HIT CS&E

第八章

Randomized Algorithms

骆吉洲
计算机科学与工程系

HIT CS&E

提要

HIT CS&E

参考文献

- 《计算机算法设计与分析》
  - 第7章
- 《Randomized Algorithms》

  Rajeev Motwani and Prabhakar, Raghavan, Cambridge University Press
- 《网站资料》
  - 第8章

HIT CS&E

8.1 Introduction to Randomized Algorithms

- 随机算法的基本概念
- 随机算法的分类
- 随机算法的性能分析方法

HIT CS&E

随机算法的基本概念

- 什么是随机算法
  - 随机算法是一种使用概率和统计方法在其执行过程中对于下一计算步骤作出随机选择的算法
- 随机算法的优越性
  - 对于有些问题：算法简单
  - 对于有些问题：时间复杂性低
  - 对于有些问题：同时兼有简单和时间复杂性低
- 随机算法的随机性
  - 对于同一实例的多次执行，效果可能完全不同
  - 时间复杂性的一个随机变量
  - 解的正确性和准确性也是随机的

HIT CS&E

随机算法的分类

- 随机数值算法
  - 主要用于数值问题求解
  - 算法的输出往往是近似解
  - 近似解的精确度与算法执行时间成正比
- Monte Carlo算法
  - 主要用于求解需要准确解的问题
  - 算法可能给出错误解
  - 获得精确解概率与算法执行时间成正比

HIT CS&E

- Las Vegas算法
  - 一旦找到一个解，该解一定是正确的
  - 找到解的概率与算法执行时间成正比
  - 增加对问题反复求解次数，可是求解无效的概率任意小
- Sherwood算法
  - 一定能够求得一个正确解
  - 确定算法的最坏与平均复杂性差别大时，加入随机性，即得到Sherwood算法
  - 消除最坏行为与特定实例的联系

HIT CS&E

随机算法的性能分析

- 随机算法分析的特征
  - 仅依赖于随机选择，不依赖于输入的分布
  - 确定算法的平均复杂性分析：
    依赖于输入的分布
  - 对于每个输入都要考虑算法的概率统计性能
- 随机算法分析的目标
  - 平均时间复杂性：时间复杂性随机变量的均值
  - 获得正确解的概率
  - 获得优化解的概率
  - 解的精确度估计

HIT CS&E

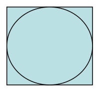
8.2 Randomized Numerical Algorithms

- 计算π值
- 计算定积分

HIT CS&E

计算π值

- 数学基础
  - 设有一个半径为r的圆及其外切四边形



  - 向正方形随机地投掷n个点，设k个点落入圆内
  - 投掷点落入圆内的概率为 $(\pi r^2)/(4r^2)=\pi/4$.
  - 用k/n逼近$\pi/4$，即$k/n\approx\pi/4$，于是$\pi\approx(4k)/n$.
  - 我们可以令$r=1$用投掷n个点的方法计算π

- 算法

  $K=0$;

  For $i=1$ To n

  随机地产生四边形中的一点(x, y);

  If $x^2+y^2\le 1$ Then $k=k+1$;

  Endfor

  Return $(4k)/n$
- 时间复杂性$=O(n)$
  - 不是输入的大小，而是随机样本的大小
- 解的精确度
  - 随着随机样本大小n增加而增加

HIT CS&E

计算定积分

- 问题
  - 计算积分 $\int_a^b g(x)dx$
- 数学基础
  - 令$f(x)$是区间$[a, b]$上的一组独立、同分布的随机变量$\{\xi_i\}$的任意密度函数
  - 令$g^*(x)=g(x)/f(x)$，则$\{g^*(\xi_i)\}$是密度为$f(x)$的随机变量集合，而且

$$E(g^*(\xi_i))=\int_a^b g^*(x)f(x)dx=\int_a^b g(x)dx=I$$

$$E(g^*(\xi_i)) = \int_a^b g^*(x)f(x)dx = \int_a^b g(x)dx = I$$

- 由强大数定律 $\Pr\left(\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} g^*(\xi_i) = I\right) = 1$
- 我们可以用 $\left(\frac{1}{n}\sum_{i=1}^{n} g^*(\xi_i)\right)$ 来近似计算 I
- 令 $f(x)=1/(b-a)$ $\quad a \le x \le b$
- 所求积分可以由如下 I' 来近似计算 I

$$I' = \frac{1}{n}\sum_{i=1}^{n} g^*(\xi_i) = \frac{1}{n}\sum_{i=1}^{n} g(\xi_i)/f(\xi_i) = \frac{1}{n}\sum_{i=1}^{n}(b-a)g(\xi_i)$$

HIT CS&E

- **算法**

  $I=0$;
  For $i=1$ To n
  随机产生 $[a, b]$ 中点 x;
  $I=I+g(x)$;
  Endfor
  Return $(b-a)*I/n$
- **时间复杂性** $=O(n)$
  - 不是输入的大小，而是随机样本的大小
- **解的精确度**
  - 随着随机样本大小 n 增加而增加

HIT CS&E

8.3 Randomized Selection Algorithms

- 问题的定义
- 随机算法
- 算法的性能分析

HIT CS&E

问题的定义

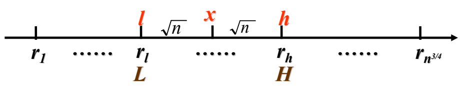
- 输入：$S=\{x_1, x_2, \ldots, x_n\}$，
  整数 k，$1\le k\le n$.
- 输出：S 中第 k 个最小元素.

记号
$\text{Rank}(Q, t)$ = 集合 Q 中的元素 t 的rank
(第 k 小元素的rank是 k)
$\min(Q, i)$ = 集合 Q 中第 i 个最小元素.

HIT CS&E

随机算法

- **基本思想**
  - 从 S 中随机地抽取 $n^{3/4}$ 个样本存入 R，排序 R
  - S 中第 k 最小元素可能成为 R 中 $x=kn^{3/4}/n$ 最小元素
  - 为了解决误差问题，我们考察区间 $[x-n^{1/2}, x+n^{1/2}]$



  - 把 S 中属于 $[L, H]$ 数据存入 P
  - 在 P 中查找 $\min(S, k)$

LAZYSELECT(S, k)

1. R=独立、均匀、可放回地从 S 随机选取的 $n^{3/4}$ 元素;
2. 在 $O(n)$ 时间内排序 R;
3. $x=(k/n)n^{3/4}$; /* $(k/n)n^{3/4}=kn^{-1/4}$ */
4. $l=\max\{\lfloor x-\sqrt{n}\rfloor, 0\}$; $h=\min\{\lfloor x+\sqrt{n}\rfloor, n^{3/4}\}$;
5. $L=\min(R, l)$; $H=\min(R, h)$;
6. $L_p=Rank(S,L)$, $H_p=Rank(S,H)$; /* L 和 H 与 S 元素比较 */
7. $P=\{y\in S \mid L\le y\le H\}$;
8. If $\min(S, k)\in P$ and $|P|\le 4n^{3/4}+1$
   /* $\max(S, k)\in P$ 可由 $L_p\le k\le H_p$ 确定 */
9. Then 排序 P, $\min(S, k)=\min(P, (k-L_p))$, 算法结束;
10. ELSE goto 1.

HIT CS&E

算法的性能分析

- **数学基础**
  - 数学期望
    - 离散随机变量X的数学期望$E[X]=\sum_i i\times P(X=i)$
    - 若$f(x)$是定义在整数集上的实数值函数，则
      $E[f(X)]=\sum_i f(i)\times P(X=i)$.
  - Markov不等式
    - $P(Y\geq t)\leq E[Y]/t$, 其中Y为非负随机变量, $t>0$.

HIT CS&E

- 方差的性质与Chebyshev不等式
  - 方差$\sigma_x^2=E[(X-\mu_x)^2]$, μ_x为随机变量X的数学期望
  - σ_x称为标准差
  - Chebyshev不等式：$P(|X-\mu_x|>t\sigma_x)\leq 1/t^2$
  - 如果随机变量X满足$P(X=1)=p$, $P(X=0)=1-p$, 则
    $\sigma_x^2=p(1-p)$.
  - 若$X=\sum_{1\leq i\leq n}X_i$, $\sigma_x^2=\sum_{1\leq i\leq n}\sigma_{x_i}^2$, X_i是独立随机变量
  - 若随机变量X_i满足$P(X_i=1)=p$, $P(X_i=0)=1-p$, 则
    $\sigma_x^2=np(1-p)$.

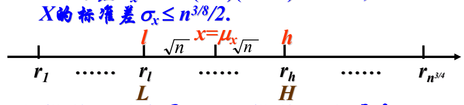
HIT CS&E

- **算法的性能分析**

定理. 算法执行1-9步一遍就可求出$min(S,k)$的概率是$1-O(n^{-1/4})$,
即算法需要$O(n)$次比较就可求出$min(S,k)$的概率是$1-O(n^{-1/4})$.

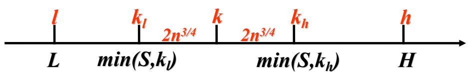
证明. 若算法执行1-9一遍可求出$min(S, k)$, 则第6步需$2n$次比较,
其他步需$O(n)$次比较, 总需$O(n)$次比较.
往证算法执行1-9一遍可求出$min(S, k)$的概率是$1-O(n^{-1/4})$.
算法执行1-9一遍可求出$min(S, k)$的条件是:
(1). $min(S, k)$在L和H之间即P包含$min(S, k)$,
(2). $|P|\leq 4n^{3/4}+1$.
我们首先来计算上述两个条件失败的概率.

A. 计算条件(1)不成立的概率
条件(1)不成立当且仅当$L>min(S, k)$或$H<min(S, k)$.
令$X_i=1$如果第i个随机样本$\leq min(S, k)$, 否则$X_i=0$.
于是, $P(X_i=1)=k/n$, $P(X_i=0)=1-k/n$.
令 $X=\sum_{1\leq i\leq n^{3/4}}X_i$是$R$中小于等于$min(S, k)$ 的样本数. 我们有
X的数学期望$\mu_x=n^{3/4}k/n=kn^{-1/4}$,
X的方差$\sigma_x^2= n^{3/4}(k/n)(1-k/n)\leq n^{3/4}/4$,
X的标准差$\sigma_x\leq n^{3/8}/2$.



如果$L>min(S, k)$, $X<l$. 如果$H<min(S, k)$, $X\geq h$. 于是
$P(L>min(S, k))=P(X<l)= P(X<\mu_x-n^{1/2})=P(|X-\mu_x|>n^{1/2})$,
$P(H<min(S, k))=P(X\geq h)=P(X>h)+P(X=h)=P(|X-\mu_x|\geq n^{1/2})+(n^{3/4}+1)^{-1}$.
应用Chebyshev不等式, 又由$2n^{1/8}\sigma_x\leq n^{1/2}$, 我们有
$P(|X-\mu_x|>n^{1/2})\leq P(|X-\mu_x|>2n^{1/8}\sigma_x)\leq 1/(2n^{1/8})^2=O(n^{-1/4})$. 于是
$P(L>min(S, k))=P(H<min(S, k))=O(n^{-1/4})$.

B. 计算P包含$min(S, k)$但$|P|\leq 4n^{3/4}+1$不成立的概率
令$k_l=\max\{0, k-2n^{3/4}\}$, $k_h=\min\{k+2n^{3/4}, n\}$.
"P包含$min(S, k)$但$|P|\leq 4n^{3/4}+1$不成立"发生当且仅当
$L<min(S, k_l)$或$H>min(S, k_h)$.
类似与上面A中的分析，
$P(L<min(S, k_l))=P(H>min(S, k_h))=O(n^{-1/4})$.
由A和B, "算法执行1-9一遍就可以求出$min(S, k)$"不成立的概率是$O(n^{-1/4})$.
即，"算法执行1-9一遍就可以求出$min(S, k)$"的概率是$1-O(n^{-1/4})$.



HIT CS&E

8.4 Randomized Algorithm to Test Whether Number is Prime

- 问题的定义
- 随机算法设计
- 算法的性能分析

HIT
CS&E

问题的定义

- **输入**
  - 一个正整数N
- **输出**
  - N是否素数

HIT
CS&E

随机算法的设计

- **基本思想**
  - 对N进行m次测试
  - 如果有一次测试成功，则回答N是合数
  - 如果m次测试均失败，则回答N是素数
  - 回答N是合数时，答案百分之百正确
  - 回答N是素数时，答案正确的概率是$1-2^{-m}$

- **随机算法**

1. 随机地选择m个数$\{b_1, b_2, \ldots, b_m\}$，满足
   $1 \le b_1, b_2, \ldots, b_m \le N$;
2. For $i=1$ To m Do
3. If $W(b_i)$成立 Then Return (N是合数);
4. Return (N是素数)

$W(b_i)$定义如下：

(1) $b_i^{N-1} \neq 1 \bmod N$, 或

(2) $\exists j[(N-1)/2^j=k$是整数, $1 < (b_i^k$与N的最大公因子$) < N]$.

HIT
CS&E

例1. 给定$N=12$. 选择测试数集$\{2, 3, 7\}$

测试 2: $2^{12-1} = 2048 \neq 1 \bmod 12$, $W(2)$成立.

N是合数.

HIT
CS&E

例2. 给定$N=11$,选择测试数集$\{2, 5, 7\}$

测试 2: $2^{11-1} = 1024 = 1 \bmod 11$,

测试 5: $5^{11-1} = 9765625 = 1 \bmod 11$,

测试 7: $7^{11-1} = 282475249 = 1 \bmod 11$,

结论: 11可能是素数

答案正确的概率为$1-2^{-3}$

HIT
CS&E

算法性能的分析

定理1. (1) 如果对于任意$1 \le b < N$, $W(b)$成立，则N是合数.

(2) 如果N是合数，则$(N-1)/2 \le |\{b \mid 1 \le b < N, W(b)\}|$

*(1)说明算法是正确的.

*(2)说明，如果N是合数，则至少一半$b(b<N)$使$W(b)$成立

定理2. 算法的回答"N是素数"正确的概率是$1-2^{-m}$.

HIT CS&E

8.5 Randomized Sorting Algorithm

- 问题的定义
- 随机算法
- 算法性能的分析

HIT CS&E

问题的定义

- 输入
  - $S=\{x_1, x_2, \ldots, x_n\}$
- 输出
  - 排序的S

HIT CS&E

随机算法

- 基本思想
  - 采用随机抽样的方法确定集合的划分点
  - 把集合划分为两个子集合
  - 分别递归地在每个子集合上使用随机排序算法

HIT CS&E

- 算法
  1. 均匀等可能地在S中随机抽取一个样本y;
  2. 比较S中每个元素，把S划分为如下两个集合:

     $S_1=\{x \mid x\in S, x<y\}$, $S_2=\{x \mid x\in S, x>y\}$;
  3. 递归地排序S_1和S_2;
  4. 顺序输出排序的S_1, y, S_2;

HIT CS&E

算法性能的分析

- 基本概念
  - $S_{(i)}$表示S中阶为i的元素

    例如，$S_{(1)}$和$S_{(n)}$分别是最小和最大元素
  - 随机变量X_{ij}定义如下:

    $X_{ij}=1$如果$S_{(i)}$和$S_{(j)}$在运行中被比较，否则为0
  - X_{ij}是$S_{(i)}$和$S_{(j)}$的比较次数
  - 算法的比较次数为 $\sum_{i=1}^{n}\sum_{j>i}X_{ij}$
  - 算法的平均复杂性为 $E[\sum_{i=1}^{n}\sum_{j>i}X_{ij}]=\sum_{i=1}^{n}\sum_{j>i}E[X_{ij}]$

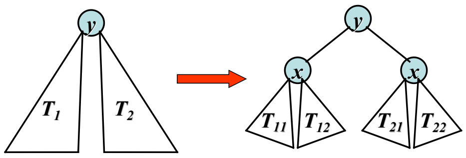
HIT CS&E

- 计算$E[X_{ij}]$
  - 设p_{ij}为$S_{(i)}$和$S_{(j)}$在运行中比较的概率，则

    $E[X_{ij}]=p_{ij}\times 1+(1-p_{ij})\times 0=p_{ij}$

关键问题成为求解p_{ij}

- 求解P_{ij}
  - 我们可以用树表示算法的计算过程



  - 我们可以观察到如下事实：
    - 一个子树的根必须与其子树的所有节点比较
    - 不同子树中的节点不可能比较
    - 任意两个节点至多比较一次

HIT CS&E

- 当$S_{(i)}, S_{(i+1)}, \ldots, S_{(j)}$在同一子树时，$S_{(i)}$和$S_{(j)}$才可能比较
- 由随机算法的特点，$S_{(i)}, S_{(i+1)}, \ldots, S_{(j)}$在同一子树的概率为1
- 只有$S_{(i)}$或$S_{(j)}$被选为划分点时，$S_{(i)}$和$S_{(j)}$才可能比较
- $S_{(i)}, S_{(i+1)}, \ldots, S_{(j)}$等可能地被选为划分点，所以$S_{(i)}$和$S_{(j)}$
  进行比较的概率是：$2/(j-i+1)$，即

$$p_{ij}=2/(j-i+1)$$

HIT CS&E

- 现在我们有

$$\sum_{i=1}^{n}\sum_{j>i}E[X_{ij}]=\sum_{i=1}^{n}\sum_{j>i}p_{ij}=\sum_{i=1}^{n}\sum_{j>i}\frac{2}{j-i+1}$$

$$\le\sum_{i=1}^{n}\sum_{k=1}^{n-i+1}\frac{2}{k}\le 2\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{1}{k}=2nH_n=O(n\log n)$$

定理．随机排序算法的期望时间复杂性为$O(n\log n)$

HIT CS&E

8.6 A Min-Cut Algorithm

- 问题定义
- 随机算法
- 算法性能的分析

HIT CS&E

问题定义

- 输入：
  - 无向多重连通图G
- 输出
  - 一个Min-Cut

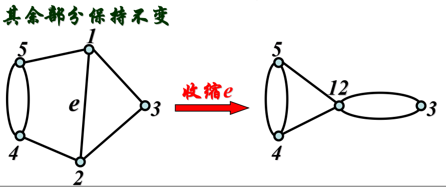
- 图G的一个Cut是一组边，从G中删除这个Cut将导致两个或多个连通分量
- Cut的大小是其边数，多重边重复计算
- 最小Cut是具有最少边的Cut

HIT CS&E

随机算法

- 基本概念
  - Cut可以视为节点集的划分$V=(C, V-C)$, Cut是所有G中连接C和$V-C$的边集合.
  - 图G的边(x, y)的contraction:
    - 用新节点代替节点x和y或边(x, y),
    - $\forall v\in V$，用边(v, z)代替边(x, v)或(y, v),
    - G的其余部分保持不变

例如：



HIT CS&E

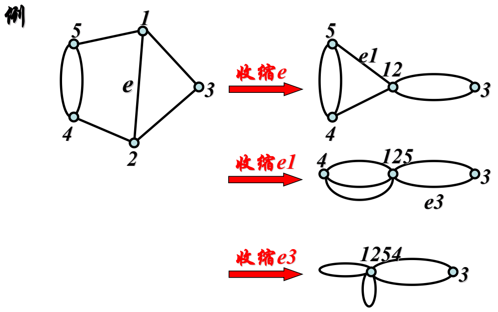
- 我们用$G/(x, y)$表示G的边(x, y)的收缩
- 边集合$F \subseteq G$收缩记作G/F
- 图G/F的节点集合表示为V/F
- 图G/F的节点集合表示为E/F

HIT CS&E

- 随机算法

1. $H=G$;
2. While $|H(V)|>2$ Do
3. 随机地从$H(E)$中选择一条边(x, y);
4. $F=F \cup \{(x, y)\}$;
5. $H=H/(x, y)$;
6. Cut=连接H中两个元节点的G的所有边.

例



$Cut=\{(1,3), (2, 3)\}$

HIT CS&E

算法的性能分析

定理1. 如果算法的输入是具有n个节点的多重图，则算法的时间复杂性为$O(n^2)$.

证明. 一次边收缩需要$O(n)$时间.
至多进行O(n)次收缩.
于是，算法时间复杂性为$O(n^2)$.

注意:

我们仅证明了在$O(n^2)$时间内算法能够求出一个Cut，但是这个Cut不一定是优化的.

HIT CS&E

引理1. 如果k是min-cut的大小，则G至少有$kn/2$条边.

证. 如果$|G(E)|<kn/2$，则存在一个度小于k的节点p.
删除与p相关连的k条，把G划分为两个连通分量，其一是仅包含p.
于是，与p相关连的边集合是一个cut.
但是这个cut的大小$<k$，与min-cut大小为k矛盾.

引理2. 算法输出的cut是连接两个剩余节点的没有被收缩过的边.

证. 从算法定义可以看到，算法输出的cut是连接两个剩余节点的没有被收缩的边的集合.

HIT CS&E

引理1. 如果k是图G的一个min-cut的大小，则G至少有$kn/2$条边.

证. 如果$|G(E)|<kn/2$，则存在一个度小于k的节点p.
删除与p相关连的k条，把G划分为两个连通分量，其一是仅包含p的连通分量.
于是，与p相关连的边集合是一个cut.
但是这个cut的大小$<k$，与min-cut大小为k矛盾.

定理2. 设C是一个min-cut, 其大小为k. 在算法结束时, C中无边被收缩过的概率大于$2/n^2$.

证. A_i表示第i步没有选中C的边, $1 \le i \le n-2$.

在第1步, 选中的边在C中的概率至多为$k/(kn/2)=2/n$, 即

$$Pr(A_1) \ge 1-2/n.$$

在第2步, 若A_1发生, 则至少有$k(n-1)/2$条边(每次收缩减少一个节点), 选中C中边的概率为$2/(n-1)$, 即

$$Pr(A_2 \mid A_1) \ge 1-2/(n-1).$$

在第i步, 若A_1至A_{i-1}发生, 则有$n-i+1$个节点, 即至少有$k(n-i+1)/2$条边, 于是

$$Pr(A_i \mid \cap_{1 \le j \le i-1} A_j) \ge 1-2/(n-i+1)$$

最后我们有

$$Pr(\cap_{1 \le i \le n-2} A_i) \ge \Pi_{1 \le i \le n-2}(1-2/(n-i+1)) = 2/n(n-1) > 2/n^2$$

HIT
CS&E

推论1. 如果重复运行算法$n^2/2$次, 每次独立随机地选择收缩边, 不能发现一个min-cut的概率为

$$\left(1-\frac{2}{n^2}\right)^{n^2/2} < \frac{1}{e}$$