数据流

概率计数

Morris 算法

考虑这样一个计数器问题：数据流中只有一种数据，也就是“按动计数器”操作。你的计数器可能随时被查询总共被按了多少次

这个问题十分平凡，一个精确算法如下：

- 维护一个数 n，初始时为 0
- 计数器被按动时，令 $n \leftarrow n+1$
- 查询时，返回 n

显然我们需要 $\Theta(\log n)$ 个比特来维护这个数据

但是实际上算法可以使用更少的空间。以下给出一个空间复杂度为 $\mathcal{O}\left(\log\log n\right)$ 的算法（Morris）

- 维护一个数 X，初始时为 0
- 计数器被按动时，以 2^{-X} 的概率令 $X \leftarrow X+1$
- 查询时，返回 $\tilde{n} = 2^X - 1$

算法分析

> 一个好的算法需要什么呢？需要以很大的概率给出小误差
>
> 只要 $\mathbb{E}[\tilde{n}] = n$ 且 $\mathrm{Var}[\tilde{n}]$ 不太大，那么我们可以使用切比雪夫不等式来证明算法以很大的概率相对误差比较小。我们首先证明 $\mathbb{E}[\tilde{n}] = n$

设 X_n 是按下计数器 n 次后 X 的值

引理： $\mathbb{E}\left[2^{X_n}\right] = n+1$

> 我们尝试找出 $\mathbb{E}\left[2^{X_{n+1}}\right]$ 与 $\mathbb{E}\left[2^{X_n}\right]$ 之间的关系
>
> 假如 $X_n = j$，那么我们很容易得出 X_{n+1} 的分布：有 2^{-j} 的概率变成 $j+1$，有 $1-2^{-j}$ 的概率不变。以此可以求出 $X_n = j$ 的时候 X_{n+1} 的期望。同时 $X_n = j$ 的时候 X_n 的期望就是 j，我们在这个基础上进行递推

证明： 考虑到 $X_0 = 0$ 即 $\mathbb{E}\left[2^{X_0}\right] = 1$，而且

$$
\begin{aligned}
\mathbb{E}\left[2^{X_{n+1}}\right] &= \sum_{j=0}^{\infty} \mathbb{E}\left[2^{X_{n+1}} \mid X_n = j\right] \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^{j+1} \cdot \Pr[X_{n+1} = j+1 \mid X_n = j] + 2^j \cdot \Pr[X_{n+1} = j \mid X_n = j]\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^{j+1} \cdot 2^{-j} + 2^j \cdot (1 - 2^{-j})\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^j + 1\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} 2^j \Pr[X_n = j] + \sum_{j=0}^{\infty} \Pr[X_n = j] \\
&= \mathbb{E}[2^{X_n}] + 1
\end{aligned}
$$

很容易递推得到 $\mathbb{E}\left[2^{X_n}\right] = n + 1$

由以上引理与 $\tilde{n} = 2^{X_n} - 1$ 可以得知 $\mathbb{E}[\tilde{n}] = n$

> 然后我们求出 $\mathrm{Var}[\tilde{n}]$，使用与前文类似的方法

观察到 $\mathrm{Var}[\tilde{n}] = \mathrm{Var}\left[2^{X_n}\right] = \mathbb{E}\left[\left(2^{X_n}\right)^2\right] - \mathbb{E}^2\left[2^{X_n}\right] = \mathbb{E}\left[2^{2X_n}\right] - (n+1)^2$。我们首先求出 $\mathbb{E}\left[2^{2X_n}\right]$

考虑到 $X_0 = 0$ 即 $\mathbb{E}\left[2^{2X_0}\right] = 1$，而且

$$
\begin{aligned}
\mathbb{E}\left[2^{2X_{n+1}}\right] &= \sum_{j=0}^{\infty} \mathbb{E}\left[2^{2X_{n+1}} \mid X_n = j\right] \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^{2j+2} \cdot \Pr[X_{n+1} = j+1 \mid X_n = j] + 2^{2j} \cdot \Pr[X_{n+1} = j \mid X_n = j]\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^{2j+2} \cdot 2^{-j} + 2^{2j} \cdot (1 - 2^{-j})\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} \left(2^{2j} + 3 \cdot 2^j\right) \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} 2^{2j} \Pr[X_n = j] + 3 \sum_{j=0}^{\infty} 2^j \Pr[X_n = j] \\
&= \mathbb{E}[2^{2X_n}] + 3\mathbb{E}\left[2^{X_n}\right] \\
&= \mathbb{E}\left[2^{2X_n}\right] + 3(n+1)
\end{aligned}
$$

易知

$$
\mathbb{E}\left[2^{2X_n}\right] = \mathbb{E}\left[2^{2X_0}\right] + \sum_{i=0}^{n-1} 3(i+1) = \frac{3}{2}n^2 + \frac{3}{2}n + 1
$$

所以

$$
\mathrm{Var}[\tilde{n}] = \mathrm{Var}\left[2^{X_n}\right] = \mathbb{E}\left[2^{2X_n}\right] - \mathbb{E}^2\left[2^{X_n}\right] = \frac{3}{2}n^2 + \frac{3}{2}n + 1 - (n+1)^2 = \frac{n^2}{2} - \frac{n}{2} < \frac{n^2}{2} = \mathcal{O}\left(n^2\right)
$$

这样切比雪夫不等式就可以估计出相对误差的概率

$$\Pr[|\tilde{n}-n|>\epsilon n] \le \frac{\mathrm{Var}[\tilde{n}]}{\epsilon^2 n^2} < \frac{1}{2\epsilon^2} = \mathcal{O}\left(\epsilon^{-2}\right)$$

> 很遗憾，这个式子没啥用，代入 $\epsilon=1$，这个式子不能表明本算法有很大的概率误差小于 100%

Morris+ 算法

为了能给出一个有效的随机计数算法，我们对 Morris 算法稍作修改（记作 Morris+ 算法）

1. 独立运行 s 次 Morris 算法，设这 s 个 Morris 算法的输出分别是 $\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_s$
2. 本算法的输出是这些输出的算术平均 $\tilde{n}=\frac{1}{s}\sum_{i=1}^{s}\tilde{n}_i$

我们对这个算法进行分析。首先，因为每个子 Morris 算法输出的期望都是 n，所以 Morris+ 算法的输出也是 n，而方差缩小了 s^2 倍

$$\mathbb{E}[\tilde{n}] = \mathbb{E}\left[\frac{1}{s}\sum_{i=1}^{s}\tilde{n}_i\right] = \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}[\tilde{n}_i] = \frac{1}{s}ns = n$$
$$\mathrm{Var}[\tilde{n}] = \mathrm{Var}\left[\frac{1}{s}\sum_{i=1}^{n}\tilde{n}_i\right] = \frac{1}{s^2}\sum_{i=1}^{n}\mathrm{Var}[\tilde{n}_i] < \frac{1}{s^2}s\frac{n^2}{2} < \frac{n^2}{2s}$$

由切比雪夫不等式

$$\Pr[|\tilde{n}-n|>\epsilon n] \le \frac{\mathrm{Var}[\tilde{n}]}{\epsilon^2 n^2} < \frac{\frac{n^2}{2s}}{\epsilon^2 n^2} = \frac{1}{2s\epsilon^2}$$

这样只要 $\frac{1}{2s\epsilon^2}\le\delta$，即 $s\ge\frac{1}{2\delta\epsilon^2}$，那么 Morris+ 算法就以超过 $1-\delta$ 的概率相对误差不超过 ϵ

Morris++ 算法

以上算法还有改进空间。算法描述如下：

1. 以 $\delta=\frac{1}{3}$ 为参数（也就是以 $s=\frac{1}{2\cdot\frac{1}{3}\epsilon^2}$ 为参数）调用 t 份独立的 Morris+ 算法作为子程序，设 Morris+ 算法的输出分别是 $\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_t$
2. 输出 $\tilde{n}_1, \tilde{n}_2, \cdots, \tilde{n}_t$ 的中位数

> 我们将 $|\tilde{n}-n|\le\epsilon n$，也就是相对误差不超过 ϵ 的概率计数算法输出称为 成功的。我们以 $\delta=\frac{1}{3}$ 为参数调用了 Morris+ 算法，也就是以至多 $\frac{1}{3}$ 的概率 Morris+ 子程序输出的结果是失败的。Morris+ 算法保证了当 t 非常大的时候，大数定律直观地告诉我们有大约至少 $\frac{2}{3}t$ 的 Morris+ 算法子程序是成功的
>
> 而 Morris++ 算法只要有超过 $\frac{1}{2}t$ 的 Morris+ 算法子程序是成功就能输出一个成功结果——反着考虑，如果 Morris++ 输出了一个失败的结果，也就是如果 $|\tilde{n}-n|>\epsilon n$，那么要么 $\tilde{n}$ 太小了，也就是 $\tilde{n}-n<-\epsilon n$，要么 $\tilde{n}$ 太大了，也就是 $\tilde{n}-n>\epsilon n$。如果 $\tilde{n}$ 过小，因为 $\tilde{n}$ 是中位数，所以有 $\frac{1}{2}t$ 个 Morris+ 子程序输出比 $\tilde{n}$ 还小，它们全都是失败的；如果 $\tilde{n}$ 过大，那么有 $\frac{1}{2}t$ 个 Morris+ 子程序输出比 $\tilde{n}$ 还大，它们也全都是失败的。只要 $\tilde{n}$ 失败，那么至少有 $\frac{1}{2}t$ 个 Morris+ 子程序失败
>
> 这也就表明成功的 Morris+ 算法子程序数 $|\{i \mid |\tilde{n}_i-n|\le\epsilon n\}|$ 大于 $\frac{1}{2}t$，那么 Morris++ 算法成功，也就是 $|\{i \mid |\tilde{n}_i-n|\le\epsilon n\}|>\frac{1}{2}t \implies \tilde{n}-n\le\epsilon n$，它的逆否命题是 $\tilde{n}-n>\epsilon n \implies |\{i \mid |\tilde{n}_i-n|\}|\le\frac{1}{2}t$。这就表明了 $\Pr[|\tilde{n}-n|>\epsilon]\le\Pr\left[|\{i \mid |\tilde{n}_i-n|\}|\le\frac{t}{2}\right]$

对于每一个 $i\le t$，设 $Y_i=\begin{cases}1 & |\tilde{n}_i-n|\le\epsilon n\\ 0 & |\tilde{n}_i-n|>\epsilon n\end{cases}$

那么由 Morris+ 算法的性质知 $\Pr[Y_i=1]>\frac{2}{3}$，这表明

$$\mathbb{E}[Y_i]=1\cdot\Pr[Y_i=1]+0\cdot\Pr[Y_i=0]=\Pr[y_i=1]>\frac{2}{3}$$

设 $Y=\sum_{i=1}^{t}Y_i$，则 Y 表示有多少个 Morris+ 子程序成功了，也就是 $Y=|\{i\mid|\tilde{n}_i-n|\le\epsilon n\}|$。那么就有

$$\mathbb{E}[Y]=\mathbb{E}\left[\sum_{i=1}^{t}Y_i\right]=\sum_{i=1}^{t}\mathbb{E}[Y_i]>\frac{2}{3}t$$

所以

$$\Pr[|\tilde{n}-n|>\epsilon n]\le\Pr\left[Y\le\frac{t}{2}\right]<\Pr\left[Y-\mathbb{E}[Y]<-\frac{t}{6}\right]\le\exp\left(-2\left(\frac{t}{6}\right)^2\frac{1}{t}\right)$$

最后一个不等式成立是 Chernoff Bound 得出的

这样，只要 $t\ge 18\ln\frac{1}{\delta}$，Morris++ 算法就能成功

空间复杂度

当相对误差超过 ϵ 的概率不足 δ 的时候，Morris+ 算法调用了 $s=\Theta\left(\frac{1}{\delta\epsilon^2}\right)$ 次 Morris 算法，而 Morris++ 算法调用了 $st=\Theta\left(\frac{1}{\epsilon}^2\ln\frac{1}{\delta}\right)$ 次 Morris 算法。我们接下来证明：如果调用了 s^* 次 Morris 算法，那么以至少 δ^* 的概率，在这 n 次计数过程中每一个数都不超过 $\log\left(\frac{s^*n}{\delta^*}\right)$

假设某个 Morris 算法在过程中达到了 $X=\log\left(\frac{s^*n}{\delta^*}\right)$，那么它再增加一次的概率是 $\frac{1}{2^X}\le\frac{\delta^*}{s^*n}$。因为 X 总共只有 n 个机会增加，所以 X 在算法结束的时候有至多 $\frac{n}{2^X}\le\frac{\delta^*}{s^*}$ 的概率增加。总共有 s^* 个 Morris 算法，至多有 s^* 个数达到了 $\log\left(\frac{s^*n}{\delta^*}\right)$ 随时准备突破。所以在算法结束的时候有至多 δ^* 的概率某个 Morris 算法的 X 超过 $\log\left(\frac{s^*n}{\delta^*}\right)$ 了。这就表明有至少 $1-\delta^*$ 的概率所有到达过临界值 $\log\left(\frac{s^*n}{\delta^*}\right)$ 的 X 都不会再增长了，也就表明有至少 $1-\delta^*$ 的概率所有 Morris 算法中的 X 都不超过 $\log\left(\frac{s^*n}{\delta^*}\right)$

这就表明 Morris++ 算法以至少 $1-\delta^*$ 的概率空间复杂度为

$$\mathcal{O}\left(st\log\log\left(\frac{stn}{\delta^*}\right)\right)=\mathcal{O}\left(\left(\frac{1}{\epsilon}\right)^2\ln\frac{1}{\delta}\log\log\frac{n\log\frac{1}{\delta}}{\epsilon^2\delta^*}\right)$$

> 以上分析比较粗糙。一些古老的工作将这个复杂度改进到了 $\mathcal{O}\left(\log\frac{1}{\epsilon}+\log\log n+\log\frac{1}{\delta}\right)$，一些较新的工作将这个复杂度改进到了 $\Theta\left(\log\frac{1}{\epsilon}+\log\log n+\log\log\frac{1}{\delta}\right)$

蓄水池抽样

你面前有一个数据流，数据在不停地流过——你只能看到每个数据一次并且不能将它们全部存储下来。你在任意时刻都可能被要求：“将刚刚你看到的所有数中均匀随机抽取一个给我。”

假设你看到的数依次为 $\{a_i\}_{i=1,2,\cdots,\infty}$。类似于 `std::shuffle` 的思想，你可以实现如下地算法：

- 维护变量 s，初始时值未定义
- 当看到数据 a_m 的时候，掷骰子，以 $\frac{1}{m}$ 的概率令 $s\leftarrow a_m$，以 $1-\frac{1}{m}$ 的概率保持 s 不变
- 查询时，直接输出 s

这个算法的正确性比较显然。当数据流中流过 m 个数的时候，如果 $s=a_i$，那么第 i 次掷骰子掷得了 $\frac{1}{i}$ 将 a_i 保留下来，而在第 $i+1, i+2, \cdots m$ 的时候掷得了 $1-\frac{1}{i+1}, 1-\frac{1}{i+2}, \cdots, 1-\frac{1}{m}$ 而没有将 a_i 扔掉。这些事件都是随机的，所以

$$\Pr[s_m=a_i]=\frac{1}{i}\cdot\frac{i}{i+1}\cdot\frac{i+1}{i+2}\cdots\frac{m-1}{m}=\frac{1}{m}$$

对于任意的 $i=1,2,\cdots,m$ 均成立，也就是说 $a_1, a_2, \cdots, a_m$ 能等概率地在第 m 次出现

分析一下空间复杂度。我们需要存储被抽样的数与当前经过了多少数。假设所有数都不超过 n，那么我们的空间复杂度就是 $\mathcal{O}\left(\log n+\log m\right)$

估计不同元素个数

你需要在任意时刻估计当前数据流中不同元素的个数，并且用尽可能少的事件。设数据流是 $\{a_i\}_{i=1,2,\cdots}$ 而且所有数都是整数，你需要在逐一读取这些数据的时候随时准备好回答 $|\{a_1, a_2, \cdots, a_m\}|$

设这个数据流中数的上界是 n，你在读取完前 m 个数的时候被要求作答。如果需要求出精确解，有两种方法：

1. 维护一个长度为 n 的数组 $v\in\mathbb{R}^n$ 与计数器 X。初始时 $v=\vec{0}$。读到一个数 a_i 的时候，如果 $v_{a_i}=0$，那么就令计数器 $X\leftarrow X+1$，同时令 $v_{a_i}\leftarrow 1$；如果 $v_{a_i}=1$，表明 a_i 已经出现过了，就什么都不做。这种方法需要 n 个比特。
2. 直接将读过的数存下来，并维护一个计数器 X。当新读到一个数的时候，和之前读过的所有数比较，如果与读过的所有书都不同就令计数器加一。这种方法需要存储 $\mathcal{O}\left(m\log n\right)$ 个比特

我们想要一个空间复杂度是 $\mathcal{O}\left(\log nm\right)$ 的近似算法，也就是需要对任意的参数 $\epsilon,\delta\in(0,1)$，求出一个 $\tilde{t}$ 满足 $\Pr\left[\left|t-\tilde{t}\right|>\epsilon t\right]<\delta$，其中 t 是数据流中不同元素的个数

理想 FM 算法

理想 FM 算法描述如下：

1. 预处理出一个随机函数 $h:\{1,2,\cdots,n\}\mapsto[0,1]$
2. 维护一个 *计数器* X，记录当前看到所有数据被映射到的最小值，也就是 $X=\min_{i\in\text{stream}}h(i)$
3. 输出 $\tilde{t}=\frac{1}{X}-1$

在分析这个算法之前，我们先来证明这样一个概率论中常用的性质：若 X 是非负随机变量，则

$$\mathbb{E}[X]=\int_0^\infty\Pr[X\geq\lambda]\,\mathrm{d}\lambda$$

设 X 的概率密度函数是 $f(x)$，那么

$$
\begin{aligned}
\mathbb{E}[X] &= \int_0^{\infty} x f(x)\,\mathrm{d}x \\
&= \int_0^{\infty}\left(\int_0^{x}\mathrm{d}\lambda\right) f(x)\,\mathrm{d}x \\
&= \iint_{0\le\lambda\le x\le\infty} f(x)\,\mathrm{d}x\,\mathrm{d}\lambda \\
&= \int_0^{\infty}\left(\int_{\lambda}^{\infty} f(x)\,\mathrm{d}x\right)\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr[X\ge\lambda]\,\mathrm{d}\lambda
\end{aligned}
$$

我们先证明算法期望能输出正确的结果。算法的结果只与 X 相关，所以我们先分析 X。先分析期望。假设 $a_1, a_2, \cdots, a_m$ 是数据流，其中有 t 个不同的元素，也就是说 $|\{a_1, a_2, \cdots, a_m\}| = t$

$$
\begin{aligned}
\mathbb{E}[X] &= \int_0^{\infty}\Pr[X\ge\lambda]\,\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr[h(a_1)\ge\lambda\wedge h(a_2)\ge\lambda\wedge\cdots\wedge h(a_m)\ge\lambda]\,\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr\left[h(a)\ge\lambda\right]^{|\{a_1,a_2,\cdots,a_m\}|}\,\mathrm{d}\lambda \\
&= \int_0^{1}(1-\lambda)^t\mathrm{d}\lambda \\
&= -\frac{(1-\lambda)^{t+1}}{t+1}\Bigg|_0^1 = \frac{1}{t+1}
\end{aligned}
$$

然后我们分析方差，为此我们先计算 X^2 的期望。同样地，有

$$
\begin{aligned}
\mathbb{E}\left[X^2\right] &= \int_0^{\infty}\Pr\left[X^2\ge\lambda\right]\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr\left[X\ge\sqrt{\lambda}\right]\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr\left[\bigwedge_{i=1}^{m} h(a_i)\ge\sqrt{\lambda}\right]\mathrm{d}\lambda \\
&= \int_0^{\infty}\Pr\left[h(a)\ge\sqrt{\lambda}\right]^{|\{a_1,a_2,\cdots,a_m\}|}\mathrm{d}\lambda \\
&= \int_0^{1}\left(1-\sqrt{\lambda}\right)^t\mathrm{d}\lambda \\
&= 2\int_0^{1}(1-\lambda)^t\lambda\,\mathrm{d}\lambda \\
&= 2\left(\int_0^{1}(1-\lambda)^t\,\mathrm{d}\lambda-\int_0^{1}(1-\lambda)^{t+1}\,\mathrm{d}\lambda\right) \\
&= 2\left(\frac{1}{t+1}-\frac{1}{t+2}\right) = \frac{2}{(t+1)(t+2)}
\end{aligned}
$$

所以可以算出方差 $\mathrm{Var}(X) = \mathbb{E}\left[X^2\right]-\mathbb{E}^2[X] = \frac{t}{(t+1)^2(t+2)} < \frac{1}{(t+1)^2}$

理想 FM+ 算法

与 Morris+ 算法类似地，我们多跑几遍 FM 就可以让它的精度高一些。FM+ 算法要求精度 $\epsilon \le \frac{1}{2}$ 才能很好地运行。描述如下：

1. 独立运行 $s = \frac{25}{\epsilon^2\delta}$ 个 FM 算法，设它们的 *计数器* 返回 $X_1, X_2, \cdots, X_s$
2. 输出 $\tilde{t} = \frac{1}{\overline{X}} - 1$，$\overline{X} = \frac{1}{s}\sum_{i=1}^{s} X_i$ 是这些 FM 算法 *计数器* 的均值

> 注意，这里是调用 FM 算法的计数器 X，而非它们的返回值 $\tilde{t}$。这是因为我们上面只对 X 的期望和方差做出了分析，但是我们没有分析 $\tilde{t}$ 的期望和方差。X 的期望是 $\frac{1}{t+1}$ 不能表明 $\frac{1}{X}$ 的期望是 $t+1$，也就不能表明 $\tilde{t} = \frac{1}{X} - 1$ 的期望是 t

在本节中，我们仅考虑 $t \ge 1$ 的非平凡情况

> 对于平凡情况 $t = 0$，每个计数器都等于它的初值。只要所有计数器的初值都是 1 就可以在 $t = 0$ 的时候输出正确解

我们证明 $\overline{X}$ 离 $\frac{1}{t+1}$ 足够近。根据 FM 算法的分析，我们知道 $\mathbb{E}[X_i] = \frac{1}{t+1}$ 与 $\mathrm{Var}[X_i] < \frac{1}{(t+1)^2}$。所以

$$\mathbb{E}\left[\overline{X}\right] = \frac{1}{t+1}$$
$$\mathrm{Var}\left[\overline{X}\right] < \frac{1}{s} \cdot \frac{1}{(t+1)^2}$$

代入 $s = \frac{25}{\epsilon^2\delta}$，由切比雪夫不等式知

$$\Pr\left[\left|\overline{X} - \frac{1}{t+1}\right| > \frac{\epsilon/5}{t+1}\right] < \frac{\mathrm{Var}\left[\overline{X}\right]}{(\epsilon/5)^2} < \delta$$

所以

$$\Pr\left[\frac{1-\epsilon/5}{t+1} < \overline{X} < \frac{1+\epsilon/5}{t+1}\right] > 1-\delta$$
$$\Pr\left[\frac{t+1}{1+\epsilon/5} < \frac{1}{\overline{X}} < \frac{t+1}{1-\epsilon/5}\right] > 1-\delta$$
$$\Pr\left[\frac{t-\epsilon/5}{1+\epsilon/5} < \frac{1}{\overline{X}} - 1 < \frac{t+\epsilon/5}{1-\epsilon/5}\right] > 1-\delta$$

首先注意到

$$\frac{t-\epsilon/5}{1+\epsilon/5} > (t-\epsilon/5)(1-\epsilon/5) > t - \frac{t+1}{5}\epsilon > t - \frac{2}{5}\epsilon t > t - \epsilon t$$

然后回想先前规定 $\epsilon < \frac{1}{2}$，也就表明

$$\frac{t+\epsilon/5}{1-\epsilon/5} < (t+\epsilon/5)(1+2\epsilon/5) = t + \frac{2}{5}\epsilon t + \frac{1}{5}\epsilon + \frac{2}{5}\epsilon\cdot\epsilon \le t + \frac{2}{5}\epsilon t + \frac{1}{5}\epsilon t + \frac{2}{5}\epsilon t = t + \epsilon t$$

所以

$$\Pr\left[t-\epsilon t<\frac{1}{\overline{X}}-1<t+\epsilon t\right]\geq\Pr\left[\frac{t-\epsilon/5}{1+\epsilon/5}<\frac{1}{\overline{X}}-1<\frac{t+\epsilon/5}{1-\epsilon/5}\right]>1-\delta$$
$$\Pr\left[\left|\left(\frac{1}{\overline{X}}-1\right)-t\right|>\epsilon t\right]<\delta$$

理想 FM++ 算法

与 Morris++ 算法类似地，我们利用多次运行 FM+ 算法得到的中位数来提高算法的性能

1. 独立运行 $q=18\ln\frac{1}{\delta}$ 次成功率 $\delta_{\mathrm{FM+}}=\frac{1}{3}$ 的 FM+ 算法，设它们的估计分别为 $\tilde{t}_1,\tilde{t}_2,\cdots,\tilde{t}_q$
2. 输出它们的中位数 $\tilde{t}=\mathrm{median}(\tilde{t}_1,\tilde{t}_2,\cdots,\tilde{t}_q)$

这一算法的分析与 Morris++ 可以说是一模一样了，可以证明 $\Pr\left[\left|\tilde{t}-t\right|>\epsilon t\right]<\delta$

我们首先令 $Y_j\triangleq\left[\left|\tilde{t}_j-t\right|<\epsilon t\right]=\begin{cases}1 & \left|\tilde{t}_j-t\right|<\epsilon t\\ 0 & \text{otherwise}\end{cases}$，然后令 $Y\triangleq\sum_{i=1}^q Y_i$。由我们调用 FM+ 子算法时规定失败率至多为 $\delta_{\mathrm{FM+}}=\frac{1}{3}$，所以 $\mathbb{E}[Y]\geq\frac{2}{3}q$。代入 $q=18\ln\frac{1}{\delta}$，由 Chernoff Bound 知

$$\Pr\left[\left|\tilde{t}-t\right|>\epsilon t\right]\leq\Pr\left[Y\leq\frac{q}{2}\right]\leq\Pr\left[|Y-\mathbb{E}[Y]|<\frac{q}{6}\right]\leq\exp\left(-2\left(\frac{1}{6}\right)^2q\right)\leq\delta$$

独立哈希函数族

如果一个 $\{1,2,\cdots,a\}\mapsto\{1,2,\cdots,b\}$ 的哈希函数族 $\mathcal{H}$ 满足：对于任意的 $j_1,j_2,\cdots,j_k\in\{1,2,\cdots,b\}$ 与 $i_1,i_2,\cdots,i_k\in\{1,2,\cdots,a\}$，随机挑选一个哈希函数 $h\in_R\mathcal{H}$，有

$$\Pr[h(i_1)=j_1\wedge h(i_2)=j_2\wedge\cdots\wedge h(i_k)=j_k]=\frac{1}{b^k}$$

那么我们称这个函数族是 k 元独立的

- 如果函数族 $\mathcal{H}$ 包含了全体 $\{1,2,\cdots,a\}\mapsto\{1,2,\cdots,b\}$ 的映射，那么这个 $\mathcal{H}$ 就是一个 k 元独立的哈希函数。但是存储 $\mathcal{H}$ 中的元素需要 $a\log_2 b$ 个比特
- 如果 $q=p^r$ 是质数的幂（也就是有且仅有一个质因数），且 $a=b=q$，在 q 阶有限域中，全体度数不超过 $k-1$ 的多项式组成的函数族 $\mathcal{H}_{\mathrm{poly}}=\{h\mid h(x)=d_0+d_1x+\cdots+d_{k-1}x^{k-1}\}$ 组成一个独立哈希函数族，而存储这个哈希函数族内的元素仅需要存储这 k 个系数即可，只需要 $k\log q$ 个比特

> 对于任意的质数 p 与整数 r，阶为 p^r 的域是存在且唯一的，详细证明见这个博客，需要一些近世代数知识。简单点说就是将 $x^{p^r}-x$ 的所有“根”加入域中。这里给出一个需要一些数论知识能看懂的构造方法
>
> 度数不超过 $r-1$ 的全体模 p 意义下多项式的集合 $\{A\mid A(x)=a_0+a_1x+\cdots+a_{r-1}x^{r-1},\vec{a}\in\mathbb{F}_p^r\}$ 在以下加法与乘法意义下是一个阶为 p^r 的域 $\mathbb{F}_{p^r}$：
>
> - 加法 $+$ 是普通多项式的加法：
>
> $$(A+B)(x)=(a_0+b_0\bmod p)+(a_1+b_1\bmod p)x+\cdots+(a_{r-1}+b_{r-1}\bmod p)x^{r-1}$$
>
> - 在定义乘法 $\cdot$ 的时候，需要首先指定多项式环 $\mathbb{F}_p[x]$ 中的一个 r 次首一不可约多项式 M（也就是 M 的次数为 r，首项系数为 1 且不存在 M_1,M_2 满足 $M(x)=M_1(x)M_2(x)$ 且 $\deg M_1,\deg M_2\geq 1$）。当计算 $A\cdot B$ 的时候，首先计算 $\mathbb{F}_p[x]$ 中的多项式乘法，然后除以 $M(x)$ 并取出余数，也就是

> $$C(x) \leftarrow A(x)B(x) = \sum_{n=0}^{2r-2}\left(\sum_{i+j=n} a_i b_j \bmod p\right)x^n$$
> $$C(x) \rightarrow Q(x)M(x) + R(x), \deg R < r, R, Q \in \mathbb{F}_p$$
> $$A \cdot B \leftarrow R$$
>
> 遗憾的是以上构造不能直接用于证明阶为 p^r 域的存在性，这是因为对于任意的质数 p 与整数 r，$\mathbb{F}_p[x]$ 中 r 次首一不可约多项式的存在性是依赖于 $\mathbb{F}_{p^r}$ 的存在唯一性的

$k-$Minimum Values 算法

1. 取出一个 2 元独立的哈希函数 $h:\{1,2,\cdots,n\} \mapsto \{1,2,\cdots,M\}$，其中 $M \geq n^3$
2. 维护当前看到所有数的哈希值中，最小的 k 个（可以通过用插入排序维护一个有序数组，或者优先级队列实现）。这里 $k = \left\lceil \frac{24}{\epsilon^2} \right\rceil$
3. 在查询时
   - 如果目前收集到的哈希值不足 k 个，那么输出目前收集到的哈希值 k 个
   - 否则，设第 k 小的哈希值为 X，那么输出 $\frac{kM}{X}$

> 首先解释一下哈希函数的选取。我们选取前文提到的多项式哈希函数。因为这里要求 2 元独立哈希函数，所以取一个线性函数就行了，有两个随机参数。前文提到的多项式函数是 $\{1,2,\cdots,p^r\} \mapsto \{1,2,\cdots,p^r\}$ 的——事实上有限域的阶数必须是质数的幂。所以我们选取的 M 是不小于 n^3 的最小质数幂（一定小于 $2n^3$，因为 $n^3, n^3+1, \cdots, 2n^3-1$ 中一定有一个 2 的幂），并选取相应的随机哈希函数
>
> 在这个算法中，与先前不一样的是，我们不用最小值来估计元素个数，而是用第 k 小的数来估计元素个数
>
> 在理想 FM 算法中，如果我们用第 k 小的数来估计元素个数，那么当数据流中有 t 个不同的数时，最小的数大约是 $\frac{1}{t+1}$，那么从直观的均匀分布取理解，第 k 小的数就是 $\frac{k}{t+1}$。直观地，当使用大小为 M 的随机哈希函数时，第 k 小的数是 $\frac{k}{t+1}M$。这样
>
> $$X \approx \frac{k}{t+1}M$$
> $$t \approx \frac{kM}{X} - 1 \approx \frac{kM}{X}$$

定理：如果 $\frac{1}{\sqrt{n}} < \epsilon < \frac{1}{2}$，那么以至少 $\frac{2}{3}$ 的概率 $\tilde{t}$ 的相对误差不超过 ϵ，也就是

$$\Pr\left[\left|\tilde{t} - t\right| \leq \epsilon t\right] \geq \frac{2}{3}$$

这个算法需要维护 k 个大小不超过 n 的数与一个随机哈希函数，其中 $k = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$。这个随机哈希函数取上一节提到的多项式函数，因为是 2 元独立，所以只需要维护一个线性函数，有两个系数。每个系数的数量级是 $\mathcal{O}(M) = \mathcal{O}\left(n^3\right)$，所以每个系数需要占用 $\mathcal{O}(\log M) = \mathcal{O}(\log n)$ 个比特，总空间复杂度是 $k\,\mathcal{O}(\log n) + 2\,\mathcal{O}(\log n) = \mathcal{O}\left(\frac{\log n}{\epsilon^2}\right)$

> 以上算法的成功率只有 $\frac{2}{3}$。提高它的成功率可以用前文提到的方法，独立运行 $18\ln\frac{1}{\delta}$ 次取中位数就能让成功率提高到 $1-\delta$

证明：

我们首先估计 $\Pr\left[\tilde{t} > (1+\epsilon)t\right]$。首先对事件的表述进行一些转换

$$\tilde{t} > (1+\epsilon)t \iff \frac{kM}{X} > (1+\epsilon)t \iff X < \frac{kM}{(1+\epsilon)t}$$

因为 X 是第 k 小的哈希值，所以 $X < \frac{kM}{(1+\epsilon)t}$ 等价于在全部 t 个不同的哈希值中，有至少 k 个被哈希到的值小于 $\frac{kM}{(1+\epsilon)t}$ 的值上

与前文类似地，我们设 t 个随机布尔变量 $Y_1, Y_2, \cdots, Y_t$，$Y_i = 1$ 等价于第 i 个数被哈希到了小于 $\frac{kM}{(1+\epsilon)t}$ 的值上，同时设 $Y = \sum_{i=1}^{t} Y_i$。我们估计它的期望和方差，然后用切比雪夫不等式估计 $Y \geq k$ 的概率

首先估计期望。我们假设 h 是 2 元独立随机哈希函数，所以 $\mathbb{E}[Y_i] = \Pr[Y_i = 1] < \frac{k}{(1+\epsilon)t}$。由期望的可加性，$\mathbb{E}[Y] = \sum_{i=1}^{t} \mathbb{E}[Y_i] < \frac{tk}{(1+\epsilon)t} = \frac{k}{1+\epsilon}$

然后估计方差。因为 $\operatorname{Var}[Y_i] = \mathbb{E}\left[Y_i^2\right] - \mathbb{E}^2[Y_i] < \mathbb{E}\left[Y_i^2\right] = \mathbb{E}[Y_i] < \frac{k}{(1+\epsilon)t}$

因为 h 是 2 元独立哈希函数，也就是说对于任意的 $i \neq j$，Y_i 与 Y_j 是独立的。所以 $\mathbb{E}[Y_iY_j] = \mathbb{E}[Y_i]\mathbb{E}[Y_j]$，也就是说 $\operatorname{Var}[Y] = \sum_{i=1}^{t} \operatorname{Var}[Y_i] < \frac{k}{1+\epsilon}$

联系到定理中的假设 $\epsilon < \frac{1}{2}$ 与 $k \geq \frac{24}{\epsilon^2}$，我们有

$$\Pr\left[\tilde{t} > (1+\epsilon)t\right] = \Pr[Y \geq k] \leq \Pr\left[|Y - \mathbb{E}[Y]| > k - \frac{k}{1+\epsilon}\right] < \frac{k}{1+\epsilon}\frac{(1+\epsilon)^2}{k^2\epsilon^2} = \frac{1+\epsilon}{\epsilon^2 k} < \frac{1}{6}$$

然后我们估计 $\Pr\left[\tilde{t} < (1-\epsilon)t\right]$。首先也是对事件的表述做一些转换

$$\tilde{t} < (1-\epsilon)t \iff \frac{kM}{X} < (1-\epsilon)t \iff X > \frac{kM}{(1-\epsilon)t}$$

如果 $X > \frac{kM}{(1-\epsilon)t}$，那么哈希值小于 $\frac{kM}{(1-\epsilon)t}$ 的数一定小于 k 个

同样地，设 Z_i 表示第 i 个数的哈希值小于 $\frac{kM}{(1-\epsilon)t}$，再设 $Z = \sum_{i=1}^{t} Z_i$

那么 $\Pr[Z_i = 1] = \left\lfloor \frac{k}{(1-\epsilon)t} \right\rfloor > \frac{k}{(1-\epsilon)t} - \frac{1}{M}$。联想到 $M \geq n^3 > \frac{4t}{\epsilon k}$，我们有 $\Pr[Z_i = 1] > \frac{(1+\epsilon)k}{t} - \frac{\epsilon k}{4t}$

这样我们可以估计 $\mathbb{E}[Z]$ 与 $\operatorname{Var}[Z]$，从而估计出概率

带修改的数据流

频繁项

输入：一个整数数据流 $i_1, i_2, \cdots, i_m \in \{1, 2, \cdots, n\}$

查询：如果当前的数据中有出现次数不少于 $\frac{m}{2}$ 次的元素，那么输出它，否则随便输出一个数

解决这个问题可以使用 Misra Gries 算法：

1. 维护一个数 I 和计数器 c。初始时 $c \leftarrow 0$
2. 当数据流中流入一个数 x 时
   - 如果 $x = I$，那么 $c \leftarrow c + 1$
   - 如果 $x \neq I$ 且 $c = 0$，那么 $I \leftarrow x$ 且 $c \leftarrow 1$
   - 如果 $x \neq I$ 且 $c \neq 0$，那么 $c \leftarrow c - 1$
3. 查询时，输出 I

以上算法的正确性显然，而且只需要维护两个数，空间复杂度是 $\mathcal{O}(\log n + \log m)$

但是以上算法的应用场景受限，一个数据流中某数出现超过一半确实不是一个常见的情景。我们将问题放松成这样：

输入：一个整数数据流 $i_1, i_2, \cdots, i_m \in \{1, 2, \cdots, n\}$

查询：输出 k 个数。所有出现次数严格大于 $\frac{m}{k+1}$ 次的元素必须包含在输出的 k 个数中，但是在这个前提之下可以输出出现次数不超过 $\frac{m}{k+1}$ 的数

解决这个问题可以把前文提到的 Misra Gries 算法修改一下：

1. 维护 k 个数 $\vec{I}$ 和 k 个计数器计数器 $\vec{c}$。初始时 $\vec{c} \leftarrow \vec{0}$
2. 当数据流中流入一个数 x 时
    - *自增：*如果 $x = I_j$，那么令其计数器自增 1，也就是 $c_j \leftarrow c_j + 1$
    - *自增：*如果 x 不在 $\vec{I}$ 中但 $c_j = 0$，那么将 x 放在这个位置并将计数器设为 1，也就是 $I_j \leftarrow x$，$c_j \leftarrow 1$
    - *全部自减：*如果 x 不在 $\vec{I}$ 中且 $\vec{c}$ 中所有的元素都非零，那么让所有计数器的值均减 1，也就是 $\vec{c} \leftarrow \vec{c} - \vec{1}$
3. 查询时，输出 $\vec{I}$

命题：任意满足 $f_i > \frac{m}{k+1}$ 的数 i 在算法结束时都在这个数组中

为了证明这个命题，我们定义一个辅助变量 $\hat{f}_i$：如果 i 在算法结束的时候在 $\vec{I}$ 中，那么 $\hat{f}_i$ 就是它对应计数器的值；如果 i 不在，那么 $\hat{f}_i = 0$。也就是 $\hat{f}_i = \begin{cases} c_j & I_j = i \\ 0 & I_j \neq i, j = 1, 2, \cdots, k \end{cases}$

定理：$f_i - \frac{m}{k+1} \leq \hat{f}_i \leq f_i$ 对任意的数 $i = 1, 2, \cdots, n$ 均成立

证明：我们将算法的描述改成如下等价形式：

1. 对每个数 $i = 1, 2, \cdots, n$ 都维护一个计数器 $\hat{f}_i$，初始时 $\vec{\hat{f}} \leftarrow \vec{0}$
2. 当数据流中流入第 j 个数 i_j 时
    - *自增：*如果 $\hat{f}_{i_j} > 0$，那么 $\hat{f}_{i_j} \leftarrow \hat{f}_{i_j} + 1$
    - *自增：*如果 $\hat{f}_{i_j} = 0$ 且当前计数器 $\vec{\hat{f}}$ 中的正数不足 k 个，那么 $\hat{f}_{i_j} \leftarrow 1 = \hat{f}_{i_j} + 1$
    - *全部自减：*如果 $\hat{f}_{i_j} = 0$ 且当前计数器 $\vec{\hat{f}}$ 中的正数至少 k 个，那么令 $\vec{\hat{f}}$ 中所有的正数都自减 1

注意到以上过程中，每出现一次 i，$\hat{f}_i$ 至多自增 1；在不出现 i 的时候 $\hat{f}_i$ 一定不会增加，所以由归纳很容易地知道 $\hat{f}_i \leq f_i$ 对所有的 $i = 1, 2, \cdots, n$ 均成立。接下来证明 $\hat{f}_i \geq f_i - \frac{m}{k+1}$，也就是证明 $f_i - \hat{f}_i \leq \frac{m}{k+1}$

设 $\alpha_i = f_i - \hat{f}_i$。数据流初始化没有流入任何数的时候，有 $\alpha = 0$。当流入第 j 个数 i_j 的时候

- 如果 $i_j = i$，且当前步骤中 $\hat{f}_i$ 自增了 1，那么这对应上述的第 1 种或者第 2 种 *自增* 情况，α_i 不会变化
- 如果 $i_j = i$，且当前步骤中 $\hat{f}_i$ 没有变化，那么这对应上述的第 3 种 *全部自减* 情况，α_i 增加 1
- 如果 $i_j \neq i$，且当前步骤中 $\hat{f}_i$ 没有变化，那么很高兴地 α_i 也不会变化
- 如果 $i_j \neq i$，且当前步骤中 $\hat{f}_i$ 自减了 1，那么这一定是上述的第 3 种 *全部自减* 情况导致的，α_i 增加 1

> 如果 $i_j \neq i$ 且 $\hat{f}_i$ 没有变化，这不一定意味着本步骤不是 *全部自减*——可能当前 $\hat{f}_i = 0$，也就是 $\hat{f}_i$ 减不动了

设 *全部自减* 发生了 l 次，从上述证明种我们可以看到每次 α_i 增加 1 都是全部自减导致的，所以 $\alpha_i \le l$

每一次 *自增* 会让 $\sum \hat{f}_j$ 增加 1，每次 *全部自减* 会让 $\sum \hat{f}_j$ 减少 k

全部自减 发生了 l 次，数据流种总共流入了 m 个数，所以 *自增* 发生了 $m - l$ 次，*自增* 总共让 $\sum \hat{f}_j$ 增加了 $m - l$，而 *全部自减* 让 $\sum \hat{f}_j$ 减少了 kl

所以算法结束的时候 $\sum \hat{f}_j = m - l - kl = m - (k+1)l$

因为算法的任意时刻对于任意的 j 都有 $\hat{f}_j \ge 0$，所以算法结束的时候 $m - (k+1)l = \sum \hat{f}_j \ge 0$，也就是 $l \le \frac{m}{(k+1)l}$

这就表明对于任意的 i，均有 $f_i - \hat{f}_i = \alpha_i \le l \le \frac{m}{(k+1)l}$。证毕

设 $k+1$ 众数集合 $\mathrm{HH}_{k+1}(S) = \{j \mid f_j > \frac{m}{k+1}\}$，Misra Gries 算法输出的集合 H 一定满足 $|H| = k$ 且 $H \supseteq \mathrm{HH}_{k+1}$，也就是说 H 是 HH_{k+1} 一个不太大的超集。但是我们不知道 H 中到底有哪些元素不属于 HH_{k+1}。我们目前也没有设计出任何只让数据流流过一次就能求出 HH_{k+1} 的算法

Misra Gries 算法不能处理元素删除。传统数据流确实只有插入操作，但是接下来我们会介绍这种带删除的数据流

Turnstile Streaming Model

> 不太能翻译这个名词。这个名字的由来是：传统的计数模型就是维护一个水缸，来的数据相当于往某个水缸中倒一点水；这个模型不光能把水给倒到水缸里，还能控制水缸里的水流出来，起到了河道里闸门 (Turnstile) 的作用，水可以双向流。或许可以翻译成 *闸门数据流模型*

在经典的数据流模型中，我们会流入一列数据 $i_1, i_2, \cdots, i_m$，并维护这列数据的一些统计信息。设 $\vec{x}$ 满足 x_i 是数据流中所有等于 i 的元素出现的个数，先前的算法实际上都是维护关于 $\vec{x}$ 的信息

但有的时候我们希望能删除先前输入的一些数据。我们在流入 $i_j \in \{1, 2, \cdots, n\}$ 的时候，同时流入一个辅助变量 $\Delta_j \in \mathbb{R}$。流入 (i_j, Δ_j) 意味着令 $x_{i_j} \leftarrow x_{i_j} + \Delta_j$

Turnstile Streaming Model 不对 Δ_j 进行任何限制，只要 $\Delta_j \in \mathbb{R}$ 就行。下面列出一些特殊的闸门数据流模型：

- 传统数据流模型：$\Delta_j = 1$
- 收银机模型：$\Delta_j > 0$。收银机当然不会对顾客吐出钱
- Strict Turnstile (Streaming) Model：Δ_j 可正可负，需满足但是任意时刻 $\vec{x} \in \mathbb{R}^n_{\ge 0}$（也就是 $\Delta_j \ge -x_{i_j}$）
- 图流模型：$\Delta_j = \pm 1$，且任意时刻 $\vec{x} \in \mathbb{N}^n$。这时 $\{1, 2, \cdots, n\}$ 对应图中潜在边的集合，也就表明动态图问题中可能没有某条边或者关于某条边有很多重边，但是不可能某条边有负数条

推广的众数查询

(k, l_1) 单点查询问题指：任意时刻询问任意的 $i \in \{1, 2, \cdots, n\}$，需要输出 $\tilde{x}_i = x_i \pm \frac{1}{k}\|X\|_1$

(k, l_1) 众数查询问题指：任意时刻进行询问，需要输出 $L \subseteq \{1, 2, \cdots, n\}$ 满足：

- $|L| = \mathcal{O}(k)$

- 如果 $|x_i| > \frac{1}{k}\|X\|_1$，那么 $i \in L$

> Misra Gries 算法就是在传统数据流模型下解决以上问题的一个算法

引理：设 $\mathcal{A}$ 是一个 $(3k, l_1)$ 单点查询问题的算法，其失败概率不超过 $\frac{\delta}{n}$，且消耗空间为 s 比特。那么可以构造出一种解决 (k, l_1) 众数查询问题的算法 $\mathcal{A}'$，失败概率不超过 δ，消耗 $s + \mathcal{O}(k \log n)$ 比特

证明：

> 这个算法简单点说就是：遍历所有元素的出现次数，并找出出现最多的几个。注意到单点查询在这里并不是一个确定算法（因为在数据量很多，数据范围不小的前提下，将所有数记录下来就需要 $n \log n$ 个比特，是一个很大的负担），所以这里不能只找 k 个，顺手多找几个

在 $\mathcal{A}$ 算法的基础上我们这样构造 $\mathcal{A}'$：查询时，遍历 $i = 1, 2, \cdots, n$，使用 $\mathcal{A}$ 算法依次单点查询这 n 个点的值，同时记住前 $3k$ 大的数并输出

我们证明所有出现次数不低于 $\frac{1}{k}$ 的数都被输出了。因为一个单点查询失败的概率不超过 $\frac{\delta}{n}$，所以所有单点查询失败至少一次的概率至多为 δ，这些单点查询全部成功的概率至少为 $1 - \delta$

> 设第 i 个单点查询失败的事件为 A_i，这 $\Pr[A_i] \le \frac{\delta}{n}$，所以
> $\Pr\left[\bigcup_{i=1}^n A_i\right] \le \sum_{i=1}^n \Pr[A_i] \le n \cdot \frac{\delta}{n} = \delta$

在这些单点查询全部成功的前提下，每个返回的 $\tilde{x}_i$ 均在 $\left[x_i - \frac{1}{3k} \| x \|_1, x_i + \frac{1}{3k} \| x \|_1\right]$ 区间内

- 如果 $x_i > \frac{1}{k} \| x \|_1$，那么 $\tilde{x}_i > \frac{1}{k} \| x \|_1 - \frac{1}{3k} \| x \|_1 = \frac{2}{3k} \| x \|_1$
- 如果 $x_i \le \frac{1}{3k} \| x \|_2$，那么 $\tilde{x}_i \le \frac{1}{3k} \| x \|_1 + \frac{1}{3k} \| x \|_1 = \frac{2}{3k} \| x \|_1$

以上内容表明：所有 HH_k 中的元素 i 都满足 $\tilde{x}_i > \frac{2}{3k} \| x \|_1$，所有满足 $\tilde{x}_i > \frac{2}{3k} \| x \|_1$ 的元素 i 都满足 i 是 HH_{3k} 中的元素。HH_{3k} 中的元素至多有 $3k$ 个，所以满足 $\tilde{x}_i > \frac{2}{3k} \| x \|_1$ 的元素个数一定不超过 $3k$ 个。我们找出前 $3k$ 大的数一定包含所有满足 $\tilde{x}_i > \frac{2}{3k} \| x \|_1$ 的元素，所以也一定包含 HH_k 中的元素

Count Min Sketch

> 同不太能翻译出来。这里 Sketch 表示草稿，也就是说我们并不完整刻画这一整个数据结构，而是刻画它最显著最重要的特征。这个数据结构的本意是为了统计数据流中每个项的出现频率，目的是为了计数 (Count)；而本数据结构使用了求最小值的技巧，为了和另一个使用了中位数技巧的数据结构区分开，在这里加一个 Min 以用于标识。或许可以翻译成 *计数最小草稿*

先前的频繁项查询算法需要依赖于单点查询算法。本节我们将针对 Strict Turnstile Streaming Model 来给出一个 (k, l_1) 单点查询算法。回忆一下 Strict Turnstile Streaming Model，该模型要求输入满足任意时刻 X 中的所有元素都是非负

先给出算法。设 w, d 是两个参数，我们后面再对其选择：

1. 选择 d 个 2 元独立随机哈希函数 $h_1, h_2, \cdots, h_d$，它们将 $\{1, 2, \cdots, n\}$ 均匀地散列到 $\{1, 2, \cdots, w\}$ 上
2. 维护一个二维数组 C，初始化 $C_{i,j} = 0$
3. 当数据流中流入 $e_t = (i_t, \Delta_t)$ 的时候，遍历每个哈希函数 $h_1, h_2, \cdots, h_d$。当遍历到函数 h_l 的时候，更新 $C_{l,h_l(i_t)} \leftarrow C_{l,h_l(i_t)} + \Delta_t$
4. 记 $\tilde{x}_i \triangleq \min_{l=1}^d C_{l,h_l(i)}$，作为查询 x_i 时的输出结果

引理：在 Strict Turnstile Model 下，取 $d \ge \log \frac{1}{\delta}$ 与 $w > 2k$，则

$$\Pr\left[\tilde{x}_i \ge x_i + \frac{\|X\|_1}{k}\right] \le \delta, i = 1, 2, \cdots, n$$

证明：由定义知 $\tilde{x}_i = \min_{l=1}^{d} C_{l,h_l(i)}$。所以我们顺次考虑这 d 个数 $C_{1,h_1(i)}, C_{2,h_2(i)}, \cdots, C_{d,h_d(i)}$

首先注意到 $C_{l,j} = \sum_{h_i(k)=j} x_k$，这是因为 $h_l(k) = j$ 表明数据流中流入与 x_k 相关的更新 (k, Δ) 的时候，在二维数组的第 l 行实际上只更新了 $C_{l,h_l(k)} = C_{l,j}$ 的元素

所以 $C_{l,h_l(i)} = x_i + \sum_{i' \neq i, h_l(i')=h_l(i)} x_{i'}$

在 Strict Turnstile Model 下，$x_{i'} \ge 0$，所以 $C_{l,h_l(i)} \ge x_i$

考虑到 h_l 是 2 元独立哈希函数，所以

$$\mathbb{E}\left[C_{l,h_l(i)}\right] = x_i + \sum_{i' \neq i} \Pr\left[h_l(i') = h_l(i)\right] x_{i'} = x + \sum_{i' \neq i} \frac{1}{w} x_i \le x_i + \frac{1}{w} \sum_{i' \neq i} |x_i| < x_i + \frac{\|X\|_1}{2k}$$

> 第一个等式成立是因为：当 $h_l(i') = h_l(i)$ 的时候，$x_{i'}$ 的值会贡献到 $C_{l,h_l(i)}$ 中——也就是说有 $\Pr[h_l(i') = h_l(i)]$ 的概率 $x_{i'}$ 对 $C_{l,h_l(i)}$ 产生贡献，有 $\Pr[h_l(i') \neq h_l(i)]$ 的概率没有对 $C_{l,h_l(i)}$ 产生任何贡献

这表明 $\mathbb{E}\left[C_{l,h_l(i)} - x_i\right] \le \frac{\|X\|_1}{2k}$，而且 $C_{l,h_l(i)} - x_i \ge 0$。由马尔科夫不等式，

$$\Pr\left[C_{l,h_l(i)} - x_i > \frac{\|X\|_1}{k}\right] \le \frac{\mathbb{E}\left[C_{l,h_l(i)} - x_i\right]}{\|X\|_1 / k} < \frac{1}{2}$$

因为这些哈希函数是均匀随机选择的，所以

$$\Pr\left[\tilde{x}_i \ge x_i + \frac{\|X\|_1}{k}\right] = \Pr\left[\bigwedge_{l=1}^{d} C_{l,h_l(i)} \ge x_i + \frac{\|X\|_1}{k}\right] = \prod_{l=1}^{d} \Pr\left[C_{l,h_l(i)} \ge x_i + \frac{\| x \|_1}{k}\right] < \left(\frac{1}{2}\right)^d \le \delta$$

当 $d = \Theta(\ln n)$ 的时候，也就是 $\delta = n^{\Theta(1)}$ 的时候，算法的正确概率就能相当高。所以 Count Min Sketch 只需要 $\Theta(k \log n)$ 个计数器就可以完成 (k, l_1) 单点查询

Count Sketch

Count Min Sketch 算法中空间复杂度大约是关于 k 的线性关系，这是因为 X 中比 $\frac{1}{k}\|X\|_1$ 大的元素少于 k 个。但是 l_2 范数就没有 l_1 范数这么好的性质了，比 $\frac{1}{k}\|X\|_2$ 大的 X 中元素可能达到将近 k^2 个。不过，还是可以使用类似 Count Min Sketch 的算法与分析来求解 l_2 范数中的问题

1. 选择 d 个 2 元独立随机哈希函数 $h_1, h_2, \cdots, h_d$，它们将 $\{1, 2, \cdots, n\}$ 均匀地散列到 $\{1, 2, \cdots, w\}$ 上；再选择 d 个 2 元独立随机哈希函数 $g_1, g_2, \cdots, g_d$，它们将 $\{1, 2, \cdots, n\}$ 均匀地散列到 $\{-1, 1\}$ 上
2. 维护一个二维数组 C，初始化 $C_{i,j} = 0$
3. 当数据流中流入 $e_t = (i_t, \Delta_t)$ 的时候，遍历每个哈希函数 $h_1, h_2, \cdots, h_d$。当遍历到函数 h_l 的时候，更新 $C_{l,h_l(i_t)} \leftarrow C_{l,h_l(i_t)} + g_l(i_t)\Delta_t$
4. 记 $\tilde{x}_i \triangleq \text{median}\{g_l(i) C_{l,h_l(i)}\}_{l=1,2,\cdots,d}$，作为查询 x_i 时的输出结果

引理：在 Strict Turnstile Model 下，取 $d \ge 18 \ln \delta$ 与 $w > 3k^2$，则

$$\Pr\left[\tilde{x}_i \ge x_i + \frac{\|X\|_1}{k}\right] \le \delta, i = 1, 2, \cdots, n$$

证明：首先固定一组 $l \in \{1,2,\cdots,d\}, i \in \{1,2,\cdots,n\}$，我们记 $Z_l = g_l(i)C_{l,h_l(i)}$。同时对于每一个 $i' = 1,2,\cdots,n$，记 $Y_{i'} = [h_l(i) = h_l(i')]$

因为 $Y_{i'}$ 的取值只能是 0 或 1，所以 $\mathbb{E}[Y_{i'}] = \mathbb{E}\left[Y_{i'}^2\right] = \Pr[Y_i = 1] = \Pr[h_l(i) = h_l(i')]$。在前一问中，我们已经分析过当 $i' \neq i$ 的时候 $\Pr[h_l(i) = h_l(i')] = \frac{1}{w}$。所以

$$Z_l = g_l(i)C_{l,h_l(i)} = g_l(i)\left(g_l(i)x_i + \sum_{i'\neq i}\left[h_l(i) \neq h_l(i')\right]g_l(i')x_{i'}\right) = x_i + \sum_{i'\neq i} Y_{i'}g_l(i)g_l(i')x_{i'}$$

再联想到 $Y_{i'}$ 仅与 h 函数相关，与 g 函数是无关的。同时考虑到 g_l 是 2 元独立随机哈希函数，$g_l(i)$ 与 $g_l(i')$ 是独立的。所以：

$$\mathbb{E}[Z_l] = x_i + \sum_{i'\neq i}\mathbb{E}\left[Y_{i'}g_l(i)g_l(i')x_{i'}\right] = x_i + \sum_{i'\neq i}\mathbb{E}[Y_{i'}]\mathbb{E}[g_l(i)]\mathbb{E}\left[g_l(i')\right]x_{i'} = x_i$$

注意到 $i' \neq i''$ 的时候 $g_l(i')$ 与 $g_l(i'')$ 是独立的，而 $\left(g_l(i)\right)^2 = 1$ 一定成立，因为 g 函数的值域是 $\{-1,1\}$。同时 $Y_{i'}Y_{i''}$ 的值仅与 f 相关，与它们都独立。所以 $i' \neq i''$ 时 $\mathbb{E}\left[Y_{i'}Y_{i''}(g_l(i))^2 g_l(i')g_l(i'')\right] = \mathbb{E}[Y_{i'}Y_{i''}]\mathbb{E}[g_l(i')]\mathbb{E}[g_l(i'')] = 0$。所以

$$\begin{aligned}
\operatorname{Var}[Z_l] &= \mathbb{E}\left[(Z_l - x_i)^2\right] \\
&= \mathbb{E}\left[\sum_{i'\neq i}\sum_{i''\neq i} Y_{i'}Y_{i''}(g_l(i))^2 g_l(i')g_l(i'')x_{i'}x_{i''}\right] \\
&= \sum_{i'\neq i}\mathbb{E}\left[Y_{i'}^2(g_l(i))^2\left(g_l(i')\right)^2 x_{i'}^2\right] \\
&= \sum_{i'\neq i} x_{i'}^2\mathbb{E}\left[Y_{i'}^2\right] = \sum_{i'\neq i}\frac{x_{i'}^2}{w} \leq \frac{\|X\|_2}{w}
\end{aligned}$$

由切比雪夫不等式，

$$\mathbb{P}\left[|Z_i - x_i| > \frac{\|X\|_2}{k}\right] \leq \frac{\operatorname{Var}(Z_i)}{(\|X\|_2/k)^2} \leq \frac{k^2}{w} \leq \frac{1}{3}$$

请读者仿照 Morris++、FM++ 算法中的中位数技巧完成后续证明

这表明当 $d = \Theta(\log n)$, $w = \Theta(k^2)$ 时就可以以很高的概率解决 2 单点查询问题，需要 $\Theta(k^2 \log n)$ 个计数器。假设每个计数器的值都不超过 M（$\|X\|_1$ 是它的一个上界），那么它的空间复杂度就是 $\mathcal{O}\left(k^2 \log n \log M\right)$

Count Min Sketch 和 Count Sketch 的应用

区间询问

在一个 Strict Turnstile Streaming Model 中，每次询问一对数 i, j，需要回答 $\sum_{l=i}^{j} x_l$，也就是下标在区间 $[i, j]$ 内的元素和

当然，严格求解这个问题的第一步起码得把这些数都给存下来，这样空间复杂度就会达到大约 $\mathcal{O}\left(m \log M\right)$ 了（M 与上文一样，是 Count Min Sketch 算法中计数器的最大值），在 $\log n$ 与 $\log m$ 的意义下一定是一个指数复杂度，这是我们不可接受的

所以我们将目标放松为：输出某个 $\tilde{x}_{ij}$ 满足 $\left|\tilde{x}_{ij} - \sum_{l=i}^{j} x_l\right| \leq \epsilon \| x \|_1$

这个放松的目的已经非常明显了——我们想用 Count Min Sketch 来求解这个问题。可不可以选择某个 k，直接对 $x_i, x_{i+1}, \cdots, x_j$ 进行 (k, l_1) 单点查询呢？因为每次查询 $\tilde{x}_i$ 都会带来 $\frac{1}{k} \| x \|_1$ 的误差，所以直接这样进行 $j-i+1$ 次查询了之后，累计误差可能达到 $(j-i+1) \cdot \frac{1}{k} \| x \|_1$。为了保证在最坏情况下算法依然相对误差不超过 ϵ，而最坏情况下 $j-i+1$ 可以高达 m，我们选择的 k 必须满足 $\frac{m}{k} \le \epsilon$，也就是 $k \ge \frac{m}{\epsilon}$。回忆 Count Min Sketch 需要维护 $\Theta(k \log n)$ 个计数器，使用这种方法空间复杂度就能达到 $\Theta(\frac{m}{\epsilon} \log n)$，甚至比暴力还差了

这里使用了一种名为线段树的数据结构的思想。我的文字不能展示它的美感，这里给一个链接：https://blog.csdn.net/litble/article/details/72486558（只用到了前两节单点修改区间查询的方法）

线段树是一个树形结构，而 Count Sketch 问题只能存储一个数组——我们不能将指针也一并用 Count Sketch 维护。解决的方法是使用类似非递归线段树的思想将线段树拍扁。思想上是将线段树的根节点（深度为 0 的结点）存在数组的第 1 个位置，深度为 1 的两个结点存储在数组的第 2 到第 3 个位置，深度为 2 的 4 个结点存储在数组的第 4 到第 7 个位置，以此类推。形式化地说，我们不妨假设 n 是 2 的幂，并建立数组 $\{t_i\}_{i=1,2,\cdots,2n-1}$ 来存储这这棵线段树，其中 $t_1 = t_{(1)_2}$ 存储 $x_{[1,n]}$，也就是线段树的根节点；$t_2 = t_{(10)_2}$ 与 $t_3 = t_{(11)_2}$ 分别存储 $x_{[1,n/2]}$ 与 $x_{[n/2+1,n]}$，是 $t_{(1)_2} = x_{[1,n]}$ 的两个孩子；$t_4 = t_{(100)_2}$ 与 $t_5 = t_{(101)_2}$ 分别存储 $x_{[1,n/4]}$ 与 $x_{[n/4+1,n/2]}$，是 $t_{(10)_2}$ 的两个孩子……这样就可以通过观察下标来得出计算 $x_{[l,r]}$ 时应该对 t 数组中的哪些数求和

> 为什么不妨假设 n 是 2 的幂？因为 $[n, 2n)$ 中一定有一个 2 的幂。如果 n 不是 2 的幂，那么我们将 n 增加到离它最近的 2 的幂，这只会多维护不超过一倍的节点，不会影响算法的渐进复杂度
>
> 除了使用非递归线段树的思想以外，也可以直接使用树状数组来解决本题的问题

在 Strict Turnstile Model 中，观察到 $\left\|\vec{T}\right\|_1 = \log n \left\|\vec{x}\right\|_1$，这是因为每个点 x_i 的值除了贡献在叶子结点 $x_{[i,i]}$ 中以外，还贡献在了它的父亲 $\begin{cases} x_{[i,i+1]} & 2 \mid i \\ x_{[i-1,i]} & 2 \nmid i \end{cases}$ 中，还贡献在了它的祖父（$x_{[i,i+3]}$ 或 $x_{[i-1,i+2]}$ 或 $x_{[i-2,i+1]}$ 或 $x_{[i-3,i]}$）中，依次类推，每个祖先都贡献了一遍，总共贡献了 $\log n$ 层

同时观察到一次查询需要对 $\mathcal{O}(\log n)$ 个数进行求和，所以使用 Count Min Sketch 算法维护线段树的时候，我们只需要选择 k 满足 $\frac{\mathcal{O}(\log n)}{k} \le \frac{\epsilon}{\log n}$ 即可，也就是 $k = \mathcal{O}\left(\frac{\log^2 n}{\epsilon}\right)$ 即可。空间复杂度是 $\mathcal{O}(k \log n \log M) = \mathcal{O}\left(\frac{1}{\epsilon} \log^3 n \log M\right)$

稀疏恢复

我们有的时候需要存储一个稀疏的数组——数组中的大部分元素都很小，很接近 0，只有少数几个元素有一些比较大的值，例如在统计牛津字典中每个词在某本书中出现次数的时候

我们这样形式化地定义稀疏化的过程：给定某个 $\vec{x} \in \mathbb{R}^n$ 的向量、某个正整数 k 与某个范数 $\| \cdot \|_p$，我们需要找到一个向量 $\vec{z}$，使得 $\vec{z}$ 中的非零元素不超过 k 个（也就是 $\left\|\vec{z}\right\|_0 \le k$），而且 $\left\|\vec{x} - \vec{z}\right\|_p$ 尽可能小

假如我们将 $\vec{x}$ 给严格完整地存储了下来，那么有一种非常简单的做法：选择 $\vec{x}$ 中绝对值最大的 k 个元素，$\vec{z}$ 在这些位置与 $\vec{x}$ 相同，其他位置为 0。当然，将 $\vec{x}$ 严格存储下来需要 $\mathcal{O}(m \log M)$ 的空间，但是我们受到这种离线算法与 Count Min Sketch 的启发，设计这样的在线算法（在 Strict Turnstile 的模型下）：

1. 取 $w = \frac{3k}{\epsilon^2}$ 与 $d = \Omega(\log n)$ 作为 Count Sketch 算法的参数
2. 使用 Count Sketch 算法估计出 $\tilde{X} = \{\tilde{x}_i\}_{i=1,2,\cdots,n}$ 的值
3. 输出 $\vec{z}$，满足 $\vec{z}$ 在 $\tilde{X}$ 绝对值最大 k 个元素的地方与 $\tilde{X}$ 相同，其他位置为 0

显然这个算法的空间复杂度是 $\mathcal{O}\left(\frac{1}{\epsilon^2} k \log n \log M\right)$ 的（M 是 Count Sketch 算法中计数器的最大可能值）。下面描述它给出的解的质量

首先，即便是最好的解也会有一定的误差。我们取误差度量范数 $\|\cdot\|_p$ 为 2 范数，记最好解的误差是 $\text{err}_2^k(\vec{x}) = \min_{\vec{z},\|\vec{z}\|_0\le k}\|\vec{x}-\vec{z}\|_2$。我们可以证明：

定理：当 $\epsilon \le \frac{1}{\sqrt{5}}$ 的时候（当 ϵ 充分小的时候），以至少 $\frac{1}{n}$ 的概率本算法输出的 $\vec{z}$ 满足 $\|\vec{x}-\vec{z}\|_z \le (1+5\epsilon)\text{err}_2^k(\vec{x})$

> 如果需要相对误差不超过 ϵ，将算法描述中的 w 改成 $\frac{3k}{(\epsilon/5^2)} = \frac{75k}{\epsilon^2}$ 即可

为了证明这个定理，我们只需证明两个引理即可：

引理 1：当 ϵ 充分小的时候，以至少 $1-\frac{1}{n}$ 的概率，Count Sketch 算法给出的 $\tilde{X}$ 能保证对于每一个 $i=1,2,\cdots,n$ 都有 $|\tilde{x}_i - x_i| \le \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)$

引理 2：如果 $\vec{x}$ 与 $\vec{y}$ 满足 $\|\vec{x}-\vec{y}\|_\infty \le \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(\vec{x})$，设 T 是 $\vec{y}$ 中最大的 k 个元素对应的下标，设向量 $\vec{z}$ 满足 $z_i = \begin{cases} Y_i & i\in T \\ 0 & \text{otherwise}\end{cases}$，则 $\|\vec{x}-\vec{z}\| \le (1+5\epsilon)\text{err}_2^k(\vec{x})$

> 回忆无穷范数 $\|\vec{p}\|_\infty$ 表示 $\vec{p}$ 中绝对值最大的数。引理 1 成立表示 $\|\vec{x}-\tilde{X}\|_\infty \le \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(\vec{x})$，而引理 2 成立时将 $\vec{y} \leftarrow \tilde{X}$ 代入就可以得到定理

引理 1 证明：

设 T_{big} 是 $\vec{x}$ 中最大的 k 个数对应的下标，T_{small} 是所有其他下标

> $\{1,2,\cdots,n\} = T_{\text{big}} \cup T_{\text{small}}$
> $T_{\text{big}} \cap T_{\text{small}} = \varnothing$
> $|T_{\text{big}}| = k$
> $x_i \ge x_j \forall i \in T_{\text{big}}, j\in T_{\text{small}}$

那么在一个最优的估计中，$\vec{z}$ 在 T_{big} 对应坐标下与 $\vec{X}$ 相等，在其他坐标下为 0，这就表明 $\text{err}_2^k(x) = \sum_{i'\in T_{\text{small}}} x_{i'}^2$

对于某个固定的 $1\le i\le n$ 与 $1\le l\le d$，有如下引理

引理 1.1：设 A_l 是存在 i' 满足 $i'\in T_{\text{big}}, i'\ne i$ 且 $h_l(i') = h_l(i)$ 的事件，则 $\Pr[A_l] \le \frac{\epsilon^2}{3}$。这是因为：记 $Y_{i'} = [h_l(i) = h_l(i')]$ 和 $Y = \sum_{i'\in T_{\text{big}}\setminus\{i\}}$，则 $\Pr[Y_{i'}] = \frac{1}{w} \le \frac{\epsilon^2}{3k}$，所以 $\mathbb{E}[Y] = \sum_{i'\in T_{\text{big}}\setminus i}\mathbb{E}[Y_i] \le \frac{\epsilon^2}{3}$，由马尔科夫不等式知命题成立

引理 1.2：设 $Y_{i'} = [h_l(i) = h_l(i')]$，也就是 i 与 i' 在第 l 个哈希函数中哈希冲突的事件，设 $Z_l' = \sum_{i'\in T_{\text{small}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_{i'}$，那么 $\Pr\left[|Z_l'| \ge \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)\right] \le \frac{1}{3}$。这是因为 $\mathbb{E}[Z_l'] = 0$ 而 $\text{Var}[Z_l'] \le \frac{\epsilon^2}{3k}$

首先，设 $Z_l = g(i)C[l,h_l(i)]$，也就是 Count Sketch 中第 l 个哈希函数的估计值。考虑到 $C[l,h_l(i)] = \sum_{h_l(i')=h_l(i)} g(i')x_{i'}$，我们有

$$Z_l = x_i + \sum_{i'\in T_{\text{big}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_{i'} + \sum_{i'\in T_{\text{small}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_{i'} = x_i + \sum_{i'\in T_{\text{big}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_{i'} + Z_l'$$

引理 1.1 保证了以至少 $1-\frac{\epsilon^2}{3}$ 的概率对于所有的 $i'\in T_{\text{big}}\setminus\{i\}$ 均有 $h(i')\ne h(i)$。$h(i')\ne h(i)$ 意味着 $Y_{i'} = 0$，所有 $i'\in T_{\text{big}}\setminus\{i\}$ 均满足 $h(i')\ne h(i)$ 就意味着 $\sum_{i'\in T_{\text{big}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_i = 0$，也就是以至多 $\frac{\epsilon^2}{3}$ 的概率有 $\sum_{i'\in T_{\text{big}}\setminus\{i\}} g_l(i)g_l(i')Y_{i'}x_i \ne 0$

引理 1.2 保证了以至多 $\frac{1}{3}$ 的概率 $|Z_l| \ge \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)$

如果 $|Z_l - x_i| \geq \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)$，那么引理 1.1 与引理 1.2 中的坏事件至少发生一件。考虑到我们假设 ϵ 充分小（$\epsilon \leq \frac{1}{5}$），所以 $|Z_l - x_i| \geq \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)$ 的概率至多为 $\frac{\epsilon^2}{3} + \frac{1}{3} \leq \frac{2}{5}$

请读者使用中位数的技巧完成剩余证明

引理 2 证明：

设 $\vec{x}$ 中前 k 大元素下标的集合是 S，$\vec{y}$ 中前 k 大元素对应的下标是 T，则由 $\vec{z}$ 仅在 T 中对应位置上与 $\vec{y}$ 相等，其余位置均为 0 可以得知

$$\|\vec{x}-\vec{z}\|_2^2 = \sum_{i\in T}|x_i - y_i|^2 + \sum_{i\in S\setminus T} x_i^2 + \sum_{i\in\{1,2,\cdots,n\}\setminus(S\cup T)} x_i^2 = \sum_{i\in T}|x_i - y_i|^2 + \sum_{i\in S\setminus T} x_i^2 + \sum_{i\in\{1,2,\cdots,n\}\setminus(S\cup T)} x_i^2$$

考虑到 $\|\vec{x}-\vec{y}\|_\infty \leq \frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)$，得到 $|x_i - y_i| \leq \|\vec{x}-\vec{y}\|_\infty$，再考虑到 T 中只有 k 个元素（因为这是 $\vec{y}$ 中的前 k 大元素），所以

$$\sum_{i\in T}|x_i - y_i|^2 \leq k\left(\frac{\epsilon}{\sqrt{k}}\text{err}_2^k(x)\right)^2 = \epsilon^2\text{err}_2^k(x)$$

关于 $\sum_{i\in S\setminus T}|x_i - z_i|^2$ 与 $\sum_{i\in\{1,2,\cdots,n\}\setminus(S\cup T)} x_i^2$ 的上界估计需要用很多三角不等式的技巧，能估算上界的原理是 $\vec{x}$ 与 $\vec{y}$ 相差不太大，详见彭老师的讲义

Matrix Sketch

> 我很想将这个名词翻译成 *矩阵草图*，因为它就是为矩阵画一个速写，将它最重要的部分给突出出来。但是为了与前文一致（前文中的 Sketch 我全都没有翻译），为了突出知识的相关性，我这里暂不翻译

对于一个矩阵 $A \in \mathbb{R}^{n\times d}$，我们希望找到一个矩阵 $B \in \mathbb{R}^{k\times d}$ 满足 $A^\intercal A \approx B^\intercal B$，这样 B 还能保持 A 的一些信息，比如奇异值和右奇异空间

如果 $A^\intercal A \approx B^\intercal B$，那么 $x^\intercal A^\intercal Ax \approx x^\intercal B^\intercal Bx$，也就是 $\|Ax\|_2 \approx \|Bx\|_2$

更准确地说，我们希望我们找到的 B 满足 $0 \leq \|A\vec{x}\|_2^2 - \|B\vec{x}\|_2^2 \leq \frac{2}{k}\|A\|_F^2$，也就是

- $\|A^\intercal A - B^\intercal B\|_2 \leq \frac{2}{k}\|A\|_F^2$
- $B^\intercal B \preceq A^\intercal A$

> 回忆 $A \preceq B$ 表示 A 与 B 都是实对称矩阵，且 $B - A$ 是半正定矩阵

回忆 Misra-Gries 算法给出的每个元素频率估计 $\hat{f}_j$ 满足 $\left|\hat{f}_j - f_j\right| \leq \frac{n}{k+1}$，其中 f_j 是数 j 的频率真实值；Misra-Gries 算法能输出 n 个数中出现次数超过 $\frac{n}{k+1}$ 的数。

频繁项定义可以这样等价地转化：数据流中流入的不是一个单个的数 x_i，而是第 i 个标准正交基 $\hat{e}_{x_i}$。我们把这 n 个向量（转置后）顺着排成一个 n 行的矩阵 A，那么数 j 的频率 $f_j = \|A\hat{e}_j\|_2$。矩阵 A 中的每一行只有一个 1 其他全是 0，说明矩阵的所有元素的平方和是行数 n，也就是 $\|A\|_F^2 = n$。Misra-Gries 算法求解出来了每个向量出现频率的估计值，只有 k 个向量的估计值最后存在数表中，可能非零。将这些估计值数乘上对应的单位向量，顺次排列得到矩阵 B，那么 Misra-Gries 能保证 $\|B\hat{e}_j\|_2 = \hat{f}_j$，与 $\|A\hat{e}_j\| = f_j$ 的绝对误差不超过 $\frac{n}{k+1}$

> 举个例子。假如数据流是 $(4,3,2,1,1,3,1,1,1)$，那么原矩阵是

> $$A=\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\1&0&0&0\\1&0&0&1\\0&0&0&1\\1&0&0&0\\1&0&0&0\\1&0&0&0\end{bmatrix}$$
>
> 运行 $k=2$ 的 Misra-Gries 算法，得到的结果是
>
> $$I=\begin{bmatrix}1\\3\end{bmatrix}, c=\begin{bmatrix}5\\1\end{bmatrix}, B=[5\hat{e}_1 \quad 1\hat{e}_3]^\intercal=\begin{bmatrix}5&0&0&0\\0&0&1&0\end{bmatrix}$$
>
> 这个时候 B 就是 A 的一个很好的低维估计

我们推广传统的 Misra-Gries 算法来设计一个新算法以解决 Matrix Sketch 问题：

> 回忆一下 SVD 中的内容。对于任意的 $B\in\mathbb{R}^{k\times d}$，都存在 B 的奇异值分解
> $B=\sum_{i=1}^{r}\sigma_i\vec{u}_i\vec{v}_i^\intercal=\sum_{i=1}^{k}\sigma_i\vec{u}_i\vec{v}_i^\intercal=U\Sigma V^\intercal$（其中 $\sigma_1\ge\cdots\ge\sigma_r\ge\sigma_{r+1}=\cdots=\sigma_k=0$, $U\in\mathbb{R}^{k\times k}$, $\Sigma=\operatorname{diag}(\sigma_1,\sigma_2,\cdots,\sigma_k)$, $V\in\mathbb{R}^{k\times d}$, $U^\intercal U=V^\intercal V=I_k$）

1. 初始化 $B=O_{n\times d}$ 是一个零矩阵
2. 依次处理 $\vec{a}_1,\vec{a}_2,\cdots,\vec{a}_n$。当处理到 $\vec{a}_i$ 的时候：

   断言此时 B 中一定有某一行全是 0。将 $\vec{a}_i^\intercal$ 插入 B 的某个全是 0 的行中

   如果插入 $\vec{a}_i^\intercal$ 之后 B 的所有行都不是 $\vec{0}_d^\intercal$，那么：

   - 计算 B 的奇异值分解 $B\to U\Sigma V^\intercal$，令 $C\leftarrow\Sigma V^\intercal$, $\delta\leftarrow\sigma_{k/2}^2$（$C$ 仅供算法分析使用）
   - 令 $\tilde{\Sigma}\leftarrow\sqrt{\max(\Sigma^2-I_k\delta,0)}$
   - 令 $B\leftarrow\tilde{\Sigma}V^\intercal$
3. 返回 B

命题：本算法输出的 B 满足 $0\preceq B^\intercal B\preceq A^\intercal A$

证明：

首先，$\vec{x}^\intercal B^\intercal B\vec{x}=(B\vec{x})^\intercal(B\vec{x})\ge 0$ 对任意 $\vec{x}\in\mathbb{R}^d$ 成立

接下来我们尝试证明 $\vec{x}^\intercal(A^\intercal A-B^\intercal B)\vec{x}=\left\|A\vec{x}\right\|_2^2-\left\|B\vec{x}\right\|_2^2\ge 0$ 对任意的 $\vec{x}\in\mathbb{R}^d$ 成立。归纳：

设 $B^{(i)}$ 与 $C^{(i)}$ 依次表示 B 与 C 在算法第 i 次迭代（处理 A_i）时的向量。显然 $B^{(0)}=O, B^{(n)}=B$

> 如果某次插入之后 B 中某一行还全是 0，那么这次插入前后 $B^\intercal B$ 相较于 $A^\intercal A$ 无变化。更形式化地说，$\sum_{j=1}^{i}\vec{a}_j\vec{a}_j^\intercal-B^{(i)\intercal}B^{(i)}=\sum_{j=1}^{i-1}\vec{a}_j\vec{a}_j^\intercal-B^{(i-1)\intercal}B^{(i-1)}$，所以这里只讨论插入了 $\vec{a}_i$ 导致 B 满了的情况

首先观察到 $U^\intercal C^{(i)}$ 是 $B^{(i-1)}$ 的某个 $\vec{0}^\intercal$ 行替换为 $\vec{a}_i^\intercal$ 的结果，所以 $U^\intercal C^{(i)}\vec{x}$ 是 $B^{(i-1)}\vec{x}$ 的某个 0 元被替换为 $\vec{a}_i^\intercal\vec{x}$ 的结果。由勾股定理可知 $\left(\vec{a}_i^\intercal x\right)^2+\left\|B^{(i-1)}\vec{x}\right\|_2^2=\left\|U^\intercal C^{(i)}\vec{x}\right\|_2^2$，由 $U\in\mathbb{R}^{k\times k}$ 是正定阵知 $\left\|U^\intercal C^{(i)}\vec{x}\right\|_2^2=\left\|C^{(i)}\vec{x}\right\|_2^2$

然后注意到 $\left\|C^{(i)}\vec{x}\right\|_2^2-\left\|B^{(i)}\vec{x}\right\|_2^2\ge 0$，因为由算法的流程可以知道 $D^2\succeq\tilde{D}^2$

所以

$$
\begin{aligned}
\|A\vec{x}\|_2^2 - \|B\vec{x}\|_2^2 &= \sum_{i=1}^{n}\left(\langle \vec{a}_i, \vec{x}\rangle + \left\|B^{(i-1)}\vec{x}\right\|_2^2 - \left\|B^{(i)}\vec{x}\right\|_2^2\right) \\
&= \sum_{i=1}^{n}\left(\left\|C^{(i)}\vec{x}\right\|_2^2 - \left\|B^{(i)}\vec{x}\right\|_2^2\right) \geq 0
\end{aligned}
$$

命题： 本算法输出的 B 满足 $\|A^\intercal A - B^\intercal B\|_2 \leq \frac{2}{k}\|A\|_F^2$

证明： 设 $\vec{x}$ 是 $A^\intercal A - B^\intercal B$ 最大特征值对应的特征向量。实对称矩阵的奇异值分解与特征值分解相同，所以

$$
\begin{aligned}
\|A^\intercal A - B^\intercal B\|_2 &= \vec{x}^\intercal (A^\intercal A - B^\intercal B)\vec{x} \\
&= \|A\vec{x}\|_2^2 - \|B\vec{x}\|_2^2 \\
&= \sum_{i=1}^{n}\left(\left\|C^{(i)}\vec{x}\right\|_2^2 - \left\|B^{(i)}\vec{x}\right\|_2^2\right) \\
&= \sum_{i=1}^{n} \vec{x}^\intercal\left(C^{(i)\intercal}C^{(i)} - B^{(i)\intercal}B^{(i)}\right)\vec{x} \\
&\leq \sum_{i=1}^{n}\left\|C^{(i)\intercal}C^{(i)} - B^{(i)\intercal}B^{(i)}\right\|_2 \\
&= \sum_{i=1}^{n}\left\|D^{(i)2} - \tilde{D}^{(i)2}\right\| = \sum_{i=1}^{n}\delta^{(i)}
\end{aligned}
$$

另一方面

$$
\begin{aligned}
\|B\|_F^2 &= \sum_{i=1}^{n}\left(\left\|B^{(i)}\right\|_F^2 - \left\|B^{(i-1)}\right\|_F^2\right) \\
&= \sum_{i=1}^{n}\left[\left(\left\|C^{(i)}\right\|_F^2 - \left\|B^{(i-1)}\right\|_F^2\right) - \left(\left\|C^{(i)}\right\|_F^2 - \left\|B^{(i)}\right\|_F^2\right)\right] \\
&= \sum_{i=1}^{n}\left[\left\|\vec{a}_i\right\|_2^2 - \mathrm{tr}\left(C^{(i)\intercal}C^{(i)} - B^{(i)\intercal}B^{(i)}\right)\right] \\
&= \sum_{i=1}^{n}\left[\left\|\vec{a}_i\right\|_2^2 - \mathrm{tr}\left(D^{(i)2} - \tilde{D}^{(i)2}\right)\right] \\
&\leq \|A\|_F^2 - \sum_{i=1}^{n}\frac{k}{2}\delta^{(i)}
\end{aligned}
$$

这也就表明

$$
\sum_{i=1}^{n}\delta^{(i)} \leq \frac{2}{k}\left(\|A\|_F^2 - \|B\|_F^2\right) \leq \frac{2}{k}\|A\|_F^2
$$

由这些不等式可以直接得到结果 $\|A^\intercal A - B^\intercal B\|_2 \leq \frac{2}{k}\|A\|_F^2$

该算法仅需存储 B，所以空间复杂度为 $\mathcal{O}(dk)$ 个字。计算过程中需要进行奇异值分解，所以时间复杂度是 $\mathcal{O}(dkn)$