算法设计与分析

第15讲 随机算法

汪小林

北京大学 信息科学技术学院

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，**ZPP**
- 蒙特卡洛型随机算法
  - 单侧错误概率，**RP**，**coRP**
  - 双侧错误概率，**BPP**
- 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

9.3.1拉斯维加斯型随机算法

- 特征
  - 这种算法的结果总是正确的，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
- 例子：快速排序算法总是给出已经排序的数组，期望的时间则是$2n \ln n$.
- 把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. 快速排序算法是有效的拉斯维加斯型算法

9.3.2 蒙特卡洛型随机算法

- 特征
  - 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
- 总是在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
  - 只要让算法独立重复执行足够多次，出错概率可以降到任意小
- 例子：素数检验、多项式恒等检验的随机算法和布尔可满足性问题的随机游动算法都是这类算法

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，后面将要介绍的随机游动算法
- 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，后面的素数检验和多项式恒等检验
- 双侧错误
  - 所有输入上同时出现上述两种不同的错误

9.3.3有效随机算法的复杂性类

- 有效的拉斯维加斯型算法被称为**ZPP**类
  - 即零错误概率多项式时间随机算法
- 有效的蒙特卡洛型算法被称为**BPP**类
  - 即错误概率有界的多项式时间随机算法
  - 有效的弃真型单侧错误随机算法称为**RP**类
  - 有效的取伪型单侧错误随机算法称为**coRP**类
- 如果把**ZPP**类算法可以求解的判定问题类记作**ZPP**类，类似可以定义**BPP**、**RP**、**co-RP**等判定问题类，则

$$\mathbf{P \subseteq ZPP = RP \cap coRP \subseteq BPP}$$

有效随机算法的局限性

- 卡普-利普顿定理：

  若$NP \subseteq P/poly$，则$PH = \sum_2^P$

  - P/poly表示多项式规模电路能够求解的问题类
  - PH表示复杂性类从低到高的层次结构，也叫做多项式谱系，$\sum_2^P$ 是其中的第2层
- 不太可能为NP完全问题设计出多项式时间的随机算法
  - 已知 $BPP \subseteq P/poly$
  - 如果$NP \subseteq BPP$，则PH“塌方”到第2层
  - 研究计算复杂性的人一般认为PH是不塌方的

最大公因数与二次剩余

- 我们把a与b的最大公因数记为**gcd(a,b)**
  - 可用著名的辗转相除法求得
- 若存在b，使得$a=b^2 \pmod n$且$\gcd(b,n)=1$，则说a是模n的二次剩余
- 对于$1 \le a \le n$，定义

$$\mathrm{QR}_n(a)=\begin{cases}0, & \gcd(a,n)\neq 1\\ +1, & a\text{是模}n\text{的二次剩余}\\ -1, & \text{否则}\end{cases}$$

雅各比符号

- 对于奇数n和a，设n的全部素数因子为$p_1,\ldots,p_k$，即 $n=\prod_{i=1}^{k} p_i^{\alpha_i}$，定义雅各比符号为

$$\left(\frac{a}{n}\right)=\prod_{i=1}^{k}\mathrm{QR}_{p_i}(a)$$

  - 雅各比符号可在$O(\log a \cdot \log n)$时间内求得

- 对于所有满足$\gcd(n,a)=1$的$1\le a\le n-1$，如果n为奇素数，下式总成立；n为合数则至多有一半的a满足

$$\left(\frac{a}{n}\right)=a^{(n-1)/2}(\mathrm{mod}\, n)$$

素数检验算法

算法 9.2

输入：自然数n

输出：n是否素数

1. 从$\{1,\ldots,n-1\}$中随机选择自然数a

2. 若 $\gcd(n,a)>1$ 或 $\left(\frac{a}{n}\right) \neq a^{(n-1)/2} \pmod{n}$

则宣布 “n是合数”

3. 否则宣布 “n是素数”

定理9.2

- 定理9.2 素数检验算法在多项式时间内运行. 当n是素数时，素数检验算法总是输出正确结果；当n不是素数时，素数检验算法至少以概率1/2输出正确结果.

- 证明 由于最大公因数和雅各比符号都是多项式时间可计算的，而利用反复平方法，求幂也是多项式时间可计算的，显然素数检验算法在多项式时间内运行.

定理9.2证明(续)

如前所述，由初等数论中的结论可知，对于每个奇素数 n 和所有$1 \le a \le n-1$，都有

$$QR_n(a)=a^{(n-1)/2}(\mathrm{mod}\ n);$$

当n是素数时，算法总是正确检验出“n是素数”；

对每个合数n 和所有满足 $\gcd(n,a)=1$的a, $1 \le a \le n-1$,

至多有一半的a满足

$$\left(\frac{a}{n}\right) = a^{(n-1)/2}(\mathrm{mod}\ n)$$

算法至少以1/2概率检验出“N是合数”.

说明：这是个coRP算法

算法9.3 多项式恒等检验算法

输入：用规模为m的代数电路$C(x_1,\ldots,x_n)$表示的多项式$r(x_1,\ldots,x_n)$

输出：$r(x_1,\ldots,x_n)$是否恒等于0

1. 从$\{1,\ldots,10\cdot 2^m\}$中随机选择n个自然数$a_1,\ldots,a_n$ (可重复)；
2. 从$\{1,\ldots,2^{2m}\}$中随机选择自然数k；
3. 在模k算术下，计算

$$y = C(a_1,\ldots,a_n) \pmod{k} = r(a_1,\ldots,a_n) \pmod{k};$$

4. 若y=0，则输出“$r(x_1,\ldots,x_n)$恒等于0”；
5. 否则，输出“$r(x_1,\ldots,x_n)$不恒等于0”.

定理9. 3

定理9. 3 算法9.3在多项式时间内运行. 当$r(x_1,\ldots,x_n)$恒等于0时，算法输出正确结果；当$r(x_1,\ldots,x_n)$不恒等于0时，算法至少以$9/(40m)$概率输出正确结果.

证明 每个a_i的长度为$O(m)$位，$a_1,\ldots,a_n$的总长度为$O(n\cdot m)$位，k的长度为$O(m)$位，所有运算的中间结果的长度也都为$O(m)$位. 同余算术中加法、减法、乘法都是多项式时间的，所以上述算法在多项式时间内运行.

定理9. 3证明(续)

- 当 $r(x_1,\ldots,x_n)$ 恒等于0时，无论什么值代入后都等于零，所以算法总是输出正确结果.
- 当 $r(x_1,\ldots,x_n)$ 不恒等于0时，

$$\Pr[\, r(a_1,\ldots,a_n) \neq 0 \,] \geq 9/10.$$

若电路C的规模为m，当这m个门都是乘法时，则 r的次数 $\deg(r)$ 最多可到2^m，根据施华兹-齐普尔引理，设a_i都从$S = \{1,\ldots,10*2^m\}$中随机取值,

$$\Pr[\, r(a_1,\ldots,a_n) = 0 \,] \leq \deg(r)/|S|$$

$$\leq 2^m/(10\cdot 2^m) = 1/10.$$

定理9. 3证明(续完)

- 当 $y = r(a_1,\ldots,a_n) \neq 0$时， $\Pr[\, y \neq 0 \pmod{k}\,] \geq 1/(4m)$.

  由素数定理，$\{1,\ldots,2^{2m}\}$中素数至少有$2^{2m}/(2m)$个.

  y的素数因子至多有$\log y \leq 5m2^m = o(2^{2m}/(2m))$个.

  当$m \geq 11$时， $2^{2m}/(2m) - 5m2^m \geq 1/(4m)$

  $\{1,\ldots,2^{2m}\}$中的素数至少有$2^{2m}/(4m)$个都不是y的素因子，当k取这些值时，$y \neq 0 \pmod{k}$. 所以

$$\Pr[\, y \neq 0 \pmod{k}\,] \geq (2^{2m}/(4m)) / 2^{2m} = 1/(4m).$$

算法至少以$(9/10)\cdot(1/4m) = 9/(40m)$的概率正确输出$r$不恒等于零.

- 独立重复执行$O(m)$次，把出错概率降到1/3以下.

离散随机过程

- 一个离散随机过程就是一组随机变量

$$X = \{X_t \mid t \in T,\ T\text{是可数集}\}$$

通常 t 代表时间，X_t 就是 X 在时刻 t 的状态。

- 例如，赌徒每次抛一枚均匀硬币来赌博
  - 若正面向上，赢1元钱；反面向上，就输1元钱
  - 假设初始赌本为X_0，在时刻 t 的赌本为X_t，

    则 $\{X_t \mid t=0,1,2,\ldots\}$就是一个离散随机过程。

有限马氏链

- 一个有限马氏链是满足下列条件的离散随机过程 $\{X_0, X_1, X_2, \ldots\}$，其中每个$X_t$都从一个有限集中取值

$$\Pr[X_t=a|X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$
$$= \Pr[X_t=a|X_{t-1}=b] = p_{b,a}$$

即X_t的值依赖于X_{t-1}的值，而不依赖之前的历史.

- 在上述赌徒的例子中，当 $b \neq 0$ 或 n 时，

$$\Pr[X_t=b+1 \mid X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$
$$= \Pr[X_t= b+1|X_{t-1}=b] = 1/2$$
$$\Pr[X_t=b-1 \mid X_{t-1}=b, X_{t-2}=a_{t-2}, \ldots, X_0=a_0]$$
$$= \Pr[X_t= b-1|X_{t-1}=b] = 1/2$$

即 $$p_{b,b+1} = p_{b,b-1} = 1/2$$

一步转移矩阵

- 对于有限马氏链，不妨设X_t取值的状态空间为$\{0,1,2,\ldots,n\}$. 于是$p_{i,j}$可组成一个$n+1$阶方阵

$$P = [p_{i,j}]_{(n+1)\times(n+1)},$$

叫做一步转移矩阵. 其每一行元素之和等于1，即对任意 i，有$\Sigma_j\, p_{i,j} = 1$.

*m*步转移矩阵

- 设 $p_i(t)$ 表示在时刻 t 处在状态 i 的概率，则

$$p_i(t) = p_0(t\text{-}1)p_{0,i} + p_1(t\text{-}1)p_{1,i} + \ldots + p_{n\text{-}1}(t\text{-}1)p_{n\text{-}1,i} + p_n(t\text{-}1)p_{n,i}$$

- 设$p(t)$表示向量$(p_0(t), p_1(t), \ldots, p_{n\text{-}1}(t), p_n(t))$，则

$$p(t) = p(t\text{-}1)\, P.$$

- 对任意m，定义*m*步转移矩阵，

$$P^{(m)} = [p_{i,j}^{(m)}]_{(n+1)\times(n+1)}$$

其中

$$p_{i,j}^{(m)} = \Pr[X_{t+m} = j \mid X_t = i]$$

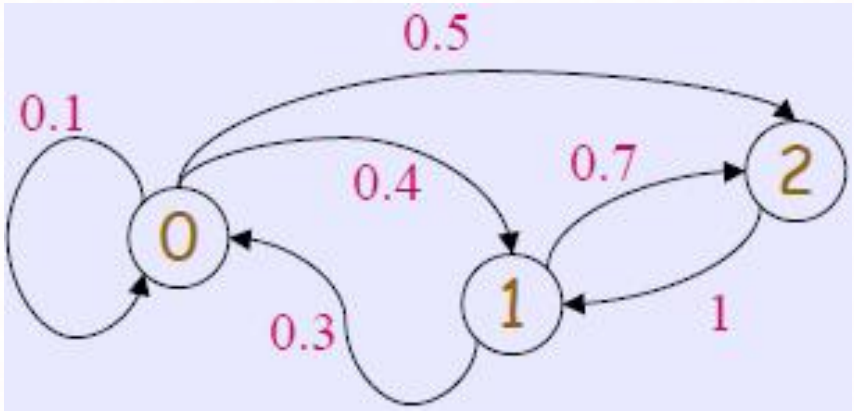
- 于是

$$P^{(m)} = P^m$$

$$p(t+m) = p(t)P^m$$

有限马氏链的图示

- 有限马氏链还可以用有向带权图来表示
  - 每个状态作为一个顶点
  - 两个状态 i, j之间的有向边所带的权就是这两个状态之间的转移概率 $p_{i,j}$
- 如图所示就是一个三状态马氏链.



算法9.4 2SAT的随机游动算法

输入：一个有n个变元和m个子句的合取范式，每个子句至多有两个文字

输出：一个满足赋值，或宣布没有满足赋值

1. 任意给所有变量赋值；
2. 若当前赋值是满足赋值，则输出这个赋值，算法结束；
3. 均匀随机选一个不满足子句，从中均匀随机选一个文字，改变该文字的赋值；
4. 重复第2-3步$2mn^2$次，若一直没有找到满足赋值，则宣布没有满足赋值.

定理9.4

- 定理9.4 若输入公式是不可满足的，则上述随机游动算法正确宣布其不可满足. 若输入公式是可满足的，则上述算法以$1-1/2^m$概率找到满足赋值.
- 证明　显然，当输入公式是不可满足的，算法无法找到满足赋值，因此将宣布不可满足. 当输入公式可满足时，算法有可能找不到满足赋值而出错. 我们证明算法出错的概率为$1/2^m$.

定理9.4证明(续)

- 假设输入公式可满足，
  - 设a^*是某个可满足赋值，
  - 设a_t是算法在执行 t 遍第3步以后的赋值，
  - 设 X_t 等于a^*和 a_t 赋值相同的变量个数.
- 令变量数为n，
  - 则当 $X_t=n$ 时$a_t=a^*$，算法将找到满足赋值a^*而停止.
  - 注意在 $X_t=n$之前，算法也可能找到别的满足赋值而停止.

定理9.4证明(续一)

考虑 X_t 和 X_{t+1} 的关系

当 $X_t=0$ 时，显然有 $X_{t+1}=1$ ，所以

$$\Pr[X_{t+1}=1|X_t=0]=1.$$

当 $X_t=j$ 且 $0<j\le n-1$ 时，$X_{t+1}=j+1$ 或 $X_{t+1}=j-1$. 假设算法选择修改子句 $C=a\vee b$ 的一个变量的赋值，则 C 是不满足子句，所以在赋值 a_t 之下 $a=b=$假. 在 a^* 下，$a=$真且 $b=$假，或 $a=$假且 $b=$真，或 $a=b=$真. 因此随机选择改变 a 或 b 的赋值时，至少有一半机会让 a^* 和 a_t 赋值相同的变量个数增加1个. 所以

$$\Pr[\,X_{t+1}=j+1|X_t=j\,]\ge 1/2,\quad \Pr[\,X_{t+1}=j-1|X_t=j\,]\le 1/2.$$

- 注意 $\{X_0, X_1, X_2,\ldots\}$ 可能不是马氏链.

定理9.4证明(续二)

定义真正的马氏链$\{Y_0, Y_1, Y_2,\ldots\}$如下：

$Y_0=X_0$

$\Pr[Y_{t+1}=1|Y_t=0]=1$

$\Pr[Y_{t+1}=j+1|Y_t=j]=1/2$

$\Pr[Y_{t+1}=j-1|Y_t=j]=1/2$

注意到

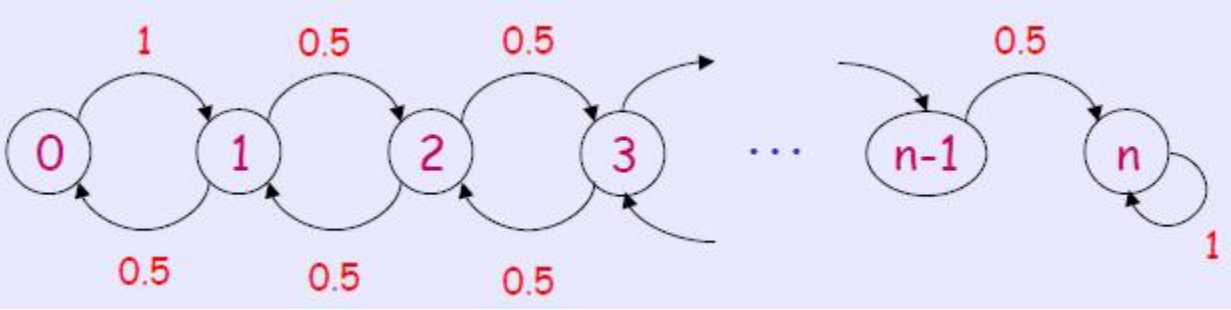
$\Pr[\ Y_{t+1}=j+1|Y_t=j\ ] = 1/2 \le \Pr[X_{t+1}=j+1|X_t=j]$

$\Pr[\ Y_{t+1}=j-1|Y_t=j\ ] = 1/2 \ge \Pr[\ X_{t+1}=j-1|X_t=j\ ]$

从任意状态出发，Y_t将比 X_t花费更长的期望时间才能进入状态n.

定理9.4证明(续三)

- 下面我们分析Y_t进入状态n所需要的期望时间，以此作为算法期望运行时间的上界.
- 注意马氏链$Y=\{Y_0, Y_1, Y_2,\ldots\}$可以用下图所示的带权有向图表示.



- 设h_j表示从状态j到达状态n的期望运行时间
  - $h_n=0$和$h_0=h_1+1$.
  - 对于其他 j 值

$$h_j=\frac{1}{2}(h_{j+1}+1)+\frac{1}{2}(h_{j-1}+1)$$

定理9.4证明(续完)

用归纳法易证：对于所有j，

$$h_j \le n^2-j^2 \le n^2$$

$$\mathrm{E}[X\text{从}X_0\text{到达}n\text{的时间}]$$

$$\le \mathrm{E}[Y\text{从}Y_0\text{到达}n\text{的时间}] \le n^2$$

由马尔科夫不等式，就有

$$\Pr[X\text{从}X_0\text{到达}n\text{的时间}>2n^2] \le 1/2$$

即若以$2n^2$步为一个阶段，则在一个阶段中算法找不到满足赋值的出错概率不超过1/2. 当重复执行每个阶段m次，即总共执行$2mn^2$步时，算法仍然找不到满足赋值的出错概率小于$1/2^m$.