大数据算法

Big Data Algorithms

- 概率基础
- 亚线性空间算法
- 亚线性时间算法
- 并行模型算法

概率基础

定义1.1[概率空间, probability space]

一个概率空间包含三要素：

- 样本空间(sample space)Ω：概率空间描述的随机行为所有可能结果
- 事件(events)$\mathcal{F}$：每个事件是Ω的一个子集合
- 概率函数$\Pr : \mathcal{F} \to [0, 1]$：满足概率测度要求

满足概率测度需要如下条件

- 事件集合$\mathcal{F}$包含Ω，对集合补操作和集合并操作封闭
- 概率函数满足
  - 非负性(Pr取值非负)
  - 规范性($Pr(\emptyset) = 0, Pr(\Omega) = 1$)
  - 完全可加性(对任意两两不相交的事件$A_1, \ldots, A_n$有 $Pr(\cup A_i) = \sum_i Pr(A_i)$)，事件序列是无穷可数的也成立

在通常的离散概率空间(discrete probability space)，每一个Ω中的元素都是一个基本事件，$\mathcal{F} = 2^{\Omega}$

概率基础

定义1.2[条件概率, conditional probability]

给定事件E_2，E_1发生的条件概率为

$$Pr(E_1|E_2) = \frac{Pr(E_1 \cap E_2)}{Pr(E_2)}$$

条件概率只有当$Pr(E_2) > 0$的时候才有意义

定义1.3[全概率公式, law of total probability]

令$E_1, E_2, \ldots, E_n$为概率空间的两两不相交事件且$\cup E_i = \Omega$，我们有

$$Pr(B) = \sum_{i=1}^{n} Pr(B \cap E_i) = \sum_{i=1}^{n} Pr(B|E_i)Pr(E_i)$$

概率基础

定义1.6[随机变量, random variable]

一个样本空间Ω上的随机变量X是一个实值函数$X : \Omega \mapsto \mathcal{R}$，一个离散随机变量是取值为有穷或者可数无穷多个值的变量

- 给定离散随机变量X和实数a，事件$X = a$代表集合$\{s \in \Omega : X(s) = a\}$

$$Pr(X = a) = \sum_{s \in \Omega : X(s) = a} Pr(s)$$

定义1.7[离散随机变量期望]

$$\mathbf{E}[X] = \sum_i i \cdot Pr(X = i)$$

定理1.2[期望的另一种计算方式]

X是一个取值为非负整数的离散随机变量，有$\mathbf{E}[X] = \sum_{i=1}^{\infty} Pr(X \geq i)$

概率基础

定义1.8[连续随机变量, continuous random variable]

一个随机变量X的概率分布由它的**分布函数**$F(x)$给出，对任意$x \in \mathbb{R}$，定义$F(x) = \Pr(X \leq x)$；X是连续随机变量，如果$F(x)$是x的连续函数。

- 连续随机变量X，必有$\Pr[X = x] = 0$，因此有$\Pr(X \leq x) = \Pr(X < x)$

定义1.9[概率密度函数]

如果存在函数$f(x)$，有

$$F(a) = \int_{-\infty}^{a} f(t)\, dt$$

，则称$f(x)$是$F(x)$的密度函数，且$f(x) = F'(x)$

定理1.3[连续随机变量的期望]

X一个连续随机变量，有$\mathbf{E}[X] = \int_{-\infty}^{+\infty} xf(x)\, dx$

概率基础

定理1.4[马尔可夫不等式]

对任意非负随机变量X和$a > 0$，有$Pr(X \geq a) \leq \frac{\mathbf{E}[X]}{a}$

定义1.11[方差]

$$Var[X] = \mathbf{E}[(X - \mathbf{E}[X])^2] = \mathbf{E}[X^2] - \mathbf{E}[X]^2$$

定理1.5[切比雪夫不等式]

对任意随机变量X和$a > 0$，有$Pr(|X - \mathbf{E}[X]| \geq a) \leq \frac{Var[X]}{a^2}$

定理1.6[切尔诺夫不等式, Chernoff/Hoeffding Bound]

令$X_1, X_2, \ldots, X_m$是独立的、取值$\in \{0, 1\}$的随机变量，变量和的期望为$\mu = \mathbf{E}[\Sigma_i X_i]$，令$\epsilon \in [0, 1]$，有$Pr[|\Sigma_i X_i - \mu| > \epsilon\mu] \leq 2e^{-\epsilon^2\mu/3}$.

- 概率基础
- 亚线性空间算法
- 亚线性时间算法
- 并行模型算法

问题1：近似计数

流模型（Streaming Model）

A stream:

$$\sigma = \langle a_1, a_2, \ldots, a_n \rangle, a_i \in [m],$$

then we can define a frequency vector

$$\mathbf{f} = (f_1, \ldots, f_m), \text{ s.t. } \sum_{1 \le i \le m} f_i = n.$$

定义1.13[The Counting Problem]
给定数据流 σ 以及a_i，计算f_i。

问题1：近似计数

为了计算 f_i，假设 a_i 共出现 n 次

- ▷ 需要 $O(\log n)$ 位的空间代价存储 n
- ▷ 如果 n 非常大，$O(\log n)$ 也许不可接受
- ▷ Can we maintain the count in less space?
  当然不可能，除非我们放宽要求
- ▷ 我们将设计一个占用$O(\log \log n)$位空间的近似算法

定义1.14[The Approximate Counting Problem]
给定数据流 σ 以及a_i，计算$\hat{f}_i$，满足

$$\mathbf{Pr}[|\hat{f}_i - f_i| > \epsilon f_i] < \delta$$

实际应用中，可令$\epsilon = 1/3$，$\delta = 1\%$

问题1：近似计数

Morris Algorithm

*/** Maintain a counter X **/*
1. $X \leftarrow 0$;
2. If the event a_i happens,
   Update $X \leftarrow X + 1$ with pro. $\frac{1}{2^X}$
3. return $\hat{f}_i = 2^X - 1$

Analysis of Morris Algorithm

- ▷ 计算期望 $\mathbf{E}[2^X - 1] = n$
- ▷ 计算方差 $\text{var}[2^X - 1] \le \dfrac{3n(n+1)}{2} + 1 = O(n^2)$
- ▷ 利用切比雪夫不等式

问题1：近似计数

定理1.10[切比雪夫不等式，Chebyshev Inequality]
对任意的 $\lambda > 0$，$\Pr[|X - \mathbf{E}[X]| > \lambda] \le \dfrac{\text{var}[X]}{\lambda^2}$.

定理1.11[推论，Chebyshev Inequality Corollary]
$X = \mathbf{E}[X] \pm \sqrt{\text{var}[X]/c}$ 成立的概率至少为 $1 - c$.

定理1.12[Morris算法分析]
至少以 $1 - c$ 的概率，Morris算法满足

$$2^{X_n} - 1 = \mathbf{E}[2^{X_n} - 1] \pm \sqrt{\text{var}[2^{X_n} - 1]/c} = n \pm O(n)$$

问题1：近似计数

如何获得一个$1 + \epsilon$近似算法

Morris+ Algorithm

*/** 重复 Morris 算法 k 次 **/*
1. Run k independent copies of Morris's algorithm.
2. Let the results be $(X_1, \ldots, X_k)$.
3. return $\frac{1}{k}\sum_{i=1}^{k}(2^{X_i} - 1)$.

Analysis of Morris+ Algorithm

- ▷ 计算期望 $\mathbf{E}[Y] = \mathbf{E}[\frac{1}{k}\sum_{i=1}^{k}(2^{X_i} - 1)] = n$
- ▷ 计算方差 $\text{var}[Y] = O(\frac{n^2}{k})$
- ▷ 利用切比雪夫不等式

问题1：近似计数

定理1.13[Analysis of Morris+ Algorithm]
至少以 $1 - c$ 的概率，Morris+算法满足

$$Y = n \pm \frac{1}{\sqrt{k}} O(n)$$

- ▷ 令$k = O(\frac{1}{\epsilon^2})$，上述算法是一个$(1 \pm \epsilon)$近似算法.
- ▷ 以常数概率如 $\frac{9}{10}$ 满足近似标准.

进一步，可以将概率改进到$1 - \delta$, 对任意$\delta > 0$.

问题1：近似计数

- ▷ 为了以$1 - \delta$概率得到$(1 \pm \epsilon)$-近似，Morris+需要$k = O(\frac{1}{\epsilon^2}\frac{1}{\delta})$次运行
- ▷ 假设，有Morris+算法以常数概率$c \ge 0.9$得到$1 \pm \epsilon$近似
- ▷ 设计一个$1 - \delta$概率$(1 \pm \epsilon)$-近似算法，代价为$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\log n)$

Morris++ Algorithm

*/** Median Trick **/*
1. 重复Morris+算法 $t = O(log(1/\delta))$ 次
2. 取 t 次估计结果的中间值（median）返回

Analysis of Morris++ Algorithm

- ▷ 计算期望
- ▷ 利用切尔诺夫不等式

问题1：近似计数

定理1.14[切尔诺夫不等式，Chernoff/Hoeffding Bound]
令$X_1, X_2, \ldots, X_m$是独立的、取值$\in \{0, 1\}$的随机变量，变量和的期望为$\mu = \mathbf{E}[\Sigma_i X_i]$，令$\epsilon \in [0, 1]$，有$\Pr[|\Sigma_i X_i - \mu| > \epsilon\mu] \le 2e^{-\epsilon^2\mu/3}$.

令随机变量 $X_i = 1 \Leftrightarrow$第 i 个Morris+算法运行的结果在正确范围内
$\mathbf{E}[X_i] = 0.9$ 且 $\mu = 0.9t$

易知，如果$\Sigma_i X_i > 0.5t$，那么Morris++算法将返回正确范围值

$$\begin{aligned}
\Pr[\Sigma_i X_i < 0.5t] &= \Pr[\Sigma_i X_i - 0.9t < -0.4t] \\
&\le \Pr[|\Sigma_i X_i - \mu| > 0.4t] \\
&= \Pr[|\Sigma X_i - \mu| > \mu 0.4/0.9] \\
\text{利用切尔诺夫不等式} \Rightarrow \quad &\le 2e^{-\frac{16}{81 \times 3} \cdot 0.9t} \\
&\le e^{-c't}
\end{aligned}$$

若想要 $\Pr[\Sigma_i X_i < 0.5t] \le e^{-c't} < \delta$，只需要 $t = O(log(1/\delta))$

问题2：不重复元素数

流模型（Streaming Model）

A stream:

$$\sigma = \langle a_1, a_2, \ldots, a_n \rangle, a_i \in [m],$$

then we can define a frequency vector

$$\mathbf{f} = (f_1, \ldots, f_m), \text{ s.t. } \sum_{1 \le i \le n} f_i = n.$$

定义1.15[The Distinct Elements Problem]

给定数据流 σ，计算$\sum_{1 \le i \le m} \mathbf{I}[f_i > 0]$，$\mathbf{I}[f_i > 0] = 1$当且仅当$f_i > 0$。

问题2：不重复元素数

精确计算需要多少空间代价？

- ▷ 方法1：$O(m)$ 位.
  为每一个$[m]$中的元素维护一个位，长度为m的向量.
- ▷ 方法2：$O(n\log(m))$ 位.
  维护n个数，每一个使用$log(m)$位.

问题2：不重复元素数

- ▷ 一个理想化的解决方案：假设可以存储实数
- ▷ 利用哈希函数（hash function）
  - ○ $h : [m] \mapsto [0,1]$
  - ○ $h(i)$的函数值是$[0,1]$实数，均匀分布

FM Algorithm [Flajolet-Martin 1985]

/** *Maintain a variable z* **/
1. 随机选取一个哈希函数$h : [m] \mapsto [0,1]$
2. $z = 1$.
3. 每遇到一个元素i，更新: $z = \min(z, h(i))$
4. return $1/z - 1$.

问题2：不重复元素数

FM算法的分析：令$X = \frac{1}{z} - 1$，$\mathbf{E}[z] = \frac{1}{D+1}$，$\mathrm{var}[z] \le \frac{2}{(D+1)(D+2)}$

$$\Pr[|z - \frac{1}{D+1}| > c\frac{1}{D+1}] < \frac{2(D+1)^2}{c^2 \cdot (D+1)(D+2)} < \frac{2}{c^2}$$

$$\begin{aligned} &\Pr[|X - D| > \epsilon D] \\ &\le \Pr[|z - \frac{1}{D+1}| > \frac{\epsilon D}{(D+1+\epsilon D)} \frac{1}{(D+1)}] \end{aligned}$$

$$c = \frac{\epsilon D}{D+1+\epsilon D}$$

$$\begin{aligned} &< \frac{2(D+1+\epsilon D)^2}{\epsilon^2 D^2} \\ &= 2(\frac{D+1}{\epsilon D} + 1)^2 \\ &< 2(\frac{2}{\epsilon} + 1)^2 \end{aligned}$$

问题2：不重复元素数

$$\begin{aligned} &\Pr[|X - D| > \epsilon D] \\ \Leftrightarrow &\Pr[|1/z - 1 - D| > \epsilon D] \\ \Leftrightarrow &\Pr[\frac{1}{z} < 1 + (1-\epsilon)D \text{或} \frac{1}{z} > 1 + (1+\epsilon)D] \\ \Leftrightarrow &\Pr[z > \frac{1}{1+(1-\epsilon)D} \text{或} z < \frac{1}{1+(1+\epsilon)D}] \\ \Leftrightarrow &\Pr[z - \frac{1}{1+D} > \frac{\epsilon D}{(D+1)(1+D-\epsilon D)} \\ &\text{或} z - \frac{1}{1+D} < \frac{-\epsilon D}{(D+1)(1+D+\epsilon D)}] \\ \le &\Pr[|z - \frac{1}{D+1}| > \frac{\epsilon D}{1+D+\epsilon D} \frac{1}{D+1}] \end{aligned}$$

问题2：不重复元素数

利用多次运行

FM+ Algorithm

/** *Maintain a variable z* **/
1. **for** j from 1 to k
2. 随机选取一个哈希函数$h_j : [m] \mapsto [0,1]$
3. $z_j = 1$.
4. 每次遇到 i，更新: $z_j = \min(z_j, h_j(i))$
5. $Z = \frac{1}{k}\sum_{j=1}^{k} z_j$;
6. **return** $1/Z - 1$.

问题2：不重复元素数

FM+算法分析：$X = \frac{1}{Z} - 1$，$Z = \frac{1}{k}\sum_{j=1}^{k} z_j$，$\mathbf{E}[Z] = \mathbf{E}[z]$，$\mathrm{var}[Z] = \frac{\mathrm{var}[z]}{k}$

$$\Pr[|Z - \frac{1}{D+1}| > c\frac{1}{D+1}] < \frac{(D+1)^2}{c^2}\mathrm{var}[Z]$$

$$c = \frac{\epsilon D}{D+1+\epsilon D} \Rightarrow \Pr[|X - D| > \epsilon D] < \frac{2(D+1+\epsilon D)^2}{k\epsilon^2 D^2} < \frac{2}{k}(\frac{2}{\epsilon} + 1)^2$$

这里，ϵ为精度要求，假设概率要求为$1-\delta$，只需

$$\frac{2}{k}(\frac{2}{\epsilon} + 1)^2 < \delta$$

$$\Rightarrow k > \frac{2}{\delta}(\frac{2}{\epsilon} + 1)^2 = O(\frac{1}{\epsilon^2 \delta})$$

问题2：不重复元素数

利用Median技术

FM++ Algorithm

1. 运行具有常数概率的FM+算法$m = \Theta(\log\frac{1}{\delta})$次；
2. $\hat{D}$为所有 m 个结果的中间值；
3. **return** $\hat{D}$.

FM++是一个以$1-\delta$概率保证$(1 \pm \epsilon)$近似的算法

整体代价为$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$

问题2：不重复元素数

利用多次运行，另一种做法

FM′+ (Bottom-k) Algorithm

*/** Maintain a variable z **/*

1. 随机选取一个哈希函数$h : [m] \mapsto [0, 1]$
2. $(z_1, z_2, \ldots, z_k) = 1$.
3. 维护当前看到的最小的k个哈希值，满足$z_1 \le z_2 \le \cdots \le z_k$
4. **return** k/z_k.

FM′+是一个$(1 \pm \epsilon)$近似算法

$k = O(\frac{1}{\epsilon^2}) \Rightarrow$ FM′+以常数概率成功

问题2：不重复元素数

FM′+算法的分析

$$\Pr[|\frac{k}{z_k} - D| > \epsilon D] < ?$$

- $|\frac{k}{z_k} - D| > \epsilon D \Leftrightarrow z_k < \frac{k}{(1+\epsilon)D}$ 或者 $z_k > \frac{k}{(1-\epsilon)D}$
- 令随机变量$X_i = 1 \Leftrightarrow h(i) < \frac{k}{(1+\epsilon)D}$
  - $z_k < \frac{k}{(1+\epsilon)D} \quad \equiv \quad \sum_{i=1}^m X_i > k$
- 令随机变量$Y_i = 1 \Leftrightarrow h(i) < \frac{k}{(1-\epsilon)D}$
  - $z_k > \frac{k}{(1-\epsilon)D} \quad \equiv \quad \sum_{i=1}^m Y_i < k$
- $\mathbf{E}[\sum X_i] = \frac{k}{(1+\epsilon)} \Leftarrow m = D$
- $\mathrm{var}[\sum X_i] = D \cdot \mathrm{var}[X_i] \le D \cdot \mathbf{E}[X_i^2] = D \cdot \mathbf{E}[X_i] = \frac{k}{1+\epsilon}$
- $\mathbf{E}[\sum Y_i] = \frac{k}{(1-\epsilon)}$
- $\mathrm{var}[\sum Y_i] = D \cdot \mathrm{var}[Y_i] \le D \cdot \mathbf{E}[Y_i^2] = D \cdot \mathbf{E}[Y_i] = \frac{k}{1-\epsilon}$

问题2：不重复元素数

FM′+算法的分析

$$\begin{aligned}\Pr[\textstyle\sum X_i > k] &= \Pr[\textstyle\sum X_i - \frac{k}{1+\epsilon} > k - \frac{k}{1+\epsilon}] \\ &= \Pr[\textstyle\sum X_i - \frac{k}{1+\epsilon} > \frac{\epsilon k}{1+\epsilon}] \\ &\le \Pr[|\textstyle\sum X_i - \frac{k}{1+\epsilon}| > \frac{\epsilon k}{1+\epsilon}] \\ &\le \frac{(1+\epsilon)^2 \mathrm{var}[\sum X_i]}{\epsilon^2 k^2} \le \frac{1+\epsilon}{\epsilon^2 k} \le \frac{c}{2}\end{aligned}$$

$\Rightarrow k \ge \frac{2(1+\epsilon)}{c\epsilon^2}$

$$\begin{aligned}\Pr[\textstyle\sum Y_i < k] &= \Pr[\textstyle\sum Y_i - \frac{k}{1-\epsilon} < k - \frac{k}{1-\epsilon}] \\ &= \Pr[\textstyle\sum Y_i - \frac{k}{1-\epsilon} < -\frac{\epsilon k}{1-\epsilon}] \\ &\le \Pr[|\textstyle\sum Y_i - \frac{k}{1-\epsilon}| > \frac{\epsilon k}{1-\epsilon}] \\ &\le \frac{(1-\epsilon)^2 \mathrm{var}[\sum Y_i]}{\epsilon^2 k^2} \le \frac{1-\epsilon}{\epsilon^2 k} \le \frac{c}{2}\end{aligned}$$

$\Rightarrow k \ge \frac{2(1-\epsilon)}{c\epsilon^2}$

$$k \ge \max\{\frac{2(1-\epsilon)}{c\epsilon^2}, \frac{2(1+\epsilon)}{c\epsilon^2}\} = O(\frac{1}{\epsilon^2})$$

问题2：不重复元素数

事实上，我们无法存储实数

- 设计一个$O(\log m)$位空间代价的$O(1)$近似算法
- $\frac{1}{C} \times D \le \hat{D} \le C \times D$

Practical FM Algorithm

1. 从2-wise independent哈希函数族中随机选取 $h : [m] \mapsto [m]$;
2. $z = 0$;
3. **if** $\mathrm{zeros}(h(j)) > z$ **then** $z = \mathrm{zeros}(h(j))$;

$\mathrm{zeros}(p) = \max\{i \,|\, p\%2^i = 0\}$ $\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{0}$ $\Rightarrow$ zeros(·)=2

4. **return** $\hat{D} = 2^{z+1/2}$;

问题2：不重复元素数

定义1.16[k-wise independent hash functions]

一个从 $[a]$ 到 $[b]$ 的哈希函数族 $\mathcal{H}$ 被称作是 k-wise independent 的，如果 $\forall j_1, \ldots, j_k \in [b]$ 和 $\forall\, i_1, \ldots, i_k \in [a]$，其中$i_x \ne i_y$，有

$$\Pr_{h \in \mathcal{H}}\left(h(i_1) = j_1 \wedge \ldots \wedge h(i_k) = j_k\right) = 1/b^k$$

- 理论上，我们可以利用$\log_2 |\mathcal{H}|$位存储某个$h \in \mathcal{H}$

例1.2[k-wise independent例子]

- 所有形如$[a] \to [b]$的函数构成的$\mathcal{H}$，$|\mathcal{H}| = b^a$;
- $\mathcal{H}_{poly}$：至多为$k-1$阶多项式，系数在$[n]$中，要求$n = p^r$，p为素数，$|\mathcal{H}_{poly}| = n^k$，存储需$O(k \log n)$位

- 存储2-wise的$\mathcal{H}_{poly}$某个函数，需要$O(2 \log n)$位

问题2：不重复元素数

存储哈希函数需要$O(\log m)$位；存储z需要$\log\log m$位

定理1.22

Practical FM算法以$1 - \frac{2\sqrt{2}}{C}$的概率满足$D/C \le \hat{D} \le C \times D$

- 对$j \in [m]$和$r \ge 0$，令$X_{r,j} = 1 \Leftrightarrow \mathrm{zeros}(h(j)) \ge r$
- $Y_r = \sum_{j \in [m]} X_{r,j} = \sum_{j: f_j > 0} X_{r,j}$
  - Y_r：哈希函数值的二进制形式尾部 0 数目 $\ge r$ 的元素数
- 假设算法结束时，$z = t$
  - $t \ge r \Leftrightarrow$ 存在 j 使得 $\mathrm{zeros}(h(j)) \ge r \Leftrightarrow X_{r,j} = 1 \Leftrightarrow Y_r > 0$
  - $t \le r - 1 \Leftrightarrow \forall j$ 有 $\mathrm{zeros}(h(j)) \le r - 1 \Leftrightarrow X_{r,j} = 0 \Leftrightarrow Y_r = 0$

$$\mathbf{E}[X_{r,j}] = \Pr\left(\mathrm{zeros}(h(j)) \ge r\right) = \frac{1}{2^r}$$

$$\mathbf{E}[Y_r] = \sum_{j: f_j > 0} \mathbf{E}[X_{r,j}] = \frac{D}{2^r}$$

问题2：不重复元素数

$$\begin{aligned}\mathrm{var}[Y_r] &= \mathrm{var}[\textstyle\sum X_{r,j}] \\ &= \mathbf{E}[(\textstyle\sum X_{r,j})^2] - \mathbf{E}[\textstyle\sum X_{r,j}]^2 \\ &= \textstyle\sum_{i,j} \mathbf{E}[X_{r,i} \cdot X_{r,j}] - \sum_{i,j} (\mathbf{E}[X_{r,i}] \cdot \mathbf{E}[X_{r,j}])\end{aligned}$$

2-wise-indep. $\equiv \quad i \ne j \Rightarrow \mathbf{E}[X_{r,i} \cdot X_{r,j}] = \mathbf{E}[X_{r,i}] \cdot \mathbf{E}[X_{r,j}]$

$$\begin{aligned}&= \textstyle\sum \mathbf{E}[X_{r,j}^2] - \sum \mathbf{E}[X_{r,j}]^2 \\ &\le \textstyle\sum \mathbf{E}[X_{r,j}^2] = \sum \mathbf{E}[X_{r,j}] = \frac{D}{2^r}\end{aligned}$$

由Markov不等式有

$$\Pr[Y_r > 0] = \Pr[Y_r \ge 1] \le \frac{\mathbf{E}[Y_r]}{1} = \frac{D}{2^r}$$

由切比雪夫不等式有

$$\Pr[Y_r = 0] \le \Pr[|Y_r - \mathbf{E}[Y_r]| \ge \frac{D}{2^r}] \le \frac{\mathrm{var}[Y_r]}{(D/2^r)^2} \le \frac{2^r}{D}$$

问题2：不重复元素数

再一次，利用Median技术

- 运行有常数概率Practical FM算法$t = \Theta(\log \frac{1}{\delta})$次
- $\hat{D}$为所有t个结果的中间值
- 上述算法以$1 - \delta$概率保证$O(1)$近似的算法
- 整体代价为$O(\log m \log \frac{1}{\delta})$

问题2：不重复元素数

▷ $O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$空间代价的算法
▷ 主要思想：利用两层hash

BJKST Algorithm

1. 随机选2-wise independent哈希函数$h:[m]\to[m]$;
2. 随机选2-wise independent哈希函数$g:[m]\to[b\epsilon^{-4}\log^2 m]$;
3. $z=0$, $B=\emptyset$;
4. **if** zeros$(h(j))\ge z$ **then**
5. $\quad B = B\cup\{(g(j),\text{zeros}(h(j)))\}$;
6. $\quad$**if** $|B|>\frac{c}{\epsilon^2}$ **then**
7. $\quad\quad z=z+1$;
8. $\quad\quad$从B中删除(α,β)，其中$\beta<z$;
9. **return** $\hat{D}=|B|2^z$;

问题2：不重复元素数

定理1.23[BJKST算法性能]
BJKST算法使用$O(\log m + \frac{1}{\epsilon^2}(\log\frac{1}{\epsilon} + \log\log m))$空间，可以实现至少2/3概率保证$(1\pm\epsilon)$近似。

▷ 存储 h 需$O(\log m)$
▷ 存储 $|B|$ 个 g 取值需 $O(\epsilon^{-2}\log(b\epsilon^{-4}\log^2 m))$

问题3：点查询

流模型（The Streaming Model）

A stream:

$$\sigma = \langle a_1, a_2, \ldots, a_n\rangle, a_i\in[m],$$

then we can define a frequency vector

$$\mathbf{f} = (f_1,\ldots,f_m), \text{ s.t. } \sum_{1\le i\le n} f_i = n.$$

定义1.18[点查询（point query）问题]
给定数据流 σ 和事先未知查询 a_i，输出 f_i。

问题3：点查询

ℓ_p范数：$||x||_p = (\sum_i |x_i|^p)^{1/p}$

定义1.19[ℓ_p-近似点查询（频度估计）]
给定数据流 σ 以及事先未知查询 a_i 输出 $\hat{f}_i$ 满足 $\hat{f}_i\in f_i\pm\epsilon||\mathbf{f}||_p$。

▷ $||x||_1\ge||x||_2\ge\cdots\ge||x||_\infty$，$p$越大，估计越准确
▷ $||x||_0$是不同元素的数目
▷ $||x||_1$是流的长度
▷ $||x||_\infty$是最大频度

问题3：点查询–ℓ_1近似

Misra-Gries 算法[1982]

/** *Maintain a set A consists of pairs* $(i,\hat{f}_i)$ **/
1. $A\leftarrow\emptyset$;
2. 对每一个数据流中的元素e,
   (a) 如果 $e\in A$, 令 $(e,\hat{f}_e)\leftarrow(e,\hat{f}_e+1)$
   (b) 否则，如果 $|A|<1/\epsilon$, 将 $(e,1)$ 插入 A
   (c) 否则，将所有 A 中计数减1
   $((j,\hat{f}_j)\leftarrow(j,\hat{f}_j-1))$,
   如果 $\hat{f}_j=0$, 从 A 中删除 $(j,0)$
3. 对于查询 i, 如果 $i\in A$, **return** $\hat{f}_i$, 否则 **return** 0;

问题3：点查询–ℓ_1近似

$||x||_1 = n$

定理1.24
Misra-Gries算法空间代价为$O(\epsilon^{-1}\log mn)$，对任意查询 i，返回 $\hat{f}_i$ 满足

$$f_i-\epsilon n\le\hat{f}_i\le f_i$$

▷ 如果不发生减1的情况，那么$\hat{f}_i=f_i$
▷ 如果发生了减1的情况，有$\hat{f}_i<f_i$
▷ 假设共发生了c次减1的情况，总数减少$\frac{c}{\epsilon}\le n$
▷ 每个计数至多减少c，$\hat{f}_i\ge f_i-c\ge f_i-\epsilon n$

问题3：点查询–ℓ_1近似

Metwally 算法[2005]

/** *Maintain a set A consists of pairs* $(i,\hat{f}_i)$ **/
1. $A\leftarrow\emptyset$;
2. 对每一个数据流中的元素e,
   (a) 如果 $e\in A$, 令 $(e,\hat{f}_e)\leftarrow(e,\hat{f}_e+1)$
   (b) 否则，如果$|A|<1/\epsilon$，将 $(e,1)$ 插入 A
   否则，将 $(e, MIN+1)$ 插入 A，并删除一个满足$\hat{f}_e=MIN$的元素
   /**这里，$MIN=\min\{\hat{f}_e\}$**/
3. 查询 i, 如果 $i\in A$, **return** $\hat{f}_i$, 否则 **return** MIN;

问题3：点查询–ℓ_1近似

定理1.25
Metwally算法空间代价为$O(\epsilon^{-1}\log mn)$，对任意查询 i，返回 $\hat{f}_i$ 满足

$$f_i\le\hat{f}_i\le f_i+\epsilon n$$

▷ 如果不发生删除的情况，那么$\hat{f}_i=f_i$
▷ 如果删除，计数一定不大于删除后的MIN，有$\hat{f}_i\ge f_i$
▷ A中元素和总是n, $MIN\frac{1}{\epsilon}\le n\Rightarrow MIN\le\epsilon n$
▷ 每个元素至多超出真实值MIN, $\hat{f}_i\le f_i+\epsilon n$

问题3：点查询–ℓ_1近似

The (turnstile) Streaming Model

A stream including deletion:

$$\sigma = \langle \pm a_1, \pm a_2, \ldots, \pm a_n \rangle, a_i \in [m],$$

then we can define a frequency vector

$$\mathbf{f} = (f_1, \ldots, f_m).$$

- 初始化向量 $\mathbf{f} \in \mathbb{R}^m$ 为0
- 每次更新 (i, Δ)，使得 $f_i \leftarrow f_i + \Delta$，$f_i$可以比0小
- $\Delta = 1$ 即是之前所用的流模型（Vanilla）
- $\Delta > 0$：流模型（Cash Register）

问题3：点查询–ℓ_1近似

之前讲述方法有局限性

- 不支持删除操作
  令$\Delta = \pm 1$即可定义插入、删除操作
- Misra-Gries和Metwally算法不支持数据流连接'$\circ$'
  无法简单地根据σ_1和σ_2的结果计算$\sigma_1 \circ \sigma_2$的结果

定义1.20[Sketches]

定义在流数据σ上的数据结构$DS(\sigma)$是一个Sketch，如果存在一个Space-Efficient的合并算法COMB使得

$$COMB(DS(\sigma_1), DS(\sigma_2)) = DS(\sigma_1 \circ \sigma_2).$$

定义1.21[Linear Sketches]

定义在$[n]$上的流数据σ上的sketching算法输出sk(σ)，如果sk(σ)取值为维度$\ell = \ell(n)$的向量，并且是 $\mathbf{f}(\sigma)$ 的线性函数，那么sk(σ)是一个linear sketch，ℓ是sketch的维度。

问题3：点查询–ℓ_1近似

Count-Min Sketch 算法[2005]

1. $k = \frac{4}{\epsilon}$;
2. $C[1\ldots k] \leftarrow 0$;
3. 随机选择一个2-wise独立哈希函数$h : [m] \to [k]$;
4. **for** 每一次更新 (j, c) **do**
5. $\quad C[h(j)] = C[h(j)] + c$;
6. **return** $\hat{f}_a = C[h(a)]$ 对查询元素 a;

- 空间代价$O(\frac{1}{\epsilon} \cdot \log n)$
- 针对Cash Register模型，有近似质量保证
- 正确性：$\hat{f}_a \geq f_a$并且$\Pr[\hat{f}_a \geq f_a + \epsilon ||\mathbf{f}_{-a}||_1] \leq \frac{1}{4}$

问题3：点查询–ℓ_1近似

Count-Min+ Sketch 算法[2005]

1. $C[1\ldots t][1\ldots k] \leftarrow \mathbf{0}$, $k = \frac{4}{\epsilon}$ and $t = \lceil \log \frac{1}{\delta} \rceil$;
2. 随机选择 t 个2-wise独立哈希函数$h_i : [m] \to [k]$;
3. **for** 每一次更新 (j, c) **do**
4. $\quad$ **for** $i = 1$ to t **do**
5. $\quad\quad C[i][h_i(j)] = C[i][h_i(j)] + c$;
6. **return** $\hat{f}_a = \min_{1 \leq i \leq t} C[i][h_i(a)]$ 对查询元素 a;

- 空间代价$O(\frac{1}{\epsilon} \log \frac{1}{\delta} \cdot \log n)$
- 针对Cash Register模型，有近似质量保证

问题3：点查询–ℓ_1近似

定理1.26[Count-Min+]

Count-Min+ 算法以 $1 - \delta$ 概率给出ℓ_1近似–点查询的$(1 + \epsilon)$近似，具体地

$$f_a \leq \hat{f}_a \leq f_a + \epsilon ||\mathbf{f}_{-a}||_1$$

这里，$||\mathbf{f}_{-a}||_1 = ||\mathbf{f}||_1 - f_a$

问题3：点查询–ℓ_1近似

- 考虑更一般的Turnstile模型，即(j, c)中c可以为负数
- 使用Count-Min算法，$\Pr[\hat{f}_a \in f_a \pm \epsilon ||\mathbf{f}_{-a}||_1] \leq \frac{1}{4}$
- 利用median技术，可以将概率改进至$1 - \delta$

Count-Min++ Sketch 算法[2005]

1. $C[1\ldots t][1\ldots k] \leftarrow \mathbf{0}$, $k = \frac{4}{\epsilon}$ and $t = \lceil \log \frac{1}{\delta} \rceil$;
2. 随机选择t个2-wise独立哈希函数$h_i : [m] \to [k]$;
3. **for** each update (j, c) **do**
4. $\quad$ **for** $i = 1$ to t **do**
5. $\quad\quad C[i][h_i(j)] = C[i][h_i(j)] + c$;
6. **return** $\hat{f}_a = median_{1 \leq i \leq t} C[i][h_i(a)]$ 对查询 a;

- 空间代价$O(\epsilon^{-1} \log \delta^{-1} \cdot \log n)$

问题3：点查询–ℓ_1近似

定理1.27[Count-Min++]

Count-Min++ 算法以 $1 - \delta$ 概率给出ℓ_1近似–点查询的$(1 \pm \epsilon)$近似，具体

$$|\hat{f}_a - f_a| \leq \epsilon ||\mathbf{f}_{-a}||_1$$

支持删除操作（Turnstile模型）

问题3：点查询–ℓ_2近似

质量更好的估计结果

CountSketch 算法[2004]

1. $C[1\ldots k] \leftarrow \mathbf{0}$, $k = \frac{3}{\epsilon^2}$;
2. 随机选择1个2-wise独立哈希函数$h : [m] \to [k]$;
3. 随机选择1个2-wise独立哈希函数$g : [m] \to \{-1, 1\}$;
4. **for** each update (j, c) **do**
5. $\quad C[h(j)] = C[h(j)] + c \cdot g(j)$;
6. **return** $\hat{f}_a = g(a) \cdot C[h(a)]$ for query a;

- 空间代价$O(\frac{1}{\epsilon^2} \cdot \log n)$
- 针对Turnstile模型，以常数概率，有$(1 \pm \epsilon)$近似比

问题3：点查询–ℓ_2近似

利用Median技术改进估计质量

CountSketch+ 算法

1.$C[1\dots t][1\dots k] \leftarrow \mathbf{0}$, $k = \frac{3}{\epsilon^2}$, $t = O(\log\frac{1}{\delta})$;
2.随机选择t个2-wise独立哈希函数$h_i : [m] \to [k]$;
3.随机选择t个2-wise独立哈希函数$g_i : [m] \to \{-1, 1\}$;
4.**for** each update (j, c) **do**
5. **for** each $i = 1$ to t **do**
6. $C[i][h_i(j)] = C[i][h_i(j)] + c \cdot g_i(j)$;
7.**return** $\hat{f}_a = median_{1\le i\le t} g_i(a) C[i][h_i(a)]$ for query a;

空间代价$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot\log n)$

问题3：点查询–ℓ_2近似

$t = \log\frac{1}{\delta}$使得切尔诺夫不等式

定理1.28[CountSketch+]

CountSketch+ 算法至少以 $1-\delta$ 概率给出ℓ_2近似–点查询的$(1\pm\epsilon)$近似估计，具体地

$$|\hat{f}_a - f_a| \le \epsilon||\mathbf{f}_{-a}||_2$$

问题3：点查询

点查询（频度估计）问题的算法

Method	$\hat{f}_a - f_a \in$	Space	Pr	Mod.
Misra-Gries	$[-\epsilon\|\|\mathbf{f}_{-a}\|\|_1, 0]$	$O(\frac{\log mn}{\epsilon}) = O(\frac{\log n}{\epsilon})$	0	+
Metwally	$[0, \epsilon\|\|\mathbf{f}_{-a}\|\|_1]$	$O(\frac{\log mn}{\epsilon}) = O(\frac{\log n}{\epsilon})$	0	+
CountSketch+	$\pm\epsilon\|\|\mathbf{f}_{-a}\|\|_2$	$O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log n)$	δ	$\pm$
Count-Min+	$[0, \epsilon\|\|\mathbf{f}_{-a}\|\|_1]$	$O(\frac{1}{\epsilon}\log\frac{1}{\delta}\log n)$	δ	+
Count-Min++	$\pm\epsilon\|\|\mathbf{f}_{-a}\|\|_1$	$O(\frac{1}{\epsilon}\log\frac{1}{\delta}\log n)$	δ	$\pm$

- $m < n$
- $+$表示Cash Register Model，$\pm$表示Turnstile Model

问题4：频度矩估计

假设我们现在处理Vanilla Model

定义1.22[频度矩--Frequency Moments]

给定数据流 $\sigma = \langle a_1, \dots, a_n\rangle$，假设$a_i \in [m]$，令$\mathbf{f} = \mathbf{f}(\sigma) = (f_1, f_2, \dots, f_m)$，显然，$\sum f_i = n$。数据流 σ 的第 k 阶频度矩（kth frequency moment）表示为 $F_k(\sigma)$ 或者 F_k，

$$F_k = \sum_{i=1}^{m} f_j^k = ||\mathbf{f}||_k^k$$

事实上，这里k可以扩展为实数$k \ge 0$，很多方法可以扩展到Cash Register Model

问题4：频度矩估计

- F_0是不重复元素的数目
- F_1是计数问题
- F_2可表示关系数据自连接（self join）操作结果大小
- 这部分给出AMS算法，为$F_k(k \ge 2)$估计问题提供亚线性空间的算法

问题4：频度矩估计

Basic AMS 算法

1. $(L, r, a) \leftarrow (0, 0, 0)$;
2. **for** 第 j 个元素 a_j **do**
3. $L \leftarrow L + 1$;
4. $\beta \leftarrow$ random bit with $\Pr[\beta = 1] = \frac{1}{L}$;
5.   **if** $\beta = 1$ **then**
6. $a \leftarrow a_j$;
7. $r \leftarrow 0$;
8.   **if** $a_j == a$ **then**
9. $r \leftarrow r + 1$;
10.**return** $X = L(r^k - (r-1)^k)$;

问题4：频度矩估计

空间代价

- 存储 L 和 r 需要 $\log n$ 位
- 存储 a 需要 $\log m$ 位
- 共计 $O(\log m + \log n)$ 位代价

直观思想

- 随机选取一个位置 $y \in [n]$
- $a = a_y$
- $r = |\{j | j \ge y$并且$a_j = a_y\}|$

$$\Pr[|X - \mathbf{E}[X]| \ge \epsilon\mathbf{E}[X]] \le \frac{kn^{1-\frac{1}{k}}F_k^2}{\epsilon^2 F_k^2} = \frac{kn^{1-\frac{1}{k}}}{\epsilon^2}$$

问题4：频度矩估计

- Basic AMS算法的方差太大，无法直接利用Median技术将获得$(1+\epsilon)$近似的失败概率降低到满足要求
- 首先需要利用averaging技术，将方差降低

更一般化的表达

定理1.29[Median-of-Means]

存在常数 c 满足：令 X 是一个实数量 Q 的无偏估计，令 $\{X_{ij}\}_{i\in[t], j\in[k]}$ 是与 X 同分布的独立变量集合，

$$t = c\log\frac{1}{\delta}\text{并且}b = \frac{3\text{var}[X]}{\epsilon^2\mathbf{E}[X]^2}$$

令 $Z = median_{i\in[t]}(\frac{1}{b}\sum_{j=1}^{b} X_{ij})$，有 $\Pr[|Z - Q| \ge \epsilon Q] \le \delta$。

问题4：频度矩估计

Final AMS 算法

1. 利用Median-of-Mean技术调用Basic AMS算法;
2. 计算$t = c\log\frac{1}{\delta}$个平均值;
3. 每个平均值是$r = \frac{3k}{\epsilon^2}\cdot n^{1-\frac{1}{k}}$次调用的平均值;
4. **return** t个数值的中间值;

- ▷ $r = \frac{3var[X]}{\epsilon^2\mathbf{E}[X]^2} \le \frac{3kn^{1-\frac{1}{k}}F_k^2}{\epsilon^2F_k^2} = \frac{3k}{\epsilon^2}\cdot n^{1-\frac{1}{k}}$
- ▷ 空间代价：
  $tr\cdot O(\log m+\log n) = O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\cdot kn^{1-\frac{1}{k}}(\log m+\log n))$
- ▷ 忽略$\log m, \log n, \log\frac{1}{\delta}$,对常数$k$，有$\tilde{O}(\frac{1}{\epsilon^2}n^{1-\frac{1}{k}})$
- ▷ 更优的：$\tilde{O}(\frac{1}{\epsilon^2}n^{1-\frac{2}{k}})$，下界$\tilde{\Omega}(n^{1-\frac{2}{k}})$和$\tilde{\Omega}(\epsilon^{-2})$

问题4：频度矩估计

上述算法在$k=2$时，代价为$\tilde{O}(\epsilon^{-2}n^{0.5})$，过大

Basic F_2 AMS 算法(Tug-of-War Sketch)

1. 随机选择4-wise独立的哈希函数$h:[m]\to\{-1,1\}$;
2. $x \leftarrow 0$;
3. **for** each update (j, c) **do**
4. $\quad x \leftarrow x + c\cdot h(j)$;
5. **return** x^2;

- ▷ 空间代价：$O(\log n)$
- ▷ $\mathbf{E}[X^2] = F_2$
- ▷ $\mathrm{var}[X^2] \le 2F_2^2$

问题4：频度矩估计

- ▷ 利用media-of-mean技术，可获得(ϵ,δ)近似算法
- ▷ 令$t = c\log\frac{1}{\delta}$
- ▷ 令$k = \frac{3\mathrm{var}[X^2]}{\epsilon^2\mathbf{E}[X^2]^2} \le \frac{6}{\epsilon^2}$
- ▷ 空间代价$\tilde{O}(\epsilon^{-2}\log\frac{1}{\delta}) = O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log n)$

问题5：频繁元素

定义1.23[频繁元素，Frequent Items，heavy hitters]

给定数据流 σ 和参数 k，输出频繁元素集合 $\{j \mid f_j > n/k\}$.

利用点查询的算法求解

- ▷ 令 $\epsilon = 1/k$
- ▷ 执行Misra-Gries 算法一次，获得A
- ▷ 输出A中元素

如果 $f_j > n/k$，那么 $\tilde{f}_j > 0$，$j \in A$

问题5：频繁元素

- ▷ 想估计最大频度，是否可以在亚线性代价内实现？
- ▷ 考虑一个hard情况，每个元素频度都差不多，最后一个元素决定最大频度是多少
- ▷ 放松要求：最频繁的元素是heavy的，设计算法

定义1.24[Heavy Hitters]

给定参数 $\phi \in (0,1)$ 和 $p \in [1,\infty)$，对于数据流 σ，假设对应的频度向量为 $\mathbf{f}$，heavy hitters定义如下

$$HH_\phi^p(\mathbf{f}) = \{i \in [m] : f_i \ge \phi||\mathbf{f}||_p\}$$

我们关注 $p = 1$ 的情况

定义1.25[ϕ-heavy]

元素 i 是 ϕ-heavy的，如果 $f_i \ge \phi\sum f_j$

问题5：频繁元素

定义1.26[Approximate Heavy Hitters for $p=1$]

给定参数 $\phi, \epsilon \in (0,1)$，对于数据流 σ，假设对应的频度向量为 $\mathbf{f}$，计算一个集合 $S \subseteq [m]$ 使得

$$HH_\phi \subseteq S \subseteq HH_{(1-\epsilon)\phi}$$

利用 ℓ_1 点查询算法求解，设计(ϵ', δ')近似算法

- ▷ 利用Count-Min++算法实现(ϵ,δ)近似点查询算法
- ▷ 令$\epsilon = \epsilon'\phi/2$，$\delta = \delta'/m$，为每一个元素 $i \in [m]$ 估计 $\hat{f}_i$
- ▷ 返回集合 $S = \{i \in [m] : \hat{f}_i \ge (\phi - \epsilon'\phi/2)||\mathbf{f}||_1\}$

插入更新 $||\mathbf{f}||_1$ 非常容易获得，有删除就很难

- ▷ 算法的空间代价：$O(\frac{1}{\phi}\frac{1}{\epsilon}(\log\frac{1}{\delta'} + \log m)\log n)$

问题6：固定大小采样

- ▷ 问题描述：时刻维护一个大小为 s 的采样，使得 σ 中的每个元素出现的概率均是 $\frac{s}{n}$

水库抽样算法

1. $n \leftarrow 0$;
2. 用流数据的前 s 个元素初始化 $A[1,\ldots,s]$，$n \leftarrow s$;
3. **for** 每一个元素 x **do**
4. $\quad x$ 以 $\frac{s}{n+1}$ 概率进入 A;
5. $\quad$**if** x 加入 A **then**
6. $\quad\quad$任意选择 A 中元素剔除;
7. $\quad n \leftarrow n+1$;

问题7：Bloom Filter

定义1.27[Membership Problem]

令 U 是整数集合 $\{1,\ldots,|U|\}$，S 是 U 的一个大小为 n 的子集合，预处理 S，给定整数 $q \in U$，判定是否 $q \in S$。

- ▷ U 中整数可以用 $w = \log|U|$ 位表示
- ▷ 哈希表，查询期望时间是 $O(1)$，总代价 $O(nw)$位
- ▷ 是否可以在 $o(nw)$ 代价解决？
  如果要求精确答案，那么至少用 $\log\binom{|U|}{n}$位，当 $|U| \gg n$，即 $\Omega(nw)$
- ▷ 放松要求
  允许false positives，不允许false negatives

问题7：Bloom Filter

定义1.28[Approximate Membership Problem]

给定整数 $q \in U$，

- $q \in S \Rightarrow$ 输出'yes'
- $q \notin S \Rightarrow$ 以至少 $1-\delta$ 概率输出'no'

问题7：Bloom Filter

近似哈希的方法

1. 令$\mathcal{H}$是一族通用哈希函数：$|U| \to [m]$，$m = \frac{n}{\delta}$;
2. 随机选取 $h \in \mathcal{H}$，并维护数组$A[m]$;
3. **for** $i \in S$ **do**
4. $\quad A[h(i)] = 1$;
5. 给定查询 q，**return** 'yes' 当且仅当$A[h(i)] = 1$;

- $q \in S \Rightarrow$输出'yes'
- $q \notin S \Rightarrow$

$$\sum_{j \in S} \Pr[h(q) = h(j)] \le \frac{n}{m} = \delta$$

问题7：Bloom Filter

定义1.29[Ideal Hash Function]

哈希函数 $h: \mathcal{U} \to [m]$是理想的（ideal），如果对每一个 $k \in \mathcal{U}$，$h(k)$ 均匀独立地从$[1, m]$间取值。

Bloom Filter方法

1. 令$\mathcal{H}$是一族独立的理想哈希函数：$|U| \to [m]$;
2. 随机选取 $h_1 \dots h_d \in \mathcal{H}$，并维护数组 $A[m]$;
3. **for** $i \in S$ **do**
4. $\quad$ **for** $j \in [1, d]$ **do**
5. $\quad\quad A[h_j(i)] = 1$;
6. 给定查询 q，
   return 'yes' 当且仅当$\forall j \in [d]$, $A[h_j(q)] = 1$;

代价：$m = O(n \log \frac{1}{\delta}) \Rightarrow O(n \log \frac{1}{\delta})$

- 概率基础
- 亚线性空间算法
- **亚线性时间算法**
- 并行模型算法

亚线性时间算法

高效算法的标准是什么？

- 多项式时间算法
- 线性时间算法
  - 输入大小为 n，算法的时间代价是 cn

如果时间资源受限，甚至不允许完整读取数据一遍，我们能做什么？

- 无法回答"for all"或者"exactly"类似的问题：数组中是否所有元素均比10大？数组中比10大的有多少？
- 或许能够回答下列问题：数组中比10大的元素大概有多少？数组中的平均数是否以很大概率大于10？

算法的计算目标需要修改

亚线性时间算法

算法的计算目标需要修改

- 对于绝大多数问题，我们只能给出近似的答案

什么是"近似"

- 经典的近似算法概念：
  - 算法的结果有一个度量函数：路径长度
  - 算法结果度量值与最优解度量值"接近"：近似比
- Property Testing（判定问题）
  - 算法的结果是yes或者no
  - 算法输出要么与输入 I 对应的答案相同，要么与 I' 对应的答案相同，这里 I' 与 I 非常接近

问题1：点集合的直径

点集合的直径问题

输入：m个点，距离用矩阵 D表示

(1) D_{ij}是i点到j点的距离

(2) D是对称的，并且满足三角不等式

$D_{ij} \le D_{ij} + D_{kj}$

输出：(i, j)使得D_{ij}是最大的，D_{ij}是点集合的直径

问题1：点集合的直径

亚线性求解直径算法（Indyk's Algorithm）

/** *input D* **/

1. 选取任意一个 k;
2. 选取 l 最大化 D_{kl};
3. **return** (k, l)

近似比分析：$opt/2 \le D_{kl} \le opt$

$$\begin{aligned} opt = D_{ij} &\le D_{ik} + D_{kj} && \text{(三角不等式)} \\ &\le D_{kl} + D_{kl} && \text{(l 是最大化选择)} \\ &\le 2D_{kl} \end{aligned}$$

运行时间：$O(m) = O(\sqrt{n})$

问题2：连通分量的数目

连通分量的数目（#CC）

输入：$G=(V,E),\epsilon,d=deg(G)$
图G用邻接链表表示
$|V|=n,|E|=m\leq dn$
输出：y, 令C为#CC
$C-\epsilon n\leq y\leq C+\epsilon n$ （additive appro.）

#CC可以在线性时间求解（DFS或者BFS）

问题2：连通分量的数目

n_v：顶点v所属的连接分量中的节点数目
A：$A\subseteq V$是一个连通分量的点集合

$$\sum_{u\in A}\frac{1}{n_u}=\sum_{u\in A}\frac{1}{|A|}=1$$

$$C=\#cc=\sum_{u\in V}\frac{1}{n_u}$$

为什么用这种表示？

- ▷ 计算$\frac{1}{n_u}$需要$O(n)$时间？
- ▷ 需要对$O(n)$项求和？
- ▷ 可以estimate

问题2：连通分量的数目

估计 $C=\sum_{u\in V}\frac{1}{n_u}\Rightarrow$ 估计$\frac{1}{n_u}\Rightarrow$ 估计$\sum_{u\in V}\frac{1}{n_u}$

估计$\frac{1}{n_u}$

想法：n_u很大，精确计算很难，但此时$\frac{1}{n_u}$很小，可以用一个很小的常量代替$\frac{1}{n_u}$（0或者$\epsilon/2$）

$\hat{n_u}\ =\min\{n_u,\frac{2}{\epsilon}\}$
$\hat{C}\ =\sum_{u\in V}\frac{1}{\hat{n_u}}$

引理1.6

$\forall u\in V$，有$|\frac{1}{n_u}-\frac{1}{\hat{n_u}}|\leq\epsilon/2$，即$|C-\hat{C}|\leq\frac{\epsilon n}{2}$

问题2：连通分量的数目

估计 $C=\sum_{u\in V}\frac{1}{n_u}\Rightarrow$ 估计$\frac{1}{n_u}\Rightarrow$ 估计$\sum_{u\in V}\frac{1}{n_u}$
$\hat{n_u}=\min\{n_u,\frac{2}{\epsilon}\}$; $\hat{C}=\sum_{u\in V}\frac{1}{\hat{n_u}}$

计算$\hat{n_u}$算法

1.从 u 开始做先广遍历（BFS）;
2.**while** BFS 访问过的节点数量 $\leq\frac{2}{\epsilon}$ **do**
3.　　继续做 BFS 遍历;
4.　　**if** 没有新的节点可以遍历 **then**
5.　　　**return** BFS 访问过的节点数量;
6.**return** $2/\epsilon$;

运行时间：$O(d\cdot 1/\epsilon)$

问题2：连通分量的数目

估计 $C=\sum_{u\in V}\frac{1}{n_u}\Rightarrow$ 估计$\frac{1}{n_u}\Rightarrow$ 估计$\sum_{u\in V}\frac{1}{n_u}$
$\hat{n_u}=\min\{n_u,\frac{2}{\epsilon}\}$; $\hat{C}=\sum_{u\in V}\frac{1}{\hat{n_u}}$

亚线性连通分量数目求解算法（用 $\tilde{C}$ 估计 $\hat{C}$）

1.$r\leftarrow b/\epsilon^2$;
2.随机从 V 中选取 $U=\{u_1,\ldots,u_r\}$;
3.计算所有的 $\hat{n_{u_i}}$;
4.**return** $\tilde{C}=\frac{n}{r}\sum_{u\in U}\frac{1}{\hat{n_{u_i}}}$;

运行时间：$O(d\cdot 1/\epsilon\cdot 1/\epsilon^2)=O(d/\epsilon^3)=o(|G|)$
近似性能：$r=O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})\Rightarrow\Pr[|C-\hat{C}|\geq\epsilon n]\leq\delta$

问题3：近似最小支撑树

最小支撑树（Min Spanning Tree）

输入：$G=(V,E),\epsilon,d=deg(G)$
图G用邻接链表表示
边(u,v)的权重是$w_{uv}\in\{1,2,\ldots,w\}\cup\{\infty\}$
输出：$\hat{M}$, 令M为$\min_{T\,spans\,G}\{W(T)\}$
$(1-\epsilon)M\leq\hat{M}\leq(1+\epsilon)M$

MST问题可以在多项式时间求解，例如Kruskal算法

问题3：近似最小支撑树

我们需要重新利用分解的方式定义M
$G^{(i)}=(V,E^{(i)})$，这里$E^{(i)}=\{(u,v)|w_{uv}\leq i\}$
$C^{(i)}=\#\text{CC in }G^{(i)}$

让我考虑两个例子

- ▷ $w=1$: 只有权重为1的边，且连通
  $M=n-1$
- ▷ $w=2$: 有权重为1和2的边，且连通
  $M=n-1+C^{(1)}-1=n-2+C^{(1)}$

问题3：近似最小支撑树

定理1.31[MST]

$$M=n-w+\sum_{i=1}^{w-1}C^{(i)}$$

α_i：任一MST中权重为 i 的边的数目

$$\sum_{i>l}\alpha_i=C^{(l)}-1$$

$$\begin{aligned}M&=\sum_{i=1}^{w}i\cdot\alpha_i=\sum_{i=1}^{w}\alpha_i+\sum_{i=2}^{w}\alpha_i+\cdots+\sum_{i=w}^{w}\alpha_i\\&=C^{(0)}-1+C^{(1)}-1+\cdots+C^{(w-1)}-1\\&=n-1+C^{(1)}-1+\cdots+C^{(w-1)}-1=n-w+\sum_{i=1}^{w-1}C^{(i)}\end{aligned}$$

问题3：近似最小支撑树

利用连通分量数目估计的算法来估计MST大小

$\tilde{O}(d/\epsilon^3) \Rightarrow \mathbf{Pr}[|C-\tilde{C}| \le \epsilon n] \ge 1-\delta$

亚线性近似最小支撑树算法（计算$\hat{M}$）

1.**for** $i=1$ to $w-1$ **do**

2. $\hat{C^{(i)}}$ = appro. #CC of $G^{(i)}$ within $(\epsilon' = \frac{\epsilon}{2w}) \cdot n$;
   with probability $\ge 1-\delta' = 1-\frac{\delta}{w}$

3.**return** $\hat{M} = n - w + \sum_{i=1}^{w-1} \hat{C^{(i)}}$;

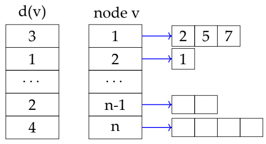
单次时间：$\tilde{O}(d \cdot 1/\epsilon'^3) = \tilde{O}(dw^3/\epsilon^3)$
共计时间：$\tilde{O}(dw^4/\epsilon^3) = o(|G|)$
近似性能：$\Pr[|\hat{M}-M| \ge \epsilon M] \le \delta$

问题4：计算图的平均度

图的平均度（Average Degree）

输入：$G=(V,E)$
图G用邻接链表表示
输出：平均度 $\bar{d} = \frac{\sum_{u\in V} d(u)}{n}$

假设：（1）G是简单图（无平行边、自环），$\Omega(n)$条边
（2）G由邻接链表和存储度数的数组表示



▷ degree query: $d(v)$
▷ neighbor query: $N(v,j)$
v的第j个邻居节点

问题4：计算图的平均度

利用Naive Sampling的方法
▷ 随机选取s个节点$\{v_1,\ldots,v_s\}$，返回$\frac{1}{s}\sum_{i=1}^{s} d(v_i)$

简单的下界分析
▷ 空图的平均度数为 $\bar{d}=0$，加一条边后变为 $\bar{d}=2/n$.
  ○ 为了区分上述两种情况，基本上需要访问所有的节点，即 $\Omega(n)$.
▷ 令 G_1 为大小为 n 的环: $\bar{d}=2$；
  令 G_2 由 $(n-c\sqrt{n})$-环和 $c\sqrt{n}$-团组成：$\bar{d} \approx 2+c^2$
  ○ 大致需要 $\frac{n}{c\sqrt{n}} = \frac{\sqrt{n}}{c}$ 步找到这个团，即 $\Omega(\sqrt{n})$

问题4：计算图的平均度

算法思想
▷ 将具有相似或者相同度数的节点分组
▷ 估计每个分组的平均度数，分组的方差都有界
  ○ 对于任意数值数组这个方法并不适用

分桶策略：$\beta = \frac{\epsilon}{c_\beta}$;

$t = \lceil \log_{(1+\beta)} n \rceil + 1 = O(\frac{\log n}{\log(1+\beta)}) \approx O(\frac{\log n}{\beta}) = O(\frac{\log n}{\epsilon})$

▷ $B_i = \{v | (1+\beta)^{i-1} < d(v) \le (1+\beta)^i\}, 0 \le i \le t-1$
▷ B_i 的总度数：$(1+\beta)^{i-1}|B_i| < d(B_i) \le (1+\beta)^i |B_i|$
▷ G 的总度数：$\sum_{i=0}^{t-1}(1+\beta)^{i-1}|B_i| < \sum d(u) \le \sum_{i=0}^{t-1}(1+\beta)^i|B_i|$
▷ $\frac{\sum_{i=0}^{t-1}(1+\beta)^{i-1}|B_i|}{n} < \bar{d} \le \frac{\sum_{i=0}^{t-1}(1+\beta)^i|B_i|}{n}$ 重点是估计$|B_i|/n$

问题4：计算图的平均度

第一个算法（naive sampling）

平均度估计算法I

1.从 V 中选取 S 个样本点;
2.$S_i \leftarrow S \cap B_i$; //利用 S_i 来估计 B_i
3.$\rho_i \leftarrow \frac{|S_i|}{|S|}$;
4.**return** $\hat{\bar{d}} = \sum_{i=0}^{t-1} \rho_i (1+\beta)^{i-1}$; //用下界，低估

▷ 令 $X_j^i = 1$ 表示样本 s_j 落到桶 B_i 中；否则，$X_j^i = 0$
▷ $\mathbf{E}[\rho_i] = \mathbf{E}[\frac{|S_i|}{|S|}] = \frac{\mathbf{E}[\sum_{j=1}^{|S|} X_j^i]}{|S|} = \frac{|S|\cdot|B_i|}{|S|\cdot n} = \frac{|B_i|}{n}$

问题4：计算图的平均度

算法I的问题：较小的B_i的估计效果会很差
考虑完全二分图$K_{3,n-3}$
▷ 有3个节点度为$n-3$，有$n-3$个节点度为3
▷ $B_a : (1+\beta)^{a-1} < 3 \le (1+\beta)^a$ $|B_a| = n-3$
▷ $B_b : (1+\beta)^{b-1} < n-3 \le (1+\beta)^b$ $|B_b| = 3$
▷ B_a 和 B_b 在 G 中连接的边数相同，均是$3(n-3)$
▷ $\bar{d} = \frac{3(n-3)+(n-3)3}{n} = 6 - \frac{18}{n} \approx 6$
▷ 算法I很可能未采样到B_b 否则需$\Omega(n)$期望时间
▷ $\hat{\bar{d}} = 3 \Rightarrow \frac{\bar{d}}{\hat{\bar{d}}}$ 至少为2（并没有分析其它worst cases）

问题4：计算图的平均度

改进算法（希望获得稳定的近似比为 2 的算法）
对于较小的桶置 0，假定已知一个 $\bar{d}$ 的下界 α

平均度估计算法II

1.从 V 中采样 S, $|S| = \Theta(\sqrt{n/\alpha} \cdot poly(\log n, 1/\epsilon))$;
2.$S_i \leftarrow S \cap B_i$;
3.**for** $i \in \{0,\cdots,t-1\}$ **do**
4. **if** $|S_i| \ge \theta_\rho |S| = \frac{1}{t}\sqrt{\frac{3}{8}\cdot\frac{\epsilon\alpha}{n}}|S|$ **then**
5. $\rho_i \leftarrow \frac{|S_i|}{|S|}$;
6. **else**
7. $\rho_i \leftarrow 0$;
8.**return** $\hat{\bar{d}} = \sum_{i=0}^{t-1} \rho_i (1+\beta)^i$;

问题4：计算图的平均度

$\beta = \frac{\epsilon}{c_\beta}$, $|S| = \frac{c_s \log n}{\epsilon^{3.5}}\sqrt{\frac{n}{\alpha}} = \tilde{O}(\frac{\log n}{\epsilon^{3.5}})\sqrt{\frac{n}{\alpha}}$,

$t = \lceil \log_{(1+\beta)} n \rceil + 1$, $|S_i| \ge \frac{1}{t}\sqrt{\frac{3}{8}\cdot\frac{\epsilon\alpha}{n}}|S| = \theta_\rho |S|$

$$\hat{\bar{d}} = \sum_{i=0}^{t-1} \rho_i (1+\beta)^i \approx \sum_{i=0}^{t-1} \frac{|B_i|}{n}(1+\beta)^i \approx \bar{d}$$

▷ 如果对于所有 B_i，ρ_i 都是非常接近 $\frac{|B_i|}{n}$，OK
▷ 如果$|B_i|$很大，根据Chernoff界，应该OK
▷ 如果$|B_i|$很小，估计的方差可能会很大，估计为0
  ○ 本质上是计算边数的2倍，即$\bar{d}\cdot n$
  ○ 当边的一个顶点在小B_i中，误差为2
  ○ 当边的两个顶点在小B_i中，误差很大，但这样的边数大小可控

问题4：计算图的平均度

$\beta = \frac{\epsilon}{c_\beta}$, $|S| = \frac{c_s \log n}{\epsilon^{3.5}}\sqrt{\frac{n}{\alpha}} = \tilde{O}(\frac{\log n}{\epsilon^{3.5}})\sqrt{\frac{n}{\alpha}}$,

$t = \lceil \log_{(1+\beta)} n \rceil + 1$, $|S_i| \geq \frac{1}{t}\sqrt{\frac{3}{8}\cdot\frac{\epsilon\alpha}{n}}|S| = \theta_\rho |S|$

$\frac{c_s}{c_\beta} = O(\log\frac{t}{\delta}) = O(\log\frac{\log n}{\epsilon\delta}) \Rightarrow$ 至少以$1-\delta_1-\delta_2$概率

$$(0.5-\epsilon)\bar{d} < \hat{\bar{d}} < (1+\epsilon)\bar{d}$$

因此，算法II是一个$(2+\epsilon)$近似比的算法，时间复杂度为$\tilde{O}(\sqrt{n}\cdot poly(\frac{1}{\epsilon}))$

问题4：计算图的平均度

算法II的改进：

- 将节点分为 U（度数小）和 $V \setminus U$（度数大），近似比中的 2 因为 $E(V \setminus U, U)$ 只计算了一次
- 估计$E(V \setminus U, U)$的比例，能校正这部分错误
- 通过从采样得到的节点邻居中随机选取一条边实现
  - 随机邻居查询（random neighbor query）
  - $rnq(v)$
    - ✓ 获得 $d(v)$
    - ✓ 随机生成 $i \in [1, d(v)]$
    - ✓ 获得 $N(v, i)$

问题4：计算图的平均度

算法II的改进：

$$|B_i|\cdot(1+\beta)^{i-1} < |E_i| \leq |B_i|\cdot(1+\beta)^i$$

访问 B_i 中的点来估计 E_i 中 $E(V \setminus U, U)$ 的比例

- 重复如下操作 m 次
  - 随机选择 $u \in B_i$ 利用S_i实现
  - 令 $u' = rnq(u)$ 利用rnq实现
  - 如果 $(u, u') \in E(V \setminus U, V \setminus U)$，令 $\delta_j = 0$
  - 否则 $(u, u') \in E(V \setminus U, U)$，令 $\delta_j = 1$ 如何判断？
- 返回 $\Delta_i = \frac{\sum \delta_j}{m}$

问题4：计算图的平均度

访问B_i中的点来估计E_i中$E(V \setminus U, U)$的比例

- 令 T_i 为 B_i 中 big-small 的边数
- 简单情况：B_i 中点度数 d 相同

$$\mathbf{E}[\Delta_i] = \Pr[\delta_j = 1] = \frac{T_i}{d\cdot|B_i|}$$

- 一般情况：B_i 中度数比值在 $(1+\beta)$ 内

$$\frac{T_i}{|B_i|(1+\beta)^i} \leq \mathbf{E}[\Delta_i] \leq \frac{T_i}{|B_i|(1+\beta)^{i-1}}$$

$$|B_i|(1+\beta)^{i-1}\mathbf{E}[\Delta_i] \leq T_i \leq |B_i|(1+\beta)^i\mathbf{E}[\Delta_i]$$

- $\mathbf{E}[\Delta_i]$ 的$(1+\epsilon)$估计
  $\Rightarrow \Delta_i\rho_i(1+\beta)^i$ 是 $\frac{T_i}{n}$ 的 $(1+\epsilon)(1+\beta)$ 估计

问题4：计算图的平均度

改进算法：近似比$(1\pm\epsilon)$；运行时间$\tilde{O}(\sqrt{n}\cdot poly(1/\epsilon))$

平均度估计算法III

```
Sample S of V, |S| = Õ(L/(ρε²)), L = poly(log n / ε), θ_ρ = (1/t)·sqrt(3ε/8 · α/n);
S_i ← S ∩ B_i;
for i ∈ {0, ···, t−1} do
    if |S_i| ≥ θ_ρ|S| then
        ρ_i ← |S_i|/|S|;
        for v ∈ S_i do
            u ← rnq(v); δ(v) ← 0;
            if u ∈ B_j ∧ |S_j| < θ_ρ|S| then δ(v) ← 1;
            Δ_i = Σ_{v∈S_i} δ(v)/|S_i|;
    else
        ρ_i ← 0;
return d̄̂ = Σ_{i=0}^{t−1} (1 + Δ_i)ρ_i(1 + β)^i;
```


问题4：计算图的平均度

消去参数$\alpha \leq \bar{d}$，即度数的下界

平均度估计算法VI

```
α ← n;
d̄̂ ← −∞;
while d̄̂ < α do
    α ← α/2;
    if α < 1/n then
        return 0;
    利用估计算法 III 在参数 α 下运行，估计结果为 d̄̂;
return d̄̂;
```

近似比：$(1+\epsilon)$；
运行时间：$\tilde{O}(\sqrt{n}\cdot poly(\epsilon^{-1}))$

问题4：计算图的平均度

如果有parallel边或者边带权重，那么估计算法的下界$\Omega(n)$

- G_1: n-cycle graph
- G_2: $n-2$-cycle graph + one $c\cdot n$ parallel edge
- $\bar{d}(G_1) = 2$
- $\bar{d}(G_2) = c + 2$

问题5：Vertex Cover

顶点覆盖（Vertex Cover）

输入：$G = (V, E)$、最大度数d
图G用邻接链表表示
$C \subseteq V$是一个VC (Vertex Cover)当且仅当
$\forall (u, v) \in E$有$u \in C$或者$v \in C$

输出：最小VC的大小 $|VC|$

- $|VC| \geq \frac{|E|}{d}$
- NP-完全问题
- 存在集中式的2近似算法

问题5：Vertex Cover

在亚线性时间内，能有多好的近似算法？

- ▷ 乘性的近似比 ($\frac{|\hat{VC}|}{|VC|}$)：No
  Graph with 0 edge ⇒ $|VC| = 0$
  Graph with 1 edge ⇒ $|VC| = 1$
  区分上述两种情况需要 $\Omega(n)$ 时间
- ▷ 加性的近似比 ($|\hat{VC}| - |VC|$)：Hard
  乘性近似比的下界是 1.36，很可能是 2

定义1.30[(α,ϵ)-近似]
对于最优解是y的最小化问题，如果
$$y \le \hat{y} \le \alpha y + \epsilon$$
则称$\hat{y}$是该问题的(α,ϵ)-近似。

问题5：Vertex Cover

设计想法：利用分布式算法设计亚线性算法

分布式网络
- ▷ max degree d
- ▷ 每个节点是一个处理器，知道它的邻居是谁
- ▷ 同步计算、通信

每一轮（同步）
- ▷ 每个节点计算基于它的输入、随机生成位、通信中收到的信息
- ▷ 向每个邻居节点发送消息
- ▷ 从每个邻居节点收取消息

问题5：Vertex Cover

设计想法：利用分布式算法设计亚线性算法

分布式顶点覆盖（Vertex Cover）问题
- ▷ 输入：网络图即是要计算顶点覆盖的图 G
- ▷ 输出：每个节点知道自己是否属于 VC

Fast Distributed Alg. ⇒ SubLinear Time Alg.
- ▷ k 轮分布式算法中，节点 v 最多依赖于距离为 k 的节点，至多 d^k 个
- ▷ 集中式算法可以在 d^k 时间内，模拟分布式算法，计算 v 是否属于VC

注：如果是随机算法，需要知道所有d^k个节点的随机位

问题5：Vertex Cover

- ▷ 存在fast VC distribute alg (local distributed alg)?
- ▷ 如何利用分布式算法设计亚线性算法

Sublinear VC Alg (Parnas-Ron framework)
1. 独立均一地从 G 中选取 $s = \frac{8}{\epsilon^2}$ 个节点构成 S;
2. 为每个 $v \in S$，构造 k 步邻居导出的子图 $G_k(v)$;
3. 为每个 $v \in S$，在 $G_k(v)$ 上模拟分布式算法 $\mathcal{D}$;
4. 如果 $\mathcal{D}$ 返回 v 为覆盖节点之一，$X_v = 1$;
5. **return** $\hat{VC} = \frac{n}{s} \cdot \sum_{v\in S} X_v + \frac{\epsilon}{2} n$;

运行时间分析：$O(s \cdot d^k) = O(\frac{d^k}{\epsilon^2})$

问题5：Vertex Cover

- ▷ 存在fast VC distribute alg (local distributed alg)?

Distributed VC Alg
1. $\tilde{VC} \leftarrow \emptyset$;
2. **for** $i = 1$ to $\log d$ **do**
3. $\quad\Delta \leftarrow \{v : v \notin \tilde{VC}, deg(v) \ge d/2^i\}$;
4. $\quad\tilde{VC} \leftarrow \tilde{VC} \cup \Delta$;
5. $\quad$从 G 中删除所有与 Δ 邻接的边;
6. **return** $\tilde{VC}$;

- ▷ 运行时间：$d^{O(\log d)}$
- ▷ 近似比：$(O(\log d), \epsilon n)$

问题5：Vertex Cover

想法：利用贪心算法设计 $(2, \epsilon n)$ 亚线性算法
- ▷ Vertex Cover v.s. Maximum Matching
- ▷ Vertex Cover v.s. Maximal Matching
- ▷ $MM \le |VC| \le 2|MM|$

贪心算法（Greedy Sequential Matching Alg.）
1. $MM \leftarrow \emptyset$;
2. **for** 每一条边 $e \in E$ **do**
3. $\quad$**if** u 和 v 都未匹配 **then**
4. $\quad\quad$将边 e 的两个顶点加入 MM;
5. **return** MM;

问题5：Vertex Cover

▷ 如何利用贪心算法设计亚线性算法

Sublinear VC Alg II (Parnas-Ron framework)
1. 假设我们有一个Oracle B，给定 e 返回 $e \in MM$?
2. 独立均一地从 G 中选取 $s = \frac{8}{\epsilon^2}$ 个节点构成 S;
3. **for** $v \in S$ **do**
4. $\quad$**if** $\exists w \in N_v$ 使得 $B(v,w) = 1$ **then**
5. $\quad\quad X_v = 1$;
6. $\quad$**else**
7. $\quad\quad X_v = 0$;
8. **return** $\hat{VC} = \frac{n}{2s} \cdot \sum_{v\in S} X_v + \frac{\epsilon}{2} n$;

Alg II 运行时间 $O(Time(B)/\epsilon^2)$

问题5：Vertex Cover

▷ 如何实现Oracle B

Oracle B
1. 为查询边 e 随机生成一个 $rank_e \in [0,1]$;
2. **for** 每条 e 的邻居边 $e' \in E$ **do**
3. $\quad$**if** $rank_{e'} < rank_e$ **then**
4. $\quad\quad$递归调用 Oracle B 检查 e';
5. $\quad\quad$**if** $e' \in MM$ **then return** 0;
6. **return** 1;

Oracle B的期望运行时间$2^{O(d)}$ ⇒ Alg II运行时间$\frac{2^{O(d)}}{\epsilon^2}$

- 概率基础
- 亚线性空间算法
- 亚线性时间算法
- **并行模型算法**

并行计算模型

分布式计算环境（1k、1M台机器）如何计算？

- ▷ PRAM模型
- ▷ Bulk Synch Parallel (BSP)模型
- ▷ MapReduce模型

并行计算模型

MapReduce模型

- ▷ 所有数据形如$\langle key, value\rangle$，计算分轮次进行
- ▷ 每一轮分为：Map、Shuffle、Reduce
- ▷ Map: 每个数据被map函数处理，输出一个新的数据集合
- ▷ Shuffle: 对编程人员透明，所有在Map中输出的数据，被按照key分组，具备相同key的数据被分配到相同的reducer
- ▷ Reduce: 输入$\langle k, v_1, v_2, \ldots\rangle$，输出新的数据集合

设计目标：更少的轮数、更少的内存、更少的工作量、更大的并行度

问题1：基本问题

构建倒排索引

- ▷ Map函数
  输入：⟨ docID, content⟩
  输出：⟨ word, docID⟩
- ▷ Reduce函数
  输入：⟨ word, docID⟩
  输出：⟨ word, list of docID⟩

问题1：基本问题

单词计数

- ▷ Map函数
  输入：⟨ docID, content⟩
  输出：⟨ word, 1⟩
- ▷ Reduce函数
  输入：⟨ word, 1⟩
  输出：⟨ word, count⟩

问题1：基本问题

检索Search

- ▷ Map函数
  输入：⟨ (docID, lineno), content⟩
  输出：⟨ docID, NULL⟩
- ▷ Reduce函数
  输入：⟨ docID, (NULL, NULL, ...)⟩
  输出：⟨ docID, NULL⟩

问题1：基本问题

矩阵乘法A

- ▷ Map()
  - $((A, i, j), a_{ij}) \to (j, (A, i, a_{ij}))$
  - $((B, j, k), b_{jk}) \to (j, (B, k, b_{jk}))$
- ▷ Reduce()
  - $(j, (A, i, a_{ij})), (j, (B, k, b_{jk})) \to ((i, k), a_{ij} * b_{jk})$
- ▷ Map+(): identity
- ▷ Reduce+()
  - $((i, k), (v_1, v_2, \ldots)) \to ((i, k), \sum v_i)$

问题1：基本问题

矩阵乘法B

- ▷ Map()
  - $((A, i, j), a_{ij}) \to ((i, x), (A, j, a_{ij}))$ for all x
  - $((B, j, k), b_{jk}) \to ((y, k), (B, j, b_{jk}))$ for all y
- ▷ Reduce()
  - $((i, k), (A, j, a_{ij})) \wedge ((i, k), (B, j, b_{jk})) \to ((i, k), \sum a_{ij} * b_{jk})$

问题2：排序问题

使用p台处理器，输入$\langle i, A[i]\rangle$

TeraSort: Round 1

map: $\langle i, A[i]\rangle$
1. 输出 $\langle i\%p, ((i, A[i]), 0)\rangle$;
2. 以概率T/n，这里$T = \log(p)/\epsilon^2$
   对所有$j \in [0, p-1]$，输出 $\langle j, (A[i], 1)\rangle$;

reduce: $\langle j, (A[i], y)\rangle$
1. 将$y = 1$的数据收集为S并排序;
2. 将$y = 0$的数据收集为B;
3. 构造$(s_1, s_2, \ldots, s_{p-1})$，$s_j$为$S$中第$j\lceil\frac{|S|}{p}\rceil$，
   对$(i, x) \in B$满足$s_j < x \leq s_{j+1}$，输出$\langle j, (i, x)\rangle$;

问题2：排序问题

使用p台处理器，输入$\langle i, A[i]\rangle$

TeraSort: Round 2

map: $\langle j, (i, x)\rangle$
1. 输出 $\langle j, (i, x)\rangle$;

reduce: $\langle j, (i, x)\rangle$
1. 将所有(i, x)根据x排序并输出

- ▷ 随机算法
- ▷ $|S| \gg p$，样本点中的$p-1$个分点
  如果s_j是S中的α分点，那么以很高概率，它处于$\alpha n \pm \epsilon n$