第11章 随机算法

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

随机算法的分类与局限性

随机快速排序算法

算法 随机快速排序算法

输入：包含 n 个元素的数组

输出：经过排序的 n 个元素的数组

1. 若数组包含 0 或 1 个元素则返回
2. 从数组中随机选择一个元素作为**枢轴**元素
3. 把数组元素分为三个子数组，并且按照 A, B, C 顺序排列

   A：包含比枢轴元素小的元素；

   B：包含与枢轴元素相等的元素；

   C：包含比枢轴元素大的元素.
4. 对 A 和 C 递归地执行上述步骤.

算法分析

定理 设数组含n个不同元素，随机快速排序算法的期望比较次数

$$T(n) \le 2\, n \ln n.$$

证明一：求解递推式

随机选取枢轴元素，其位于排序后第 i 位置 (i=0,1,…,n-1)的概率是1/n，A和C的元素数分别是 i 个和 n–i–1个, 得

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}[T(i) + T(n-i-1)] = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1}T(i)$$

解为Θ(nlogn). 可归纳证明精确上界是 $T(n) \le 2n \ln n$.

$$T(n) \le (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} 2i\ln i \le (n-1) + \frac{2}{n}\int_1^n 2x\ln x\,dx$$

$$\le (n-1) + \frac{2}{n}(n^2\ln n - \frac{n^2}{2} + \frac{1}{2}) \le 2n\ln n$$

北京大学

随机选择算法

算法 RandSelect(A, p, r, k)　//从$A[p..r]$中选第k小

1. if $p=r$ then return $A[p]$
2. $i \leftarrow$Random(p, r)
3. 以 $A[i]$ 为标准划分 A
4. $j \leftarrow$划分后小于等于 $A[i]$ 的数构成数组的大小
5. if $k \le j$
6. 　　then return RandSelect ($A, p, p+j-1, k$)
7. 　　else return RandSelect ($A, p+j, r, k-j$)

北京大學

时间期望值估计

假设1.. n 中每个数被选的概率相等，并且假设第 k 个数总是出现在划分后两个数组中较大的数组，算法的期望时间为

$$T(n) \le \frac{1}{n}(T(n-1)+T(n-2)+\ldots+T(\frac{n}{2}+1)$$

$$+T(\frac{n}{2})+T(\frac{n}{2}+1)+\ldots+T(n-1))+O(n) \le \frac{2}{n}\sum_{i=n/2}^{n-1}T(i)+O(n)$$

归纳证明 $T(n) \le cn$. 归纳步骤如下：假设对 $k<n$ 命题为真，

$$T(n) \le \frac{2}{n}[c\frac{n}{2}+c(\frac{n}{2}+1)+\ldots+c(n-1)]+tn = \frac{2c}{n}\frac{(\frac{n}{2}+n-1)\frac{n}{2}}{2}+tn$$

$$= \frac{3cn}{4}-\frac{c}{2}+tn \le \frac{3cn}{4}+tn = (\frac{3}{4}+\frac{t}{c})cn \le cn，\quad \text{取} c \ge 4t \text{ 即可}$$

n后放置的随机选择

算法BoolQueen(n)

1. $k \leftarrow 1$ // k 放皇后的行号
2. $count \leftarrow 0$ // $count$ 放好的皇后数
3. while $k \le n$ do
4. for $i \leftarrow 1$ to n do // i 为待选列号
5. 检查 i 与前面 k-1个皇后的相容性
6. 如果相容则将 i 加入S
7. if $S \ne \varnothing$ then
8. $j \leftarrow \text{Random}(1,|S|)$
9. $x_k \leftarrow S[j]$
10. $count \leftarrow count+1$
11. $k \leftarrow k+1$
12. else $k \leftarrow n+1$
13. return $count$

n后问题的随机算法

算法QueenLV(n) //重复调用随机算法BoolQueen

1. $p \leftarrow$BoolQueen(n)
2. while $p<n$ do
3. $p \leftarrow$BoolQueen(n)

改进算法---与回溯相结合

设 $stopVegas \leq n$, 表示用QueenLV算法放置的皇后数

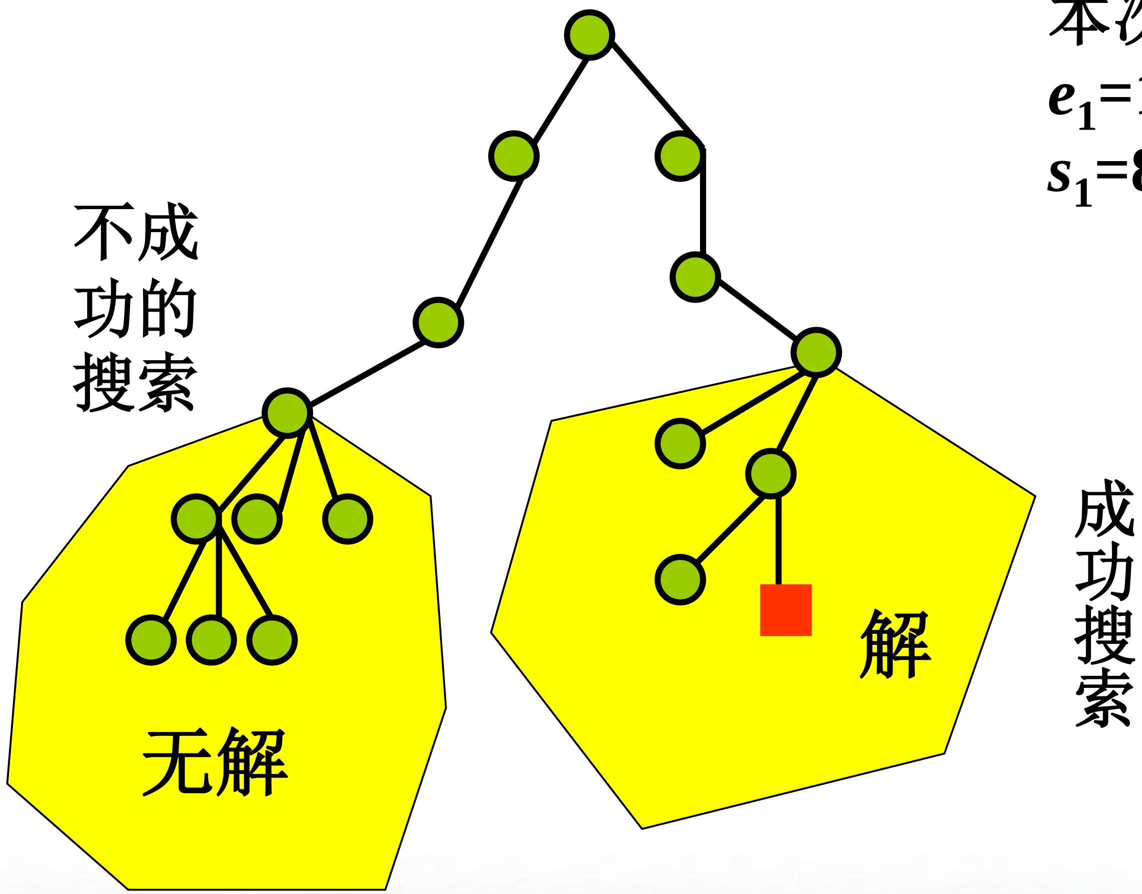
剩下 n - $stopVegas$ 个皇后用回溯方法放置

$stopVegas = 0$ 时是完全的回溯算法

$stopVegas = n$ 时是完全的Las Vegas算法

成功搜索与不成功搜索

本次抽样：
e_1=10
s_1=8



改进算法的分析

对于不同的 *stopVegas* 值，设

- p 为算法成功概率
- s 为一次成功搜索访问的结点数的平均值
- e 为一次不成功搜索访问的结点数的平均值
- t 为算法找到一个解的平均时间

$$t = ps + (1-p)(e+t) \Rightarrow t = s + e\frac{1-p}{p}$$

n=12时的统计数据：*stopVegas* = 5时算法效率高

stopVegas	*p*	*s*	*e*	*t*
0	1.0000	262.00	-	262.00
5	0.5039	33.88	47.23	80.39
12	0.0465	13.00	10.20	222.11

LasVegas型随机算法总结

- 特点

  通过修改确定性算法得到，一般将算法的某步的确定型选择变成随机选择

  一次运行可能得不到解；若得到解，则解一定是正确的

  改进途径：与确定型算法相结合

  有可能改进确定型算法平均情况下的时间复杂度

- 有效的 LasVegas 算法

  运行时间是随机变量，期望运行时间是输入的多项式且总能给出正确答案的随机算法

主元素测试

问题

主元素：出现次数超过一半以上的元素

输入：n 个元素的数组 T

输出：如果存在主元素则输出“true”，否则“false”

算法 Majority(T,n)

1. $i \leftarrow$ Random(1,n)
2. $x \leftarrow T(i)$
3. 计数 x 在 T 中出现的个数 k
4. if $k>n/2$ then return true
5. else return false

北京大学

算法的正确性

若回答true：则T 存在主元素，算法正确；若回答false，T 仍可能存在主元素，算法可能出错. 回答正确概率大于1/2.

算法 BoolMajority(T,n)

1. if Majority(T,n) then return true
2. else return Majority(T,n)

BoolMajority 算法正确的概率为

$$p+(1-p)p=2p-p^2=1-(1-p)^2>\frac{3}{4}$$

调用 k 次Majority算法正确的概率为

$$p+(1-p)p+(1-p)^2p+\ldots+(1-p)^{k-1}p=1-(1-p)^k>1-2^{-k}$$

调用次数 k	1	2	3	4	5	6
正确概率大于	0.5	0.75	0.875	0.938	0.969	0.985

改进途径

对于任意给定的 ε>0, 如果要使出错的概率不超过 ε，则调用次数 k 满足

$$\left(\tfrac{1}{2}\right)^k \le \varepsilon \Rightarrow k\log\tfrac{1}{2} \le \log\varepsilon \Rightarrow -k \le \log\varepsilon$$

$$\Rightarrow k \ge -\log\varepsilon \Rightarrow k \ge \left\lceil \log\tfrac{1}{\varepsilon} \right\rceil$$

出错概率不超过ε 的算法——MCMajority

算法 MCMajority(T,n,ε)

1. $k \leftarrow \lceil \log(1/\varepsilon) \rceil$
2. for $i \leftarrow 1$ to k
3.   if Majority(T,n) then return true
4. return false

北京大学

串相等测试

问题：A 有一个长串 x, B 有长串 y，A 和 B 希望知道 $x=y$?

方法一：

A 将 x 发送给 B，B 测试 $x = y$?

发送消耗：长串占用信道资源大

方法二：

A 用 x 导出一个短串 $f(x)$ (fingerprints)

A 将 $f(x)$ 发送到 B

B 使用同样方法导出相对于 y 的短串 $f(y)$

B 比较 $f(x)$ 与 $f(y)$

如果 $f(x) \neq f(y)$，则 $x \neq y$;

如果 $f(x) = f(y)$，则不确定.

指纹产生

设 x 和 y 的二进制表示为正整数 $I(x), I(y)$

选择素数 p, 指纹函数为

$$I_p(x) = I(x) \bmod p$$

A 传送 p 和 $I_p(x)$ 给 B. 当 p 不太大时，传送一个短串.

存在问题:

$$x = y \Rightarrow I_p(x) = I_p(y)$$

$$I_p(x) = I_p(y) \not\Rightarrow x = y$$

出错条件：固定

$$p \mid (I(x) - I(y))$$

改进算法的途径

改进方法： 随机选择素数 p 进行测试

算法 StringEqualityTest

1. 随机选择小于 M 的素数 p //M为正整数
2. A 发送 p 和 $I_p(x)$ 给 B
3. B 测试是否 $I_p(x) = I_p(y)$

出错的必要条件：

x 的位数等于 y 的位数

$p \mid (I(x)-I(y))$

有关素数的性质

函数 $\pi(t)$：小于 t 的不同的素数个数，

例如，$\pi(20)=8$，素数8个：2, 3, 5, 7, 11, 13, 17, 19

两个相关结果：

(1) 素数定理 $\pi(t) \approx t/\ln t$

(2) 若 $k<2^n$，n不太小，整除 k 的不同素数个数小于 $\pi(n)$

n	10^3	10^4	10^5	10^6	10^7
$\pi(n)$	168	1229	9592	78498	664579
$t/\ln t$	145	1086	8686	72382	620421
比值	1.159	1.132	1.104	1.085	1.071

$k < 2^{10} = 1024$，整除 k 的素数个数 $< \pi(10) = 4$，

例如 $k = 984$，整除 984 的素数只有 2, 3, 41

出错概率估计

n: x 和 y 的二进制表示的位数

$x, y \le 2^n$

若选择 $M \ge 2n^2$, 一次测试的出错概率估计:

$$\frac{|\{p \mid p\text{是小于}2^n\text{的素数，且 }p\text{ 整除 }I(x)-I(y)\}|}{\pi(M)}$$

$$\le \frac{\pi(n)}{\pi(M)} \approx \frac{n/\ln n}{2n^2/\ln(2n^2)} \approx \frac{n/\ln n}{2n^2/2\ln n} \le \frac{1}{n}$$

改进算法

重复执行 j 次，每次随机选择小于 M 的素数

算法 StringTest

输入：x, y，n位二进制数

输出： "Yes" (如果 $x = y$)； 或者 "No" (如果 $x \neq y$)

1. for $i \leftarrow 1$ to j
2. 随机选择小于 M 的素数 p // M为正整数
3. A 发送 p 和 $I_p(x)$ 给 B
4. B 测试
5. if $I_p(x) \neq I_p(y)$
6. then return "No"
7. return "Yes"

算法分析

令 $j = \lceil \log\log n \rceil$，则算法出错的概率

$$\left(\frac{1}{n}\right)^j \le \frac{1}{n^{\lceil \log\log n \rceil}}$$

实例：x 和 y 是1000000位二进制数

$n = 10^6$

$M = 2 \cdot 10^{12} = 2^{40.8631}$

素数 p 的二进制表示至多 $\lfloor \log M \rfloor + 1 = 41$ 位

$I_p(x)$ 的位数至多 $\lfloor \log(p-1) \rfloor + 1 \le \lfloor \log M \rfloor + 1 = 41$

总共传送 82 位

模式匹配

问题：输入二进制串 $X = x_1 x_2 \ldots x_n$， $Y = y_1 y_2 \ldots y_m$， $m \le n$
输出：若Y在 X 中，Y 出现的第一个位置；否则为 “0”

算法一：顺序比较

初始 Y 与 X 的首元素对齐，依次从前到后比较 X 与 Y 的元素. 如果 X 与 Y 的所有元素都相等，输出 Y 的首位置 j；否则将 Y 的位置向后移动一个字符，重复原来过程.

运行时间：$O(mn)$

算法二：利用有限状态自动机的模式匹配算法

(Knuth,Morris,Pratt), Introduction to Algorithms

运行时间：$O(m+n)$

算法三　利用串比较的随机算法

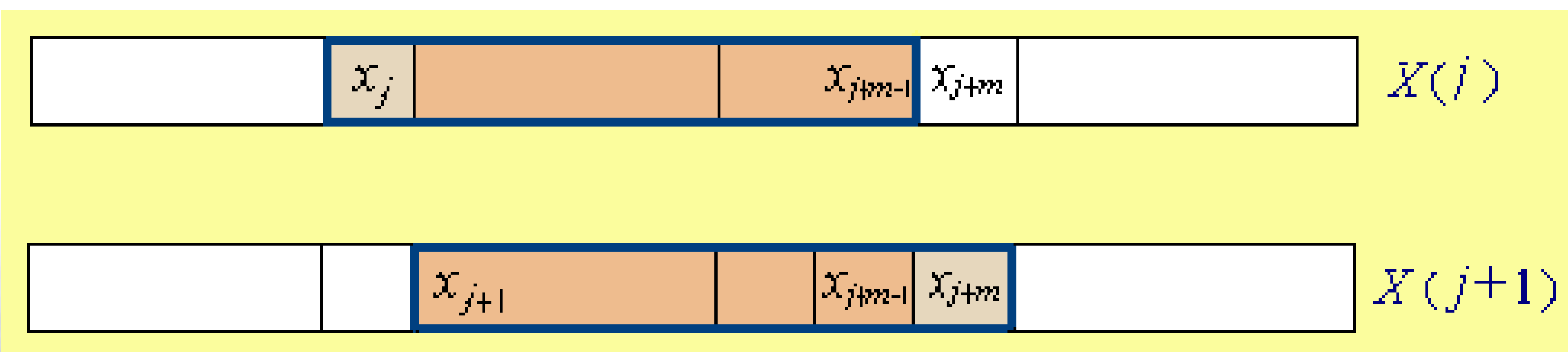
设计思想：设 $X(j) = x_j x_{j+1} \dots x_{j+m-1}$，把 $X(j)$ $(j=1,2,\dots,n-m+1)$ 与 Y 逐个字符的比较，改成对指纹 $I_p(X(j))$ 与 $I_p(Y)$ 的比较

$X(j)$与$X(j+1)$的关系

$$X(j) = 2^{m-1}x_j + \underline{2^{m-2}x_{j+1} + \dots + 2x_{j+m-2} + x_{j+m-1}}$$

$$X(j+1) = \underline{2^{m-1}x_{j+1} + 2^{m-2}x_{j+2} + \dots + 2x_{j+m-1}} + x_{j+m}$$

$$X(j+1) = 2X(j) - 2^m x_j + x_{j+m}$$



算法三的关键技术

由 $I_p(X(j))$ 求 $I_p(X(j+1))$ 的公式

$$I_p(X(j+1)) = (2I_p(X(j)) - W_p x_j + x_{j+m})(\mathrm{mod}\ p)$$

$$W_p = 2^m (\mathrm{mod}\ p)$$

该公式说明：由 $I_p(X(j))$ 计算 $I_p(X(j+1))$ 仅需要常数时间

公式的证明：

$$X(j+1) = 2X(j) - 2^m x_j + x_{j+m}$$

$$I_p(X(j+1)) = (2I_p(X(j)) - 2^m x_j + x_{j+m})(\mathrm{mod}\ p)$$

令 $W_p = 2^m (\mathrm{mod}\ p)$

$$I_p(X(j+1)) = (2I_p(X(j)) - W_p x_j + x_{j+m})(\mathrm{mod}\ p)$$

算法

算法 PatternMaching

输入：串 X 和 Y, $|X|=n$, $|Y|=m$, $m \le n$

输出：如果 Y 在 X 中，Y 出现的第一位置；否则为“0”

1. 从小于 M 的素数集合中随机选择素数 p
2. $j \leftarrow 1$
3. $W_p \leftarrow 2^m (\bmod\ p)$
4. $I_p(X(j)) \leftarrow I(X(j))(\bmod\ p)$
5. $I_p(Y) \leftarrow I(Y)(\bmod\ p)$
6. while $j \le n-m+1$ do
7.   if $I_p(X(j)) = I_p(Y)$ then return j
8. $I_p(X(j+1)) \leftarrow (2I_p(X(j)) - W_p x_j + x_{j+m})(\bmod\ p)$
9. $j \leftarrow j+1$
10. return 0

算法分析

时间复杂度为 $O(m+n)$

$W_p, I_p(Y), I_p(X(1))$ 计算 $O(m)$ 时间

从 $I_p(X(j))$ 计算 $I_p(X(j+1))$ 总共需要$O(n)$时间

出错条件：
$$Y \neq X(j) \wedge I_p(Y) = I_p(X(j))$$
$$\Leftrightarrow p \mid \prod_{\{j \mid Y \neq X(j)\}} |I(Y) - I(X(j))|$$

出错概率：乘积大小不超过 $(2^m)^n$，整除它的素数个数不超过 $\pi(mn)$，选 $M = 2mn^2$，则出错概率不超过

$$\frac{\pi(mn)}{\pi(M)} \approx \frac{mn/\ln(mn)}{2mn^2/\ln(mn^2)} = \frac{\ln(mn^2)}{2n\ln(mn)} < \frac{\ln(mn)^2}{2n\ln(mn)} = \frac{1}{n}$$

总结

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