p of Si){
            determine whether p also belongs to Sj
            if (no element of Si belongs to Sj){
                report that Si and Sj are disjoint
            }
        }
    }
}
```

3.5 指数时间 Exponential time $O(a^n)$

独立集问题

给定一个图，求这个图的最大独立集

独立集：图中节点的子集，其中任意两个节点没有边相连

$O(n^2 2^n)$ 解：枚举图中所有子集（2^n 个），再逐一检查点集中两点间是否有边相连（n^2 ）

3.6 亚线性时间 $O(logn)$

存在着所遇到的运行时间是小于线性时间的情况，因为它仅仅用线性时间读入输入

二分搜索法

二分搜索算法的运行时间是 $O(logn)$，一般的，只要涉及到这样的算法，$O(logn)$就会作为时间的界而出现，这样的迭代可以将问题缩短到常数规模