HIT
CS&E

第七章

随机算法的设计与分析原理

程思瑶
计算机科学与技术学院

HIT
CS&E

Randomized Algorithms

Rajjev Motwani (*Stanford*) and Prabhakar Raghavan (*IBM*)
CMU Press

算法设计与分析

王晓东
清华大学出版社

提要

7.1 随机算法简介

7.2 随机数值分析算法

7.3 随机选择算法

7.4 随机最小割算法

7.5 随机快速排序算法

7.6 随机Max-3-CNF算法

7.1 随机算法简介

- 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

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
  - 时间复杂性是一个随机变量
  - 解的正确性和准确性也是随机的

随机算法的分类

- Las Vegas算法
  - 可能找不到解，但一旦找到一个解，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，可使求解无效的概率任意小
  - 设p是算法获得一个解的概率，$S(n)$和$E(n)$分别是算法一次求解成功或失败所需的平均时间。则算法找到解所需的平均时间$T(n)$为：

$$T(n) = p \times S(n) + (1-p) \times (E(n) + T(n))$$

$$= S(n) + \frac{1-p}{p} \times E(n)$$

随机算法的分类

- Monte Carlo算法
  - 算法可能给出错误解
  - 获得正确解的概率与算法执行时间成正比
  - 算法的优势
    - 设p为一个实数，且$0.5<p<1$。如果一个Monte-Carlo方法对问题任一实例的得到正确解的概率不小于p，则该算法是p正确的，且p-0.5叫做此算法的优势
  - 算法是一致的
    - 如果对于同一实例，Monte-Carlo算法不会给出两个不同的正确答案，则称该算法是一致的
    - 对于一个一致的p-正确Monte-Carlo算法，要提高获得正确解的概率，只要执行该算法若干次，并选择出现频次最高的解即可

随机算法的分类

- Monte Carlo算法
  - 算法是偏真的
    - 如果解某个判定问题的Monte-Carlo算法，当返回true时是一定正确的，但当返回false时有可能产生错误解，则这个算法是偏真的
    - 当多次调用一个偏真Monte-Carlo算法时，只要有一次调用返回true，就可以断定相应的解为true。

随机算法的分类

- 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
- 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

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
- 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期望
  - 获得正确解的概率/获得优化解的概率
  - 解的精确度分析

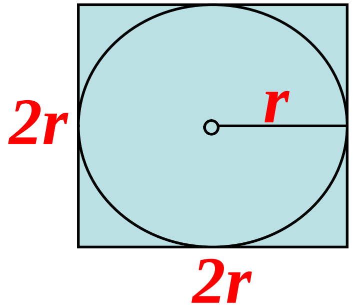
7.2 随机数值分析算法

- 计算π值
- 计算定积分

计算π值

- 随机投点法（频数法）
- 数学基础
  - 伯努利大数定律：设n_A是n次独立试验中事件A发生的次数，且事件A在每次试验中发生的概率为p，则对任意正数ε，有：

$$\lim_{n\to\infty} P\left(\left|\frac{n_A}{n} - p\right| < \varepsilon\right) = 1$$



- 方法
  - 设有一个半径为r的园及其外切四边形
  - 向正方形内随机地投掷，投掷点落入园内的概率为$(\pi r^2)/(4r^2) = \pi/4$.
  - 投掷n个点，设k个点落入园内
  - 用k/n逼近$\pi/4$，即$k/n \approx \pi/4$，于是$\pi \approx (4k)/n$.

- 算法

  $K=0$;

  For $i=1$ To n

  随机地产生四边形中的一点(x, y);

  If $x^2+y^2\leq 1$ 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

计算定积分

- 平均值法
- 问题
  - 计算定积分 $\int_a^b g(x)dx$
- 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$\{\xi_i\}$的密度函数
  - 令$g^*(x)=g(x)/f(x)$, 则$\{g^*(\xi)\}$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 $f(x)=1/(b-a)$ $a \le x \le b$

– 所求积分可以由下式来近似计算 I

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

- 算法

  $I=0;$

  For $i=1$ To n

  随机产生$[a, b]$中点x;

  $I=I+g(x);$

  Endfor /* $I=\sum_{i=1}^{n} g(\xi_i)$ */

  Return $I(b-a)/n$
- 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

7.3 随机选择算法

- 问题的定义
- 随机算法
- 算法的性能分析

问题的定义

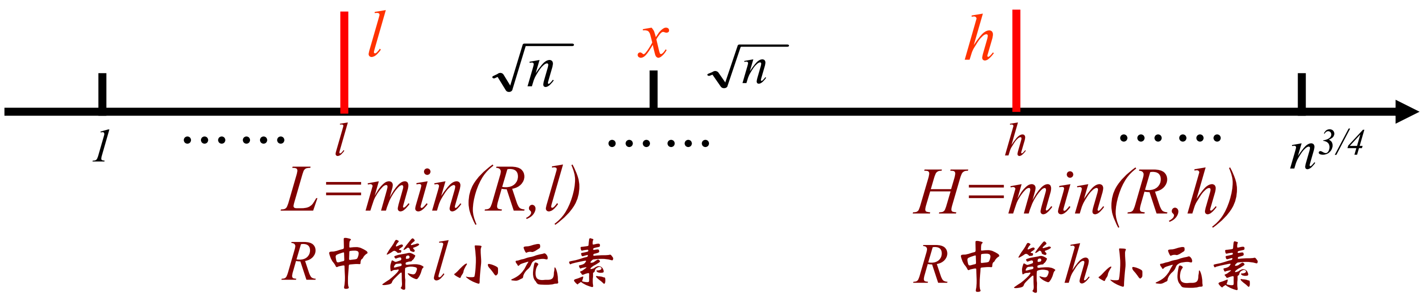
- 输入: $S=\{x_1, x_2, \ldots, x_n\}$,
  整数k, $1\le k\le n$.
- 输出: S中第k个最小元素.

记号

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小的元素.

- 基本思想
  - 从S中独立均匀可放回地抽取$n^{3/4}$个样本存入R, 排序R
  - S中第k小元素可能成为R中第$x=n^{3/4}k/n$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$min(S, k)$

Las Vegas算法

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小的元素.

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$O(n^{3/4}\log n^{3/4})$时间内排序R;
3. $x=(k/n)n^{3/4} = kn^{-1/4}$
4. $l=\max\{\lfloor x-\sqrt{n} \rfloor, 1\}$; $h=\min\{\lceil x+\sqrt{n} \rceil, n^{3/4}\}$;
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. $P=\{y\in S \mid L\leq y\leq H\}$;
8. If $min(S, k)\in P$ and $|P|\leq 4n^{3/4}+1$

   /* $min(S, k)\in P$可由$L_p\leq k\leq H_p$确定 */
9. Then 排序P, $min(S, k)=min(P,(k-L_p+1))$, 算法结束;
10. Else *goto 1*.

- 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，$t > 0$.
    - 当仅仅知道Y是非负的以及它的期望时，利用Markov可以得到一个可能的紧致界。但常常不能获得有用结果
    - 可以利用Markov不等式得到更好的界，例如Chebyshev界

– 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，是σ_X^2的正的平方根
- Chebyshev不等式: 对任意$t(t>0)$, $P(|X-\mu_X|>t\sigma_x) \le 1/t^2$

该不等式给出了任意随机变量的取值偏离其数学期望超过给定阈值的概率界

- 如果随机变量X满足$P(X=1)=p$, $P(X=0)=1-p$, 则

$$\sigma_X^2 = E[X^2]-E[X]^2 = p(1-p).$$

- 若$X=\sum_{1\le i\le n} X_i$, $\sigma_X^2=\sum_{1\le i\le n} \sigma_{X_i}^2$, 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

$$\sigma_X^2 = \sum_{1\le i\le n} \sigma_{X_i}^2 = \sum_{1\le i\le n} p(1-p) = np(1-p).$$

- 算法的性能分析

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

证明.

若算法执行1-9一遍求出$min(S,k)$,

则:

6、7步需$O(n)$次比较,

2、9步需$O(n^{3/4}\log n^{3/4}) = o(n)$次比较,

故总需$O(n)+o(n)=O(n)$次比较.

1. R=从S随机选取的$n^{3/4}$元素
2. 排序R;
3. $x=(k/n)n^{3/4}$;
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. $L_p=Rank(S,L)$, $H_p=Rank(S,H)$;
7. $P=\{y\in S \mid L\leq y\leq H\}$;
8. If $min(S, k)\in P$ and $|P|\leq 4n^{3/4}+1$
9. Then 排序P,
   $min(S,k)=min(P,(k-L_p+1))$,
   算法结束;
10. Else *goto 1*.

- 算法的性能分析

定理. 算法执行1-9步一遍求出$min(S, k)$的概率,即算法需要$O(n)+O(n^{3/4}\log n^{3/4})$次比较求出$min(S,k)$的概率是$1-O(n^{-1/4})$.

证明.

往证算法执行1-9一遍可求出$min(S, k)$的概率是$1-O(n^{-1/4})$.

执行1-9一遍求出$min(S, k)$的条件:

(1). 条件A: $min(S, k) \in P$,
即$min(S, k)$在L和H之间;

(2). 条件B: $|P| \leq 4n^{3/4}+1$.

我们首先来计算条件$(A \wedge B)$为假的概率是$O(n^{-1/4})$.

1. R=从S随机选取的$n^{3/4}$元素
2. 排序R;
3. $x=(k/n)n^{3/4}$;
4. $l=\max\{\lfloor x-\sqrt{n} \rfloor, 0\}$;
   $h=\min\{\lceil x+\sqrt{n} \rceil, n^{3/4}\}$;
5. $L=min(R, l)$; $H=min(R, h)$;
6. $L_p=Rank(S,L)$, $H_p=Rank(S,H)$;
7. $P=\{y \in S \mid L \leq y \leq H\}$;
8. If $min(S, k) \in P$ and $|P| \leq 4n^{3/4}+1$
9. Then 排序P,
   $min(S,k)=min(P,(k-L_p+1))$,
   算法结束;
10. Else *goto 1*.

$$
\begin{aligned}
\neg(A\wedge B) &= \neg A\vee \neg B \\
&= \neg A\vee((A\vee \neg A)\wedge\neg B) \\
&= \neg A\vee((A\wedge\neg B)\vee(\neg A\wedge\neg B)) \\
&= \neg A\vee(A\wedge\neg B)\vee(\neg A\wedge\neg B) \\
&= (\neg A\vee(\neg A\wedge\neg B))\vee(A\wedge\neg B) \\
&= \neg A\vee(A\wedge\neg B)
\end{aligned}
$$

- $x=(k/n)n^{3/4}=kn^{-1/4}$
- 离散随机变量X的数学期望$E[X]=\sum_i i\times P(X=i)$
- 如果随机变量X满足$P(X=1)=p$, $P(X=0)=1-p$, 则$\sigma_X^2=E[X^2]-E[X]^2=p(1-p)$.
- 若$X=\sum_{1\le i\le n}X_i$, $\sigma_X^2=\sum_{1\le i\le n}\sigma_{X_i}^2$, X_i是独立随机变量，且X_i满足$P(X_i=1)=p$, $P(X_i=0)=1-p$, 则 $\sigma_X^2=np(1-p)$.

(1). $\neg A=T$ 的概率, 即 $min(S, k)\notin P=\{y\in S \mid L\le y\le H\}$ 的概率

令$X_i=1$如果第i个随机样本$\le min(S, k)$, 否则$X_i=0$.

于是, $P(X_i=1)=k/n$, $P(X_i=0)=1-k/n$. 则$E(X_i)=k/n$

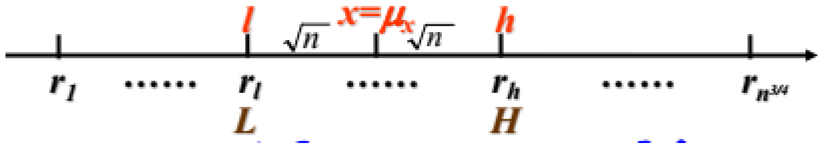
$X=\sum_{1\le i\le n^{3/4}}X_i$ 是R中小于等于$min(S, k)$ 的样本数.

则： X的数学期望$\mu_x=E(X)=\sum_{1\le i\le n^{3/4}}E(X_i)=n^{3/4}k/n=kn^{-1/4}=x$

X的方差$\sigma_x^2=|R|\times P(X_i=1)\times(1-P(X_i=1))=n^{3/4}(k/n)(1-k/n)\le n^{3/4}/4$

X的标准差$\sigma_x\le n^{3/8}/2$.

HIT



- $\mu_x=x=(k/n)n^{3/4}=kn^{-1/4}$
- Chebyshev不等式: 对任意$t(t>0)$, $P(|X-\mu_X|>t\sigma_x) \le 1/t^2$
- $X=\sum_{1\le i \le n^{3/4}} X_i$ 是R中小于等于$min(S, k)$的样本数.
- X的标准差$\sigma_x \le n^{3/8}/2$

由于$min(S, k) \notin P=\{y\in S \mid L\le y\le H\} \Leftrightarrow L>min(S, k)$ 或 $H<min(S, k)$.

说明R中小于等于$min(S, k)$的元素个数X太少或太多

$L>min(S, k)\Leftrightarrow X<l$，$H<min(S, k)\Leftrightarrow X\ge h$ (可能$min(S, k)\notin R$).

于是，$P(L>min(S, k))=P(X<l)= P(X<\mu_x-n^{1/2})\le P(|X-\mu_x|>n^{1/2})$,

$P(H<min(S, k))=P(X\ge h)=P(X>h)+P(X=h) \le P(|X-\mu_x|>n^{1/2})+(n^{3/4})^{-1}$.

应用Chebyshev不等式, 又由$2n^{1/8}\sigma_x \le n^{1/2}$, 我们有

$P(|X-\mu_x|>n^{1/2})\le P(|X-\mu_x|>2n^{1/8}\sigma_x)\le 1/(2n^{1/8})^2=O(n^{-1/4})$. 于是

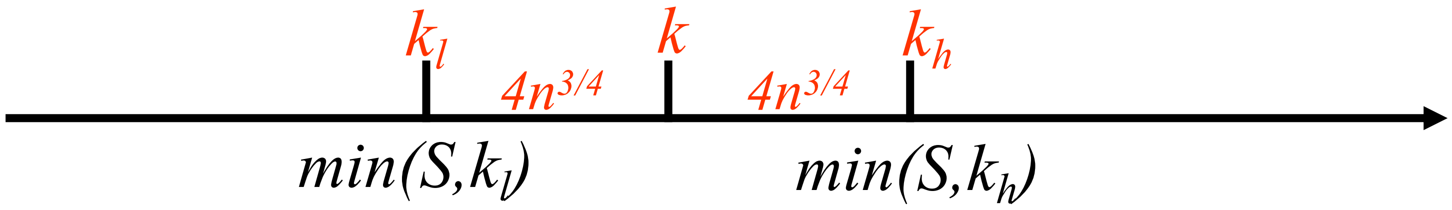
$P(L>min(S, k))=O(n^{-1/4})$, $P(H<min(S, k))=O(n^{-1/4})$.

HIT
CS&E

(2). P包含$min(S, k)$但$|P|\leq 4n^{3/4}+1$不成立的概率

$(A \wedge \neg B = T$的概率$)$

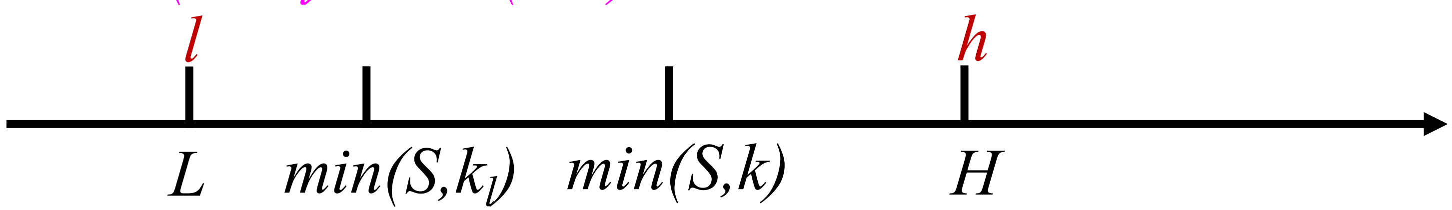
令$k_l=\max\{1, k-4n^{3/4}\}$, $k_h=\min\{k+4n^{3/4}, n\}$, 如下图所示



"P包含$min(S, k)$但$|P|\leq 4n^{3/4}+1$不成立"意味着

$L<min(S, k_l)<min(S,k)\leq H$或$L<min(S, k)<min(S,k_h)\leq H$

以$L<min(S, k_l)<min(S,k)\leq H$，如下图所示。



根据l和h的定义，分布在$[l,h]$中的样本数共为$2n^{1/2}$个。

因而，在区间$[min(S,k_l),min(S,k)]$中的样本数小于等于$2n^{1/2}$。

HIT
CS&E

(2). P包含$min(S, k)$但$|P|\le 4n^{3/4}+1$不成立的概率
($A\wedge\neg B=T$的概率)(续)

令$Y_i=1$如果第i个随机样本属于区间$[min(S,k_l),min(S,k)]$, 否则$Y_i=0$.

于是, $Pr(Y_i=1)=4n^{3/4}/n$, $Pr(Y_i=0)=1-4n^{3/4}/n$. 则$E(Y_i)=4n^{-1/4}$

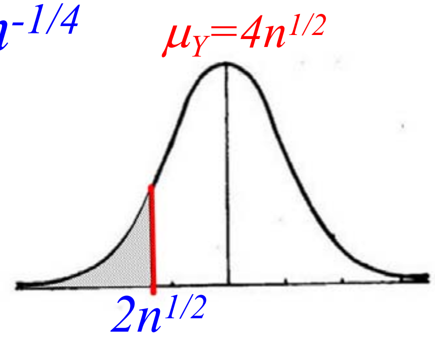
$Y=\sum_{1\le i\le n^{3/4}} Y_i$ 是处于区间$[min(S,k_l),min(S,k)]$的样本数.

则: Y的数学期望$\mu_Y=E(Y)=\sum_{1\le i\le n^{3/4}} E(Y_i)=4n^{1/2}$

X的方差$\sigma_Y^2=|R|\times Pr(X_i=1)\times(1-Pr(X_i=1))=n^{3/4}(4n^{-1/4})(1-4n^{-1/4})\le 4n^{3/4}$

根据Chebyshev不等式,

$Pr(Y\le 2n^{1/2})\le Pr(|Y-\mu_Y|\ge 2n^{1/2})\le\sigma_Y^2/(2n^{1/2})^2\le 4n^{3/4}/4n=n^{-1/4}$



即$Pr(L<min(S, k_l)<min(S,k)<=H)$

$=Pr(Y\le 2n^{1/2})\le n^{-1/4}=O(n^{-1/4})$

(2). P包含$min(S, k)$但$|P|\leq 4n^{3/4}+1$不成立的概率 ($A \wedge \neg B = T$的概率) (续)

同理可证$Pr(L<min(S,k)<min(S,kh)\leq H)=O(n^{-1/4})$,
即$Pr(A \wedge \neg B = T)=O(n^{-1/4})$.

由(1)和(2), "算法执行1-9一遍就可以求出$min(S, k)$"不成立的概率是$O(n^{-1/4})$.
于是, "算法执行1-9一遍就可以求出$min(S, k)$"的概率是$1-O(n^{-1/4})$.

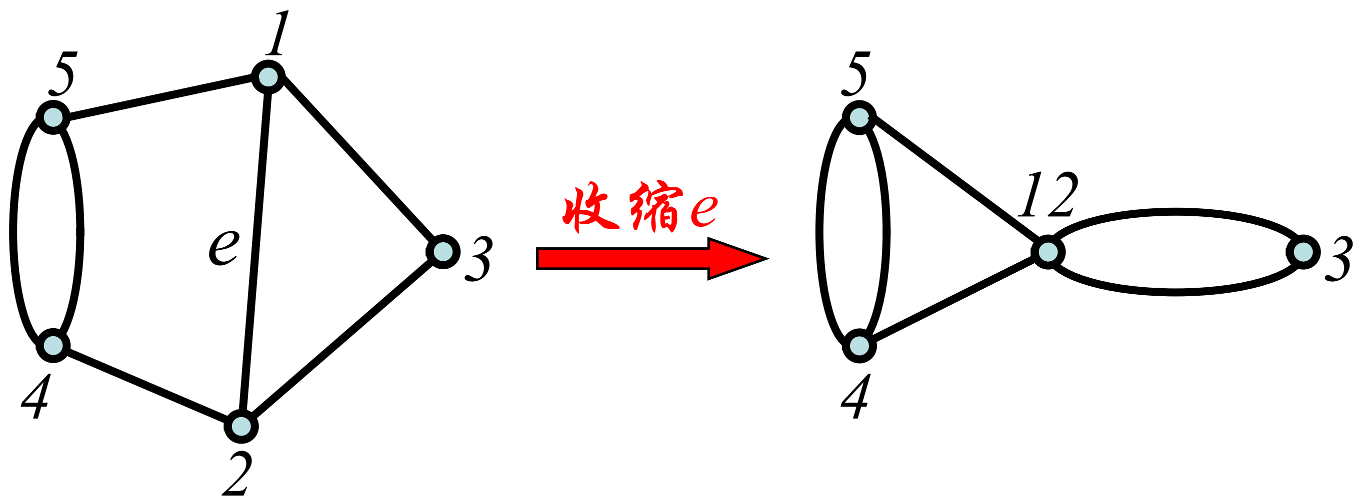
7.4 随机最小割算法

- 问题定义
- 随机算法
- 算法性能的分析

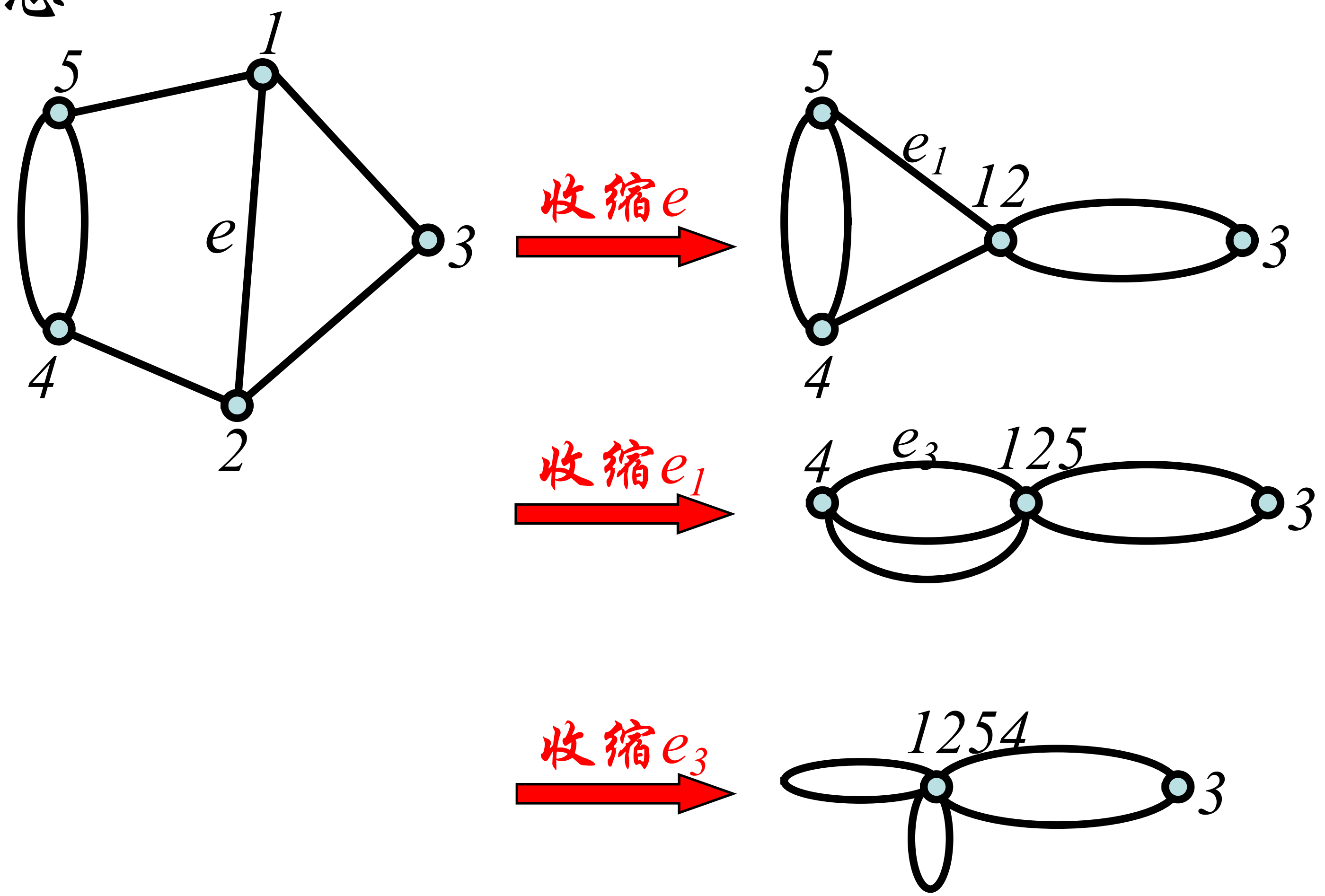
- 基本概念
  - 图G的一个Cut是一组边，从G中删除Cut里所有边将导致两个或多个连通分量
  - Cut的大小是其边数，多重边重复计算
  - Min-Cut是具有最少边的Cut

- 问题定义
  - 输入：无向多重连通图G
  - 输出：一个Min-Cut

- 基本概念
  - 可以把Cut看做是图G的顶点集的划分$P=(C, V-C)$, 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z代替边(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



- 基本思想



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

连接剩余两个顶点的G中没有被收缩的边

用$G/(x, y)$表示对图G的边(x, y)做收缩
$E(H)$表示图H的边集合

- 随机算法

1. H=G;
2. While $|V(H)| > 2$ Do
3. 随机地从$E(H)$中选择一条边(x, y);
4. F=$F\cup\{(x, y)\}$;
5. H=$H/(x, y)$;
6. Cut=连接H中剩余两个节点的G的所有边.

算法的时间复杂性为$O(n^2)$.

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. 若k是一个min-cut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相关连的边(少于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.

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.

引理1. 若k是一个min-cut的大小, 则G至少有$kn/2$条边.

定理2. 设C是一个min-cut, 其大小为k. 在算法结束时, C中的边没有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

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

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

$$Pr(A_2 | A_1) \ge 1-2/(n-1)=(n-3)/(n-1).$$

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

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

HIT
CS&E

$Pr(A_1) \geq 1-2/n=(n-2)/n$

$Pr(A_2 | A_1) \geq 1-2/(n-1)=(n-3)/(n-1)$

$Pr(A_3 | A_1 A_2) \geq 1-2/(n-2)=(n-4)/(n-2)$

……

$Pr(A_i | \cap_{1\leq j \leq i-1} A_j) \geq 1-2/(n-i+1)=(n-i-1)/(n-i+1)$

$$Pr(\cap_{1\leq i\leq n-2} A_i)= Pr(A_1)\times Pr(A_2 | A_1) \times\ldots\ldots \times Pr(A_{n-2} | \cap_{1\leq j \leq n-3} A_j)$$

$$\geq \frac{n-2}{n}\times\frac{n-3}{n-1}\times\frac{n-4}{n-2}\times\ldots\ldots\times\frac{3}{5}\times\frac{2}{4}\times\frac{1}{3}$$

$$=\frac{2}{n(n-1)} > \frac{2}{n^2}$$

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

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

证明: 由定理2，在每次算法结束时，C中边没有被收缩过的概率大于$2/n^2$.

在每次算法结束时，C中有边被收缩过(即没有发现*min-cut*)的概率至多$1-2/n^2$.

定理成立.

7.5 随机快速排序算法

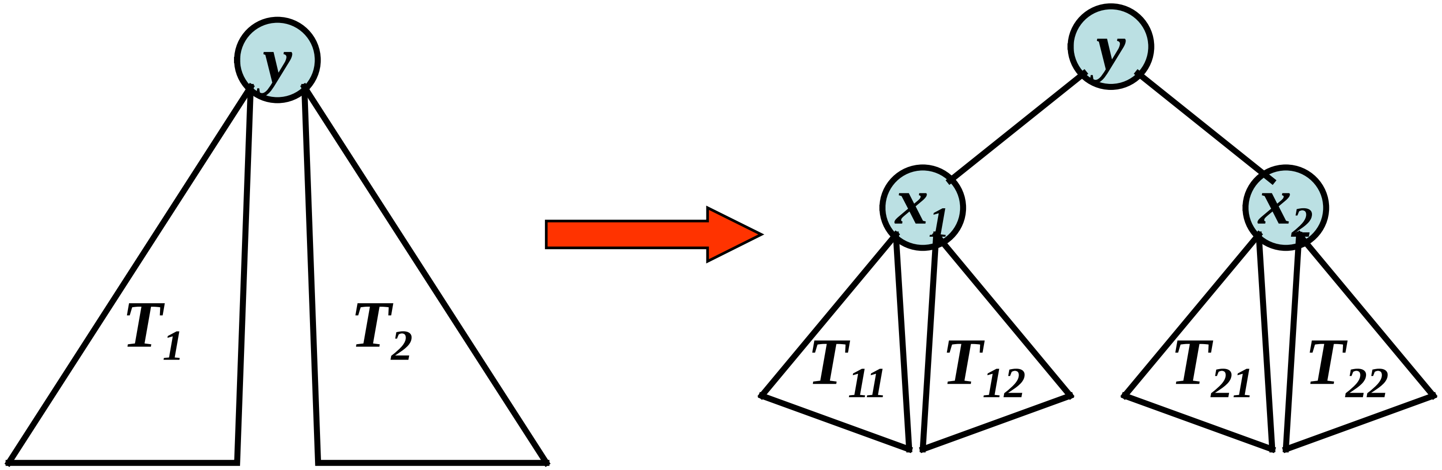
- 随机算法
- 性能分析

Randomized Algorithms

- **Randomized-Partition(A, p, r)**
  1. $i := Random(p, r)$
  2. $A[r] \leftrightarrow A[i]$;
  3. Return Partition(A, p, r)
- **Randomized-Quicksort(A, p, r)**
  1. **If** $p < r$
  2. **Then** $q :=$ Randomized-Partition(A, p, r);
  3. Randomized-Quicksort(A, p, q-1);
  4. Randomized-Quicksort(A, q+1, r).

Performance Analysis

- 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

- 基本概念
  - $x_{(i)}$表示A中Rank为i的元素

    例如，$x_{(1)}$和$x_{(n)}$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$x_{(i)}$和$x_{(j)}$在运行中被比较，否则为0

    X_{ij}是$x_{(i)}$和$x_{(j)}$的比较次数
  - 算法的比较次数为$\sum_{i=1}^{n}\sum_{j>i}X_{ij}$
  - 算法的复杂性为$T(n)=E[\sum_{i=1}^{n}\sum_{j>i}X_{ij}]=\sum_{i=1}^{n}\sum_{j>i}E[X_{ij}]$

$$T(n)=E[\sum_{i=1}^{n}\sum_{j>i}X_{ij}]=\sum_{i=1}^{n}\sum_{j>i}E[X_{ij}]$$

- 计算$E[X_{ij}]$
  - 设p_{ij}为$x_{(i)}$和$x_{(j)}$在运行中被比较的概率，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}
  - $Z_{ij}=\{x_{(i)}, x_{(i+1)}, \ldots, x_{(j)}\}$是$x_{(i)}$和$x_{(j)}$之间元素集合，$Z_{ij}$在同一棵子树时，$x_{(i)}$和$x_{(j)}$才可能比较.
  - $x_{(i)}$和$x_{(j)}$在执行中被比较，需满足下列条件：
    - Z_{ij}在同一棵子树，且
    - 当$x_{(i)}$或$x_{(j)}$被选为划分点
  - 设Z_{ij}在同一棵子树T中的概率为p
  - 一棵子树所有点等可能地被选为划分点，所以$x_{(i)}$或$x_{(j)}$被选为划分点的概率 = $2/|T|=2/(j-i+1)$.
  - $x(i)$和$x(j)$被进行比较的概率：

$$p_{ij}=p\times(2/(j-i+1))\leq 2/(j-i+1)$$

- 现在我们有

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

$$=\sum_{i=1}^{n}\sum_{k=1}^{n-i}\frac{2}{k+1}\leq\sum_{i=1}^{n}\sum_{k=1}^{n-i}\frac{2}{k}$$

$$\leq 2\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{1}{k}=2nH_{n}=O(n\log n)$$

定理. 随机排序算法的期望时间复杂性为$O(n\log n)$

7.6 随机Max-3-CNF 算法

- 问题的定义
- 随机算法
- 算法的性能分析

Max-3-CNF问题定义

输入:

合取范式*CNF*,

每个析取式具有三个变量,

没有任何变量和它的非出现在同一析取式中

输出:

一个变量赋值,最大化值为1的析取式个数

$-x_1$, $x_2 \vee x_3$, x_3, $-x_2$

求解Max-3-CNF问题随机算法

Random-Max-3-CNF(*CNF*)

1. For 对于*CNF*中的每个变量x Do
2. 随机地为x赋值:

   $x=0$的概率为$1/2$,

   $x=1$的概率为$1/2$;
3. Return.

定理. Random-Max-3-CNF是一个8/7-近似算法.

证. 假定输入CNF中具有n个变量, m个析取式, 第i个析取式的形式为 $x_{i1}\vee x_{i2}\vee x_{i3}$.

定义随机变量: $Y_i=1(1\leq i\leq m)$, 如果第i个析取式为1, 否则$Y_i=0$.

Pr(第i个析取式为0)$=Pr(x_{i1}=0)Pr(x_{i2}=0)Pr(x_{i3}=0)=(1/2)^3=1/8$.

Pr(第i个析取式为1)$=1- 1/8 = 7/8$.

$E[Y_i]=7/8$.

令$Y=Y_1+Y_2+\ldots+Y_m$. Y是CNF中值为1的析取式的个数.

$E[Y]=\sum_{1\leq i\leq m}E[Y_i]=\sum_{1\leq i\leq m}7/8=m7/8$.

显然, 优化解的代价$\leq m$. 于是近似比$\leq m/(m7/8)=8/7$.