上节课内容回顾

Las Vegas 型随机算法

- 随机快速排序
- 随机选择
- 随机 n后放置

Monte Carlo型随机算法

- 主元素测试
- 串相等测试
- 模式匹配

素数测试

- 求 x 的 m 次幂
- 求 a 的模 n 的 m 次幂
- Fermart小定理
- 测试算法分析

求 x 的 m 次幂

输入：x为实数

$m = d_k d_{k-1} \ldots d_1 d_0$ 为二进制自然数

输出：x^m

算法 Exp(x,m)

1. $y \leftarrow 1$;
2. for $j \leftarrow k$ downto 0 do
3. $\quad y \leftarrow y^2$
4. $\quad$ if $d_j=1$ then $y \leftarrow xy$
5. return y

实例

x^{101}: $d_2=1, d_1=0, d_0=1$

$y=1$

$j=2$	$j=1$	$j=0$
$y=1$	$y=x^2$	$y=x^4$
$y=x$		$y=x^5$

a模n的m次幂

输入：$a, m, n \in Z^+$, $m \le n$,

$m = b_k b_{k-1} \ldots b_1 b_0$为二进制自然数

输出：$a^m \pmod n$

算法ExpMod(a, m,n)

1. $c \leftarrow 1$
2. for $j \leftarrow k$ downto 0 do
3. $\quad c \leftarrow c^2 \pmod n$
4. $\quad$ if b_j=1 then $c \leftarrow ac \pmod n$
5. return c

$T(n)=O(k\log^2 n)=O(\log^3 n)$　以位乘作为基本运算

Fermart小定理:测试原理

定理1： 如果 n为素数，则对所有的正整数 $a \neq 0 \pmod n$有

$$a^{n-1} \equiv 1 \pmod n$$

素数测试原理：检测 $2^{n-1} \equiv 1 \pmod n$. 如是，输出“素数”否则输出“合数”

算法 Ptest1(n)

输入：奇整数 n, $n > 5$

输出： “prime” 或者 “composite”

1. if ExpMod(2,n–1,n) =1 then return prime
2. else return composite

问题：算法 Ptest1只对a=2进行测试. 如果 n为合数且算法输出“素数”，则称 n 为基2的伪素数. 例如341.

改进算法(一)

改进算法：随机选取2--n-1中的数，进行测试. 如取a=3，3^{340}(mod 341) ≡ 56，341不是素数.

算法Ptest2(n)

1. a←Random(2, n-2)
2. if Expmod(a, n-1, n)=1 then return prime
3. else return composite

- Fermat 小定理是必要条件，不是充分条件，满足该条件的也可能是合数. 对所有与 n 互素的正整数 a 都满足条件的合数 n 称为 Carmichael 数，如 561，1105，1729，2465 等. Carmichael 数非常少，小于 10^8 的只有 255 个.
- 由初等数论可以证明：如果n为合数，但不是Carmichael数，算法Ptest2 测试 n为合数的概率至少为1/2.

素数的另一个必要条件

定理2 如果 n 为素数，则方程 $x^2 \equiv 1 (\bmod n)$ 的根只有两个，即 $x = 1$，$x = -1$（或 $x = n-1$）.

证明 $x^2 (\bmod n) \equiv 1$

$$\Leftrightarrow x^2 - 1 \equiv 0 \ (\bmod n)$$
$$\Leftrightarrow (x+1)(x-1) \equiv 0 \ (\bmod n)$$
$$\Leftrightarrow x+1 \equiv 0 \ \text{或} \ x-1 \equiv 0 \quad \text{（域中没有零因子）}$$
$$\Leftrightarrow x = n-1 \ \text{或} \ x=1$$

称 $x \neq \pm 1$ 的根为非平凡的.

判别方法： 如果方程有非平凡的根，则 n 为合数.

例如：$x^2 (\bmod 12) \equiv 1 \Leftrightarrow x = 1$ 或 $x = 11$ 或 $x = 5$ 或 $x=7$

结论：由于5 和 7 是非平凡的根，12是合数

测试方法

设 n为奇素数，存在 q, m 使得 $n-1=2^q m$,（$q\geq 1$).

构造序列：

$$a^m \pmod n,\ a^{2m} \pmod n,\ a^{4m} \pmod n, \ldots, a^{2^q m} \pmod n$$

其最后一项为 $a^{n-1} \pmod n$，而且每一项是前面一项的平方.

测试方法：

1. 对于任意 i（$i = 0,1,\ldots, q-1$），判断

$$a^{2^i m} \pmod n$$

   是否为 1 和 n-1, 且它的后一项是否为1.
2. 如果其后项为1，但本项不等于 1 和 $n-1$, 则它就是非平凡的根，从而知道 n不是素数.
3. 随机选择 $a\in\{2, 3, \ldots, n-1\}$，进行上述测试.

实例

例如 n=561, n–1=560=$2^4\cdot 35$，假设 a=7，构造的序列为

$$7^{35} \pmod{561} = 241,$$

$$7^{2^1 35} \pmod{561} = 7^{70} \pmod{561} = 298,$$

$$7^{2^2 35} \pmod{561} = 7^{140} \pmod{561} = 166,$$

$$7^{2^3 35} \pmod{561} = 7^{280} \pmod{561} = 67,$$

$$7^{2^4 35} \pmod{561} = 7^{560} \pmod{561} = 1$$

第 5 项为 1，但是第 4 项等于 67，它既不等于 1 也不等于 560，是个非平凡的根，因此可以判定 n 为合数.

根据这个思想设计的计算机算法称为 **Miller-Rabin 算法**，它随机选择正整数 $a\in\{2,3,\ldots, n-1\}$，然后进行上述测试.

算法子过程(一)

算法 findq-m(n) //找 q, m 使得 n-1=$2^q m$

1. $q \leftarrow 0$; $m \leftarrow n-1$
2. repeat
3. $\quad m \leftarrow m/2$
4. $\quad q \leftarrow q+1$
5. until m是奇数

运行时间：$O(\log n)$

算法子过程（二）

算法 test(n, q, m) //检测序列是否存在非平凡的根

1. $a \leftarrow$Random(2, n-1)
2. $x_0 \leftarrow$ExpMod(a, m, n)　　//$x_0=a^m$(mod$n$), $O(\log^3 n)$
3. for $i \leftarrow 1$ to q do　　//$q=O(\log n)$
4. 　　$x_i \leftarrow x_{i-1}^2$ (mod n)　　//$O(\log^2 n)$
5. 　　if $x_i = 1$ and $x_{i-1} \neq 1$ and $x_{i-1} \neq n-1$
6. 　　then return composite
7. if $x_q \neq 1$ then return composite
8. return prime

性能分析：

- 1 次测试运行时间 $O(\log^3 n)$
- 可证明1次测试出错的概率至多 1/2. 重复运行 k 次，可将出错概率降到至多2^{-k}.

Miller-Rabin算法

令$k=\lceil \log n \rceil$, 出错的概率小于等于$2^{-k} \le 1/n$. 即算法给出正确答案的概率为$1-1/n$. 换句话说，如果 n为素数，则算法输出素数. 如果n为合数，则算法以$1-1/n$的概率输出“合数”.

算法 PremalityTest(n) // $n \ge 5$，奇整数

1. findq-m(n)
2. $k \leftarrow \lceil \log n \rceil$
3. for $i \leftarrow 1$ to k //重复执行log n次
4. test(n, q, m)

时间：$T(n) = O(\log^4 n)$ // 按位乘统计

Las Vegas型与 Monte Carlo型随机算法

- **Las Vegas型随机算法**
  - 如果得到解，总是给出正确的结果，区别只在于运行时间的长短.
  - 拉斯维加斯型随机算法的运行时间本身是一个随机变量
  - 期望运行时间是输入规模的多项式且总是给出正确答案的随机算法称为有效的拉斯维加斯型算法.
- **Monte Carlo型随机算法**
  - 这种算法有时会给出错误的答案.
  - 其运行时间和出错概率都是随机变量，通常需要分析算法的出错概率.
  - 多项式时间内运行且出错概率不超过1/3的随机算法称为有效的蒙特卡洛型算法

单侧错误和双侧错误

- 弃真型单侧错误
  - 当算法宣布接受时，结果一定是对的
  - 当算法宣布拒绝时，结果有可能是错的
  - 例如主元素测试算法
- 取伪型单侧错误
  - 当算法宣布拒绝时，结果一定是对的
  - 而当算法宣布接受时，结果有可能是错的
  - 例如素数测试
- 双侧错误
  - 在所有的输入上同时出现上述两种不同的错误

随机算法的分类与局限性

- 拉斯维加斯型随机算法
  - 零错误概率多项式时间算法(有效的)，ZPP
- 蒙特卡洛型随机算法
  - 错误概率有界的有效算法(多项式时间)，BPP
  - 弃真型单侧错误概率有界的有效算法，RP
  - 取伪型单侧错误概率有界的有效算法，coRP
- 随机算法的局限性
  - 错误概率有界的多项式时间随机算法不太可能解决NP完全问题

第12章 处理难解问题的策略

- 对问题施加限制
  固定参数算法
- 改进指数时间算法
  3SAT、指数时间假设
- 启发式方法
  启发式方法、随机化策略、重启策略、模拟退火
- 平均情形的复杂性
  $G(n, p)$、哈密顿回路、**DistNP**完全
- 难解算例的生成

对问题施加限制

- **SAT问题**

  二元可满足性（2SAT）属于P

  HornSAT：输入限制为霍恩公式（析取式中正文字，即不带否定号的变量，至多出现一次），属于P

- **图的问题**

问题	P	NPC
VC	$D \leq 2$	$D \geq 3$
HC	2	3
顶点三着色	3	4
反馈顶点集	2	3
团	给定D	任意

固定参数算法

- 通常把优化问题转化为判定问题时，都会在输入中引入一个参数，这是固定参数算法中参数的来源之一.
- 输入中带有一个参数 k，当输入规模为 n 时运行时间为 $O(f(k)\, n^c)$ 的算法，这里的 $f(k)$ 是与 n 无关的函数，c 是与 n 和 k 都无关的常数.
- 例

  VC：给定图 G，正整数 K(不超过 G 的顶点数)，问是否存在不超过 K 的顶点覆盖？

  固定常数 k，输入为(G, k). 穷举所有 k元 顶点子集，看看是否存在顶点覆盖. 算法复杂度大约是

$$O(knC_n^k) = O(kn^{k+1})$$

  存在$O(2^k kn)$的算法.

改进的指数时间算法

- O^*：表示忽略了多项式因子，如$O^*(2^n)=O(n^{O(1)}2^n)$
- 当一个问题的蛮力算法为$O^*(2^n)$时间时，对任何满足$1 < c < 2$的常数c，时间复杂度为$O^*(c^n)$的指数时间算法称为非平凡的指数时间算法，或改进的指数时间算法.
- 可证明在$O^*(1.8393^n)$时间内正确求解3SAT，截止到2010年底 最好结果：$O^*(1.321^n)$时间的随机算法和$O^*(1.439^n)$时间的确定型算法.
- 指数时间假设：对每个正整数 k，都存在常数 $c_k>0$，使得求解 kSAT 的精确算法时间复杂性不低于 $2^{c_k n}$.
- 任意色数的图的顶点着色问题都有$O^*(3^n)$的算法. 背包问题有比$O^*(2^{n/2})$更好的算法. 货郎问题也有比$O^*(2^n)$更好的算法.

其他处理难解问题的策略

- 启发式方法（**Heuristics**）：目前无法从理论上给出任何性能保证，但在实践中效果良好，就把这类方法统称为启发式方法（**Heuristics**）.

  常用的启发式方法主要包括：回溯与分支限界法、局部搜索法（随机化策略、重启策略、模拟退火）、遗传算法等.

- 平均情况下的复杂度

  有些NP完全问题在平均复杂性度量下是易解的，哈密顿回路问题的平均情况下对图$G(n,1/2)$有$O(n^3)$时间的算法.

- 难解算例生成：确定紧的实例
- 基于统计物理的消息传递算法

总结

- 素数检验
- 随机算法小结
- 处理难解问题的策略