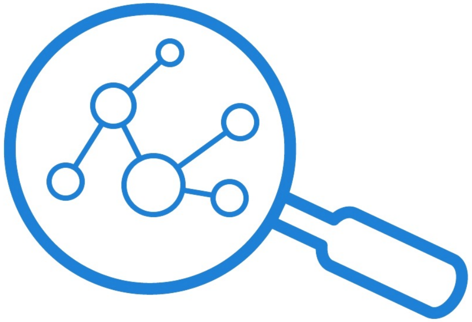
数据科学与大数据技术的数学基础

第二讲



计算机学院
余皓然

2024/4/25

课程内容

Part1 随机化方法
一致性哈希 布隆过滤器 CM Sketch方法 最小哈希 欧氏距离下的相似搜索 Jaccard相似度下的相似搜索
Part2 谱分析方法
主成分分析 奇异值分解 谱图论
Part3 最优化方法
压缩感知

布隆过滤器

从属问题

内容分发网络

Akamai（阿卡迈）搭建的内容分发网络有部署在136个国家的约30万个边缘服务器，为Apple、Microsoft等公司提供缓存服务

问题：具体应该缓存哪些文件？所有当地用户曾经请求过的文件？

内容分发网络

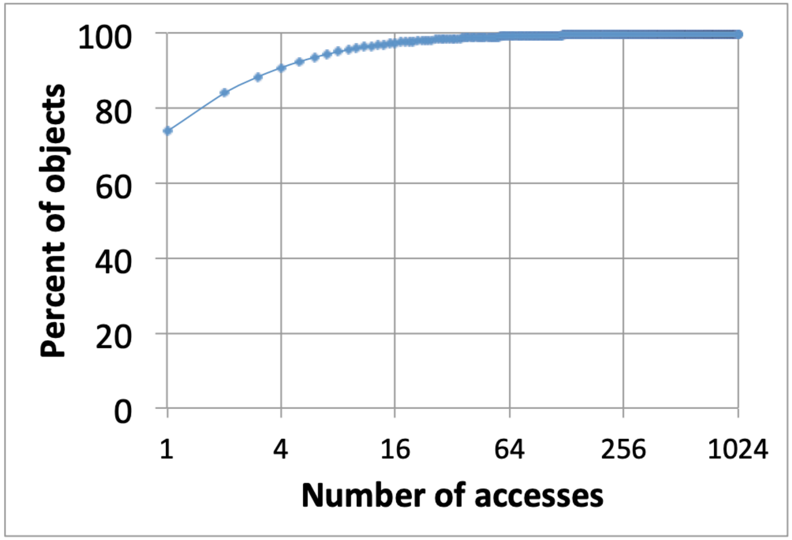


Figure 5: On a typical CDN server cluster serving web traffic over two days, 74% of the roughly 400 million objects in cache were accessed only once and 90% were accessed less than four times.

(2015年的论文）分析了Akamai的一个有45台服务器的集群，统计在两天时间内4亿份文件被用户访问的频次

说明不需要缓存所有曾经被访问的文件

图片引自B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery>

内容分发网络

一种简单的缓存策略：当用户请求访问某文件，先判断该文件是否是第一次被请求：

（1）如果是，不缓存；

（2）如果不是，缓存。

* 传输策略：如果用户请求的文件已被缓存，则从本级服务器提取文件进行传输；否则，从上级服务器中查找/提取文件进行传输

内容分发网络

一种简单的缓存策略：当用户请求访问某文件，先判断该文件是否是第一次被请求：

（1）如果是，不缓存；

（2）如果不是，缓存。

* 传输策略：如果用户请求的文件已被缓存，则从本级服务器提取文件进行传输；否则，从上级服务器中查找/提取文件进行传输

可以减少存储空间的浪费（即缓存的文件从未被用户再次请求）

如何判断文件是否是第一次被请求?

本质上需要解决“从属问题”

内容分发网络

从属判断问题（membership query）

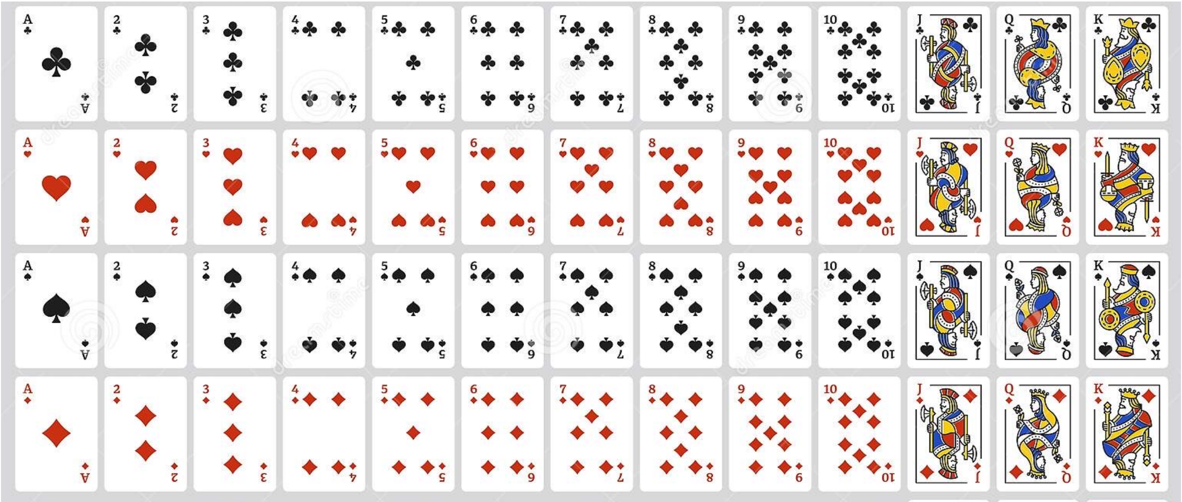
如何存储关于集合S的信息从而可以准确判断关于"元素x是否属于集S"的问题?

需要尽量减少用于存储的空间

先分析所需存储空间的下限

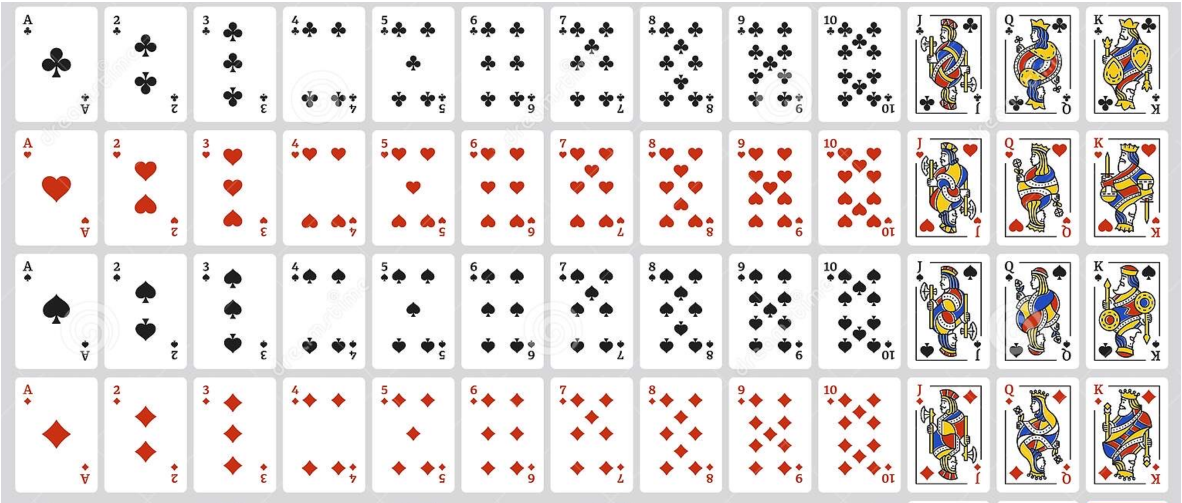
内容分发网络

已知52张牌，（不重复地）选取4张展示。至少需要多少空间（多少比特）完整存储该信息（即哪4张牌被展示了）？



内容分发网络

已知52张牌，（不重复地）选取4张展示。至少需要多少空间（多少比特）完整存储该信息（即哪4张牌被展示了）？



$$\left\lceil \log_2 \binom{52}{4} \right\rceil$$

内容分发网络

已知有$|\mathcal{U}|$个不同的物品，选取n件展示。至少需要多少空间（多少比特）完整存储该信息（即哪n件被展示了）？

$$\left\lceil \log_2 \binom{|\mathcal{U}|}{n} \right\rceil$$

内容分发网络

已知有$|\mathcal{U}|$个不同的物品，选取n件展示。至少需要多少空间（多少比特）完整存储该信息（即哪n件被展示了）？

$$\left\lceil \log_2 \binom{|\mathcal{U}|}{n} \right\rceil$$

$$\begin{aligned}
& \lg \binom{|U|}{n} \\
= \;& \lg \left(\frac{|U|\,!}{n\,!\,(|U|-n)\,!} \right) \\
\geq \;& \lg \left(\frac{(|U|-n)^n}{n^n} \right) \\
= \;& n \lg \left(\frac{|U|-n}{n} \right) \\
= \;& n \lg \left(\frac{|U|}{n} - 1 \right) \\
\geq \;& n \lg \left(\frac{|U|}{n} \right) - n \\
\geq \;& n \lg |U| - n \lg n - n \\
= \;& \boldsymbol{\Omega\left(n \lg |U| - n \lg n\right)}
\end{aligned}$$

$$\boxed{f(x) = \Omega(g(x)) \text{ as } x \to \infty \text{ if } \limsup_{x\to\infty} \left| \frac{f(x)}{g(x)} \right| > 0.}$$

推导步骤引自Keith Schwarz <Stanford CS166 - Data Structures>

内容分发网络

已知有$|\mathcal{U}|$个不同的物品，选取n件展示。至少需要多少空间（多少比特）完整存储该信息（即哪n件被展示了）？

当$|\mathcal{U}| \gg n$时，需要的存储空间至少是$\Omega(n \log_2 |\mathcal{U}|)$比特

内容分发网络

已知有$|\mathcal{U}|$个不同的物品，选取n件展示。至少需要多少空间（多少比特）完整存储该信息（即哪n件被展示了）？

当$|\mathcal{U}| \gg n$时，需要的存储空间至少是$\Omega(n\log_2|\mathcal{U}|)$比特

如果$|\mathcal{U}|$代表所有可能的网站URL、n是当地用户已经请求访问的网站URL，该数字非常大

准确记忆用户已经请求过哪些网站/文件将消耗大量的存储资源

如何以较小的存储空间解决从属问题?

内容分发网络

已知有$|\mathcal{U}|$个不同的物品，选取n件展示。至少需要多少空间（多少比特）完整存储该信息（即哪n件被展示了）？

当$|\mathcal{U}| \gg n$时，需要的存储空间至少是$\Omega(n\log_2|\mathcal{U}|)$比特

如果$|\mathcal{U}|$代表所有可能的网站URL、n是当地用户已经请求访问的网站URL，该数字非常大

准确记忆用户已经请求过哪些网站/文件将消耗大量的存储资源

如何以较小的存储空间解决从属问题?

无论采用何种办法，如果要实现准确记忆，必然占用大量存储空间（已证理论下界）

采用“以精确度换存储空间”的思想

内容分发网络

近似从属判断问题（approximate membership query）
如何存储关于集合$\mathcal{S}$的信息从而可以近似判断关于“元素x是否属于集$\mathcal{S}$”的问题?

当$x \in \mathcal{S}$时，要100%正确判断；

当$x \notin \mathcal{S}$时，要大概率正确判断：即以$1-\varepsilon$的概率返回“不属于”的答案

内容分发网络

近似从属判断问题（approximate membership query）
如何存储关于集合$\mathcal{S}$的信息从而可以近似判断关于“元素x是否属于集$\mathcal{S}$”的问题?

当$x \in \mathcal{S}$时，要100%正确判断；

当$x \notin \mathcal{S}$时，要大概率正确判断：即以$1-\varepsilon$的概率返回“不属于”的答案

允许以小概率出现误报（false positive），不允许出现漏报（false negative）

满足缓存的实际应用

内容分发网络

近似从属判断问题（approximate membership query）

如何存储关于集合$\mathcal{S}$的信息从而可以近似判断关于“元素x是否属于集$\mathcal{S}$”的问题？

当$x \in \mathcal{S}$时，要100%正确判断；
当$x \notin \mathcal{S}$时，要大概率正确判断：即以$1-\varepsilon$的概率返回“不属于”的答案

允许以小概率出现误报（false positive），不允许出现漏报（false negative）
满足缓存的实际应用

设计的存储方式的空间复杂度不能随$|\mathcal{U}|$增长

$|\mathcal{U}|$：所有可能的网站/文件的个数
$|\mathcal{S}|$：用户访问过的网站/文件的个数（同前例子中的n）

布隆过滤器

布隆过滤器的方法：思想一

布隆过滤器

布隆过滤器（Bloom Filters）由Burton Howard Bloom在1970年提出

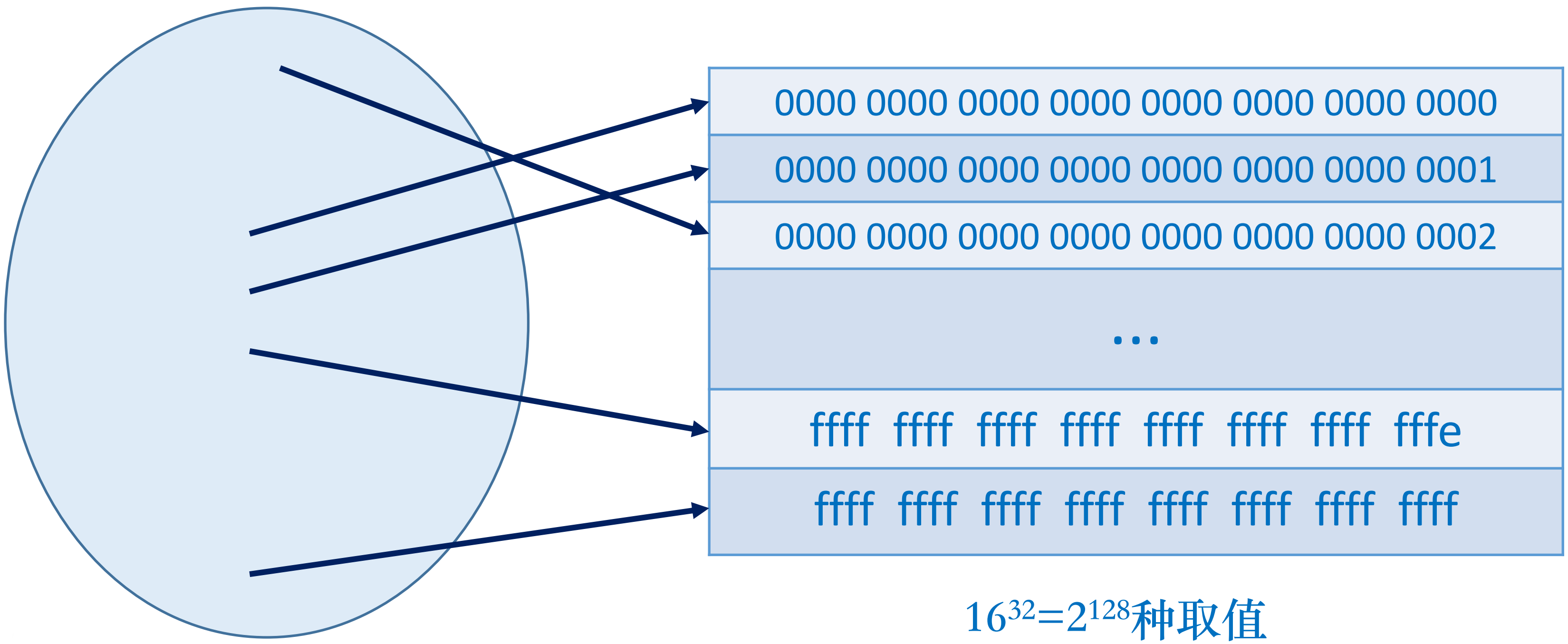
基于以下两个思想解决近似从属判断问题：

（1）通过哈希函数将网站/文件名进行映射

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

回顾哈希函数



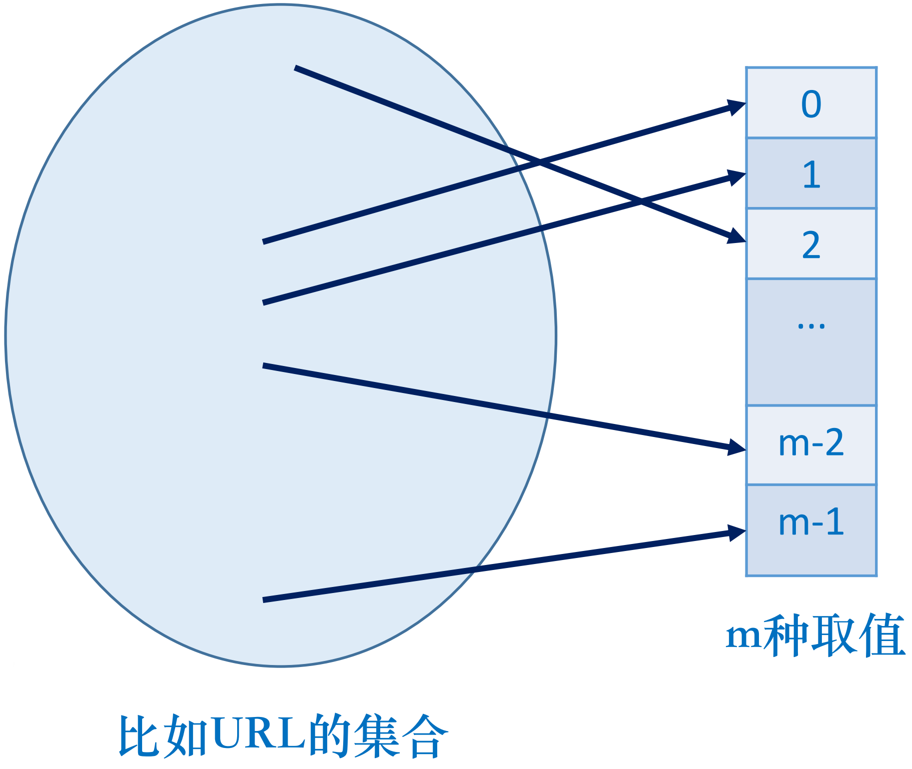
$16^{32}=2^{128}$种取值

比如URL的集合

上一讲中关于MD5的例子

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

As an example, let’s have

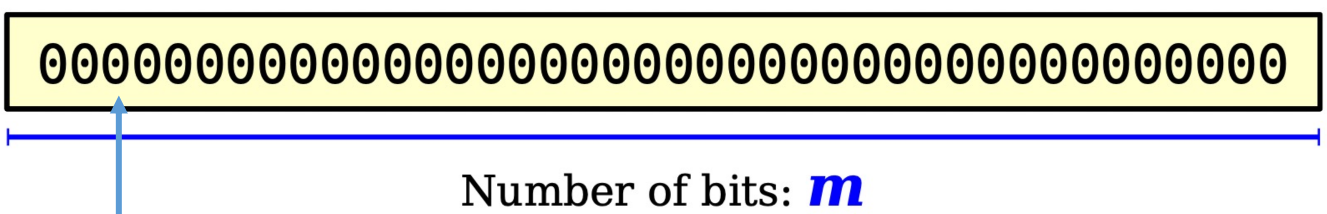
$S = \{103, 137, 166, 271, 314\}$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

As an example, let's have
$S = \{103, 137, 166, 271, 314\}$

例如$h(103) = 2$



布隆过滤器（m长的数组）

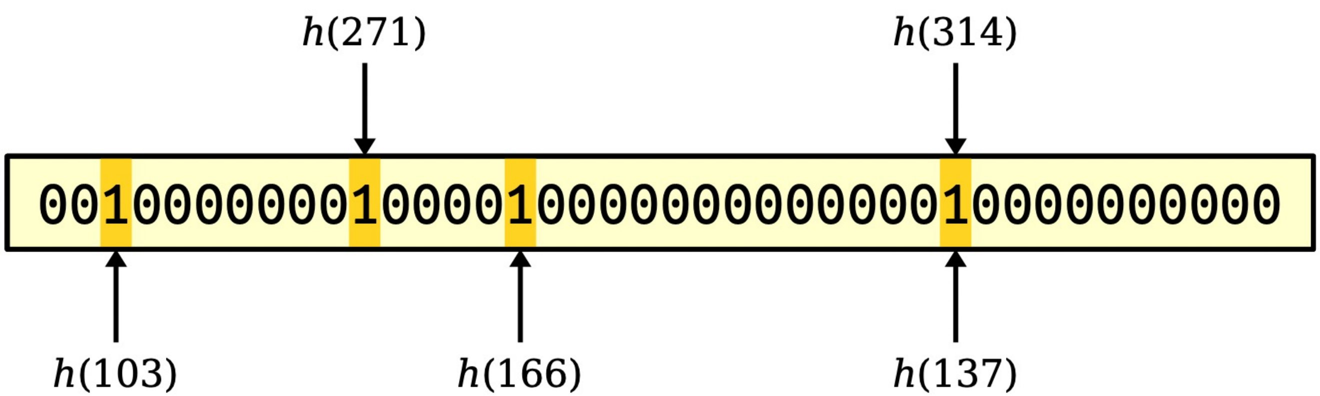
$h(103)$

例子取自Keith Schwarz <Stanford CS166 - Data Structures>

布隆过滤器

(1) 通过哈希函数将网站/文件名进行映射

As an example, let’s have
$S = \{103, 137, 166, 271, 314\}$



可能出现两个不同元素哈希值相同的情况

例子取自Keith Schwarz <Stanford CS166 - Data Structures>

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

> As an example, let's have
> $S = \{103, 137, 166, 271, 314\}$

用布隆过滤器判断103、161、261是否存在于集合S中

`0010000000100001000000000000010000000000`

$h(103)$ *True positive*

$h(161)$ *True negative*

$h(261)$ *False Positive*

即便261不在集合中，布隆过滤器仍会判断“在集合中”
原因是261的哈希值恰好等于137和314的哈希值

例子取自Keith Schwarz <Stanford CS166 - Data Structures>

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

As an example, let’s have
$S = \{103, 137, 166, 271, 314\}$

用布隆过滤器判断103、161、261是否存在于集合S中

`0010000000100001000000000000010000000000`

$h(103)$ True positive

$h(161)$ True negative

$h(261)$ False Positive

该情况不可避免，当m增大时该情况出现的可能性如何变化？

即便261不在集合中，布隆过滤器仍会判断“在集合中”
原因是261的哈希值恰好等于137和314的哈希值

例子取自Keith Schwarz <Stanford CS166 - Data Structures>

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中第一个元素的哈希值相同的概率?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中第一个元素的哈希值相同的概率为 $\frac{1}{m}$

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中至少一个元素的哈希值相同的概率为

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中第一个元素的哈希值相同的概率为 $\frac{1}{m}$

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中至少一个元素的哈希值相同的概率为$1-\left(1-\frac{1}{m}\right)^n$

即误报的概率为$1-\left(1-\frac{1}{m}\right)^n \leq \varepsilon$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中第一个元素的哈希值相同的概率为 $\frac{1}{m}$

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中至少一个元素的哈希值相同的概率为$1-\left(1-\frac{1}{m}\right)^n$

即误报的概率为$1-\left(1-\frac{1}{m}\right)^n \approx 1-e^{-\frac{n}{m}} \leq \varepsilon$ 经整理得 $m \geq \frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$

$$\boxed{\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e}}$$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

当布隆过滤器（数组）的长度m增大时，哈希值的范围增大，出现“误报”（false positive）的可能性降低

如果目标是当$x \notin \mathcal{S}$时，出现误报的概率小于等于ε，应该如何设置m?

* 已知集合$\mathcal{S}$的大小为n（即$|\mathcal{S}| = n$）

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中第一个元素的哈希值相同的概率为 $\frac{1}{m}$

元素x对应的哈希值$h(x)$与集合$\mathcal{S}$中至少一个元素的哈希值相同的概率为$1-\left(1-\frac{1}{m}\right)^n$

即误报的概率为$1-\left(1-\frac{1}{m}\right)^n \approx 1-e^{-\frac{n}{m}} \leq \varepsilon$ 经整理得 $m \geq \frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$

需要约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特的存储空间

例如，当ε为0.01时，该值约$99.5n$

能否改进从而降低该值?

布隆过滤器

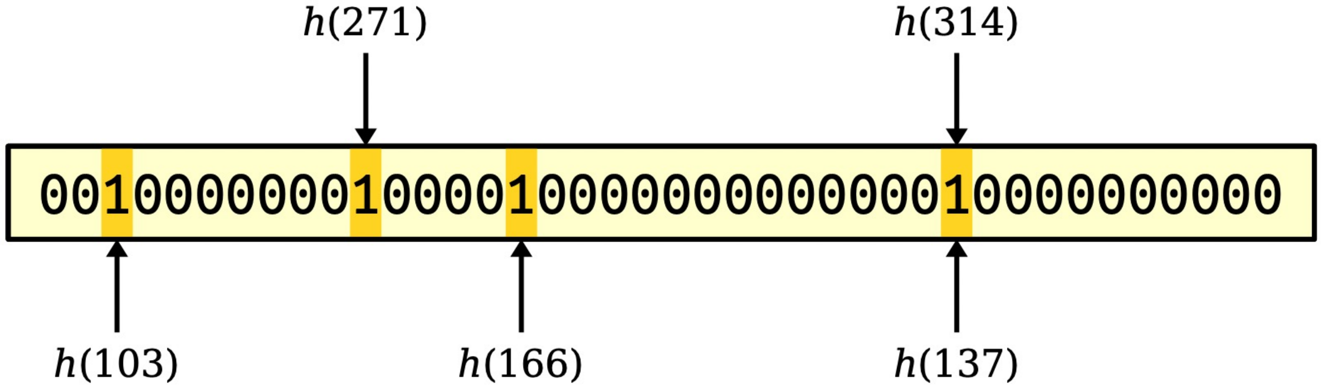
布隆过滤器的方法：思想二

布隆过滤器

(1) 通过哈希函数将网站/文件名进行映射

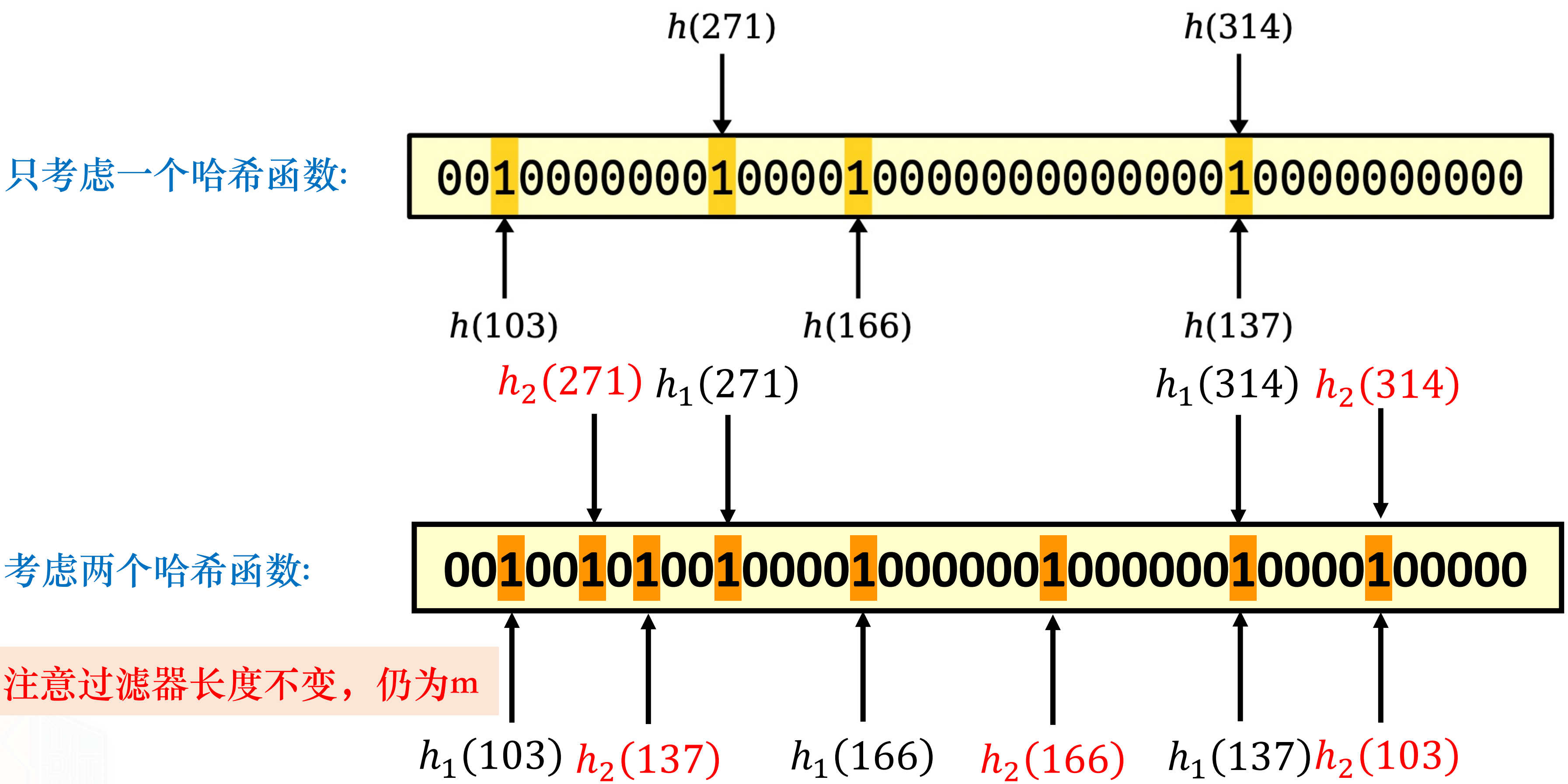
(2) 利用多个（k个）哈希函数

只考虑一个哈希函数:



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射
（2）利用多个（k个）哈希函数



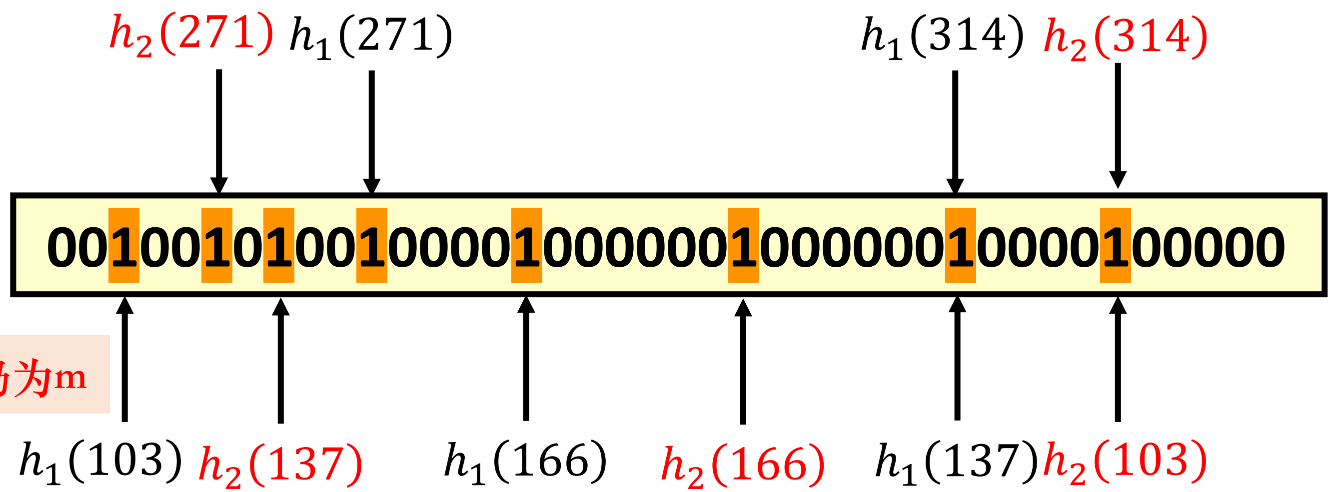
注意过滤器长度不变，仍为m

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

考虑两个哈希函数:



注意过滤器长度不变，仍为m

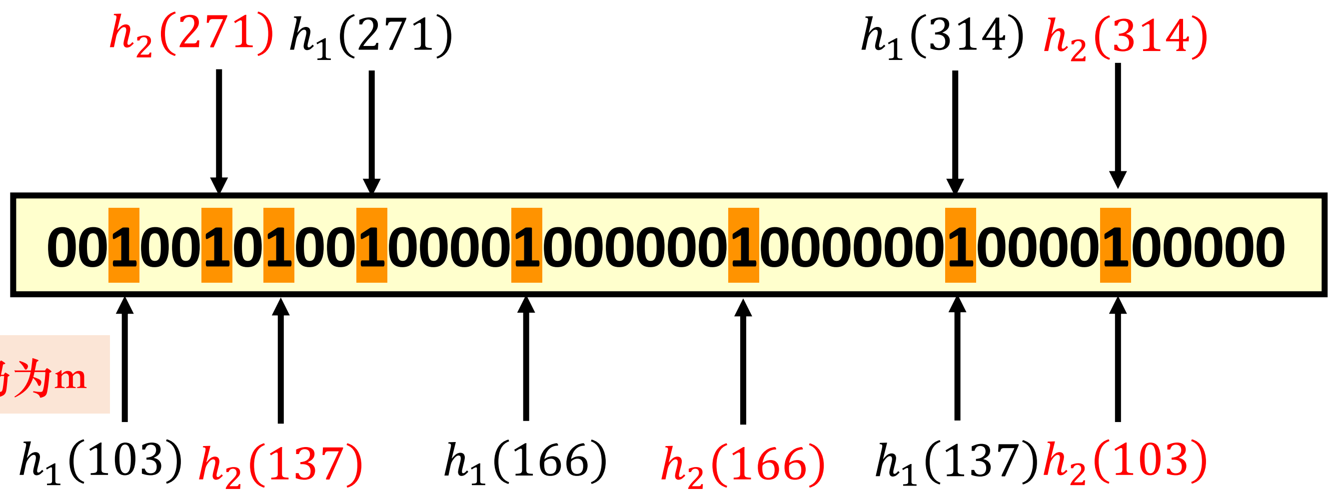
如何利用布隆过滤器判断元素是否在集合内?
比如查询261是否在集合内?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

考虑两个哈希函数:



注意过滤器长度不变，仍为m

如何查询261是否在集合内？计算$h_1(261)$和$h_2(261)$，检查过滤器相应位置是否都是1

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

考虑两个哈希函数:

0010010100100001000000100000010000100000

$h_2(261)$　　　　$h_1(261)$

注意过滤器长度不变，仍为m

如何查询261是否在集合内？计算$h_1(261)$和$h_2(261)$，检查过滤器相应位置是否都是1

因为只有当相应位置都是1的时候才会判断“在集合中”，可以减小“误报”的概率

如何计算“误报”的概率?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的哈希值$h_1(x)$在过滤器的位置取值为1的概率?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的哈希值$h_1(x)$在过滤器的位置取值为1的概率?

集合中每个元素分别对应k个哈希值，即共有nk个哈希值，其中至少有一个值对应该位置的概率为?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的哈希值$h_1(x)$在过滤器的位置取值为1的概率?

集合中每个元素分别对应k个哈希值，即共有nk个哈希值，其中至少有一个值对应该位置的概率为？ $1-\left(1-\frac{1}{m}\right)^{kn} \approx 1-e^{-\frac{kn}{m}}$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的哈希值$h_1(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的哈希值$h_2(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的哈希值$h_3(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

…

元素x对应的哈希值$h_k(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率为? $\left(1-e^{-\frac{kn}{m}}\right)^k$?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的哈希值$h_1(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的哈希值$h_2(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的哈希值$h_3(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

…

元素x对应的哈希值$h_k(x)$在过滤器的位置取值为1的概率为$1-e^{-\frac{kn}{m}}$

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率为? $\left(1-e^{-\frac{kn}{m}}\right)^k$?

其实不等于$\left(1-e^{-\frac{kn}{m}}\right)^k$，但实际概率与$\left(1-e^{-\frac{kn}{m}}\right)^k$相近。为便于分析，将$\left(1-e^{-\frac{kn}{m}}\right)^k$作为近似值

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

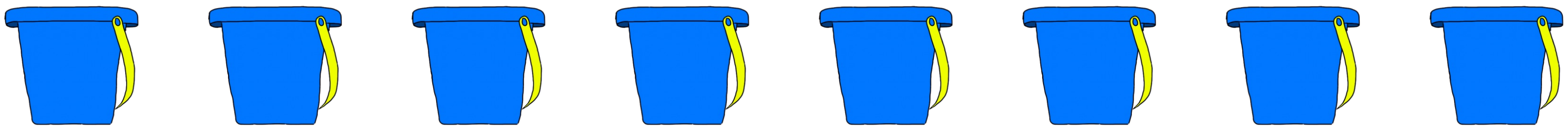
回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

$\Pr[$元素x对应的k个哈希值在过滤器的相应位置取值都为$1]$

$= \Pr[h_1(x)$位置取值为$1; h_2(x)$位置取值为$1; \ldots; h_k(x)$位置取值为$1]$

是否等于 $\Pr[h_1(x)$位置取值为$1] \times \Pr[h_2(x)$位置取值为$1] \ldots \times \Pr[h_k(x)$位置取值为$1]$

布隆过滤器



有8个桶子，每个桶子里可能有水也可能没水，具体有多少个桶里有水未知

先后依次让20个人来随机选择1个桶，随机事件“第i个人选择的桶中有水”与随机事件“第j个人选择的桶中有水”是否是独立事件?

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

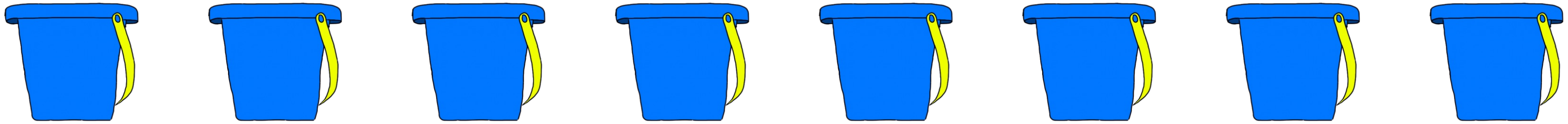
$\Pr$[元素x对应的k个哈希值在过滤器的相应位置取值都为1]

$= \Pr[h_1(x)$位置取值为1; $h_2(x)$位置取值为1; …; $h_k(x)$位置取值为1]

不等于！ $\Pr[h_1(x)$位置取值为1] $\times$ $\Pr[h_2(x)$位置取值为1] … $\times$ $\Pr[h_k(x)$位置取值为1]

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率不等于$\left(1 - e^{-\frac{kn}{m}}\right)^k$

布隆过滤器



有8个桶子，每个桶子里可能有水也可能没水，已知有4个桶子里面有水

先后依次让20个人来随机选择1个桶，随机事件“第i个人选择的桶中有水”与随机事件“第j个人选择的桶中有水”是否是独立事件？

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

$\Pr[$元素x对应的k个哈希值在过滤器的相应位置取值都为1$]$

$= \Pr[h_1(x)$位置取值为1; $h_2(x)$位置取值为1; ...; $h_k(x)$位置取值为1$]$

$= \sum_{t=0}^{m} \Pr[m$位置中有t个为0$]\ \Pr[h_1(x)$位置取值为1; ...; $h_k(x)$位置取值为1$\,|\,m$个位置中有t个为0$]$

$= \sum_{t=0}^{m} \Pr[m$位置中有t个为0$](\Pr[h_1(x)$位置取值为1$\,|\,m$位置中有t个为0$] \times \ldots \times \Pr[h_k(x)$位置取值为1$\,|\,m$位置中有$t$个为0$])$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

Pr[元素x对应的k个哈希值在过滤器的相应位置取值都为1]

$= \Pr[h_1(x)$位置取值为1; $h_2(x)$位置取值为1; ...; $h_k(x)$位置取值为1]

$= \sum_{t=0}^{m} \Pr[m$位置中有t个为0] $\Pr[h_1(x)$位置取值为1; ...; $h_k(x)$位置取值为1 | m个位置中有t个为0]

$= \sum_{t=0}^{m} \Pr[m$位置中有t个为0]($\Pr[h_1(x)$位置取值为1 | m位置中有t个为0] $\times \ldots \times \Pr[h_k(x)$位置取值为1 | m位置中有t个为0])

$= \sum_{t=0}^{m} \Pr[m$位置中有t个为0] $\left(1-\frac{t}{m}\right)^k$

$\approx \sum_{t=0}^{m} \Pr[m$位置中有$t$个为0] $e^{-\frac{kt}{m}}$

后续计算比较复杂，最终结果并不易于分析

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$\left(1-e^{-\frac{kn}{m}}\right)^k$，令其小于等于$\varepsilon$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

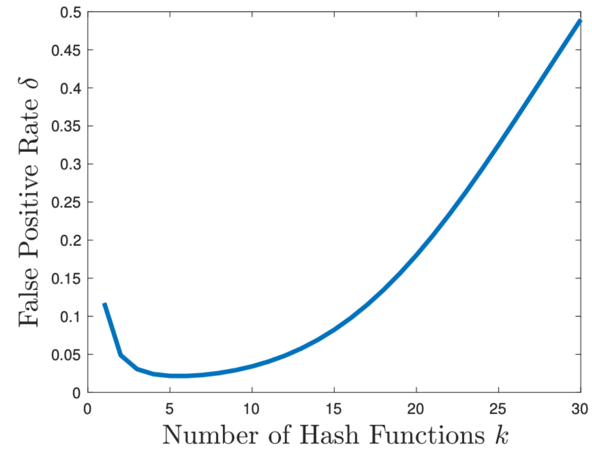
回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$\left(1-e^{-\frac{kn}{m}}\right)^k$，令其小于等于$\varepsilon$

先选择k，最小化$\left(1-e^{-\frac{kn}{m}}\right)^k$

根据先减后增的趋势，导数为0处取最优的k值

为什么先减后增?



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

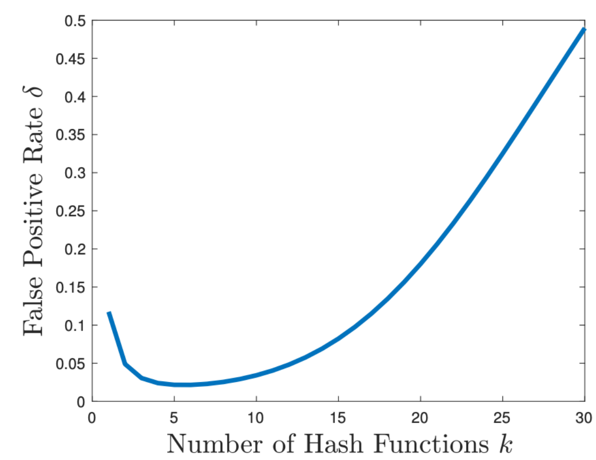
元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$\left(1-e^{-\frac{kn}{m}}\right)^k$，令其小于等于$\varepsilon$

先选择k，最小化$\left(1-e^{-\frac{kn}{m}}\right)^k$

根据先减后增的趋势，导数为0处取最优的k值

方法一：求导找导数为零的条件

$$\frac{d}{dx}\left[a^{f(x)}\right] = \underbrace{a^{f(x)}}_{\text{rewrite}} \cdot \underbrace{\ln a}_{\text{natural log of base}} \cdot \underbrace{f'(x)}_{\text{derivative of exponent}}$$



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

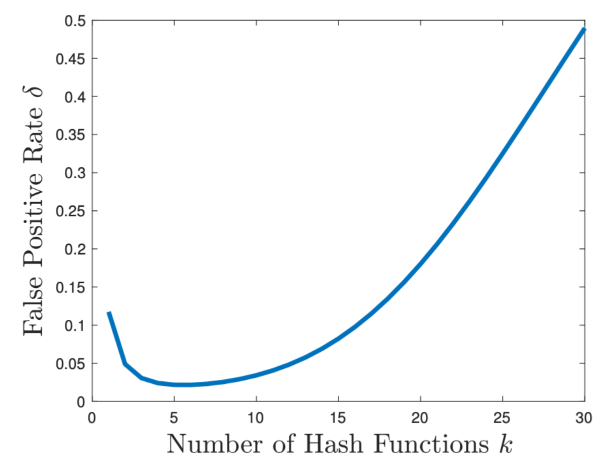
回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$\left(1-e^{-\frac{kn}{m}}\right)^k$，令其小于等于$\varepsilon$

先选择k，最小化$\left(1-e^{-\frac{kn}{m}}\right)^k$

根据先减后增的趋势，导数为0处取最优的k值

方法二：利用关系 $k\ln\left(1-e^{-\frac{kn}{m}}\right)=-\frac{m}{n}\ln e^{-\frac{kn}{m}}\ln\left(1-e^{-\frac{kn}{m}}\right)$



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

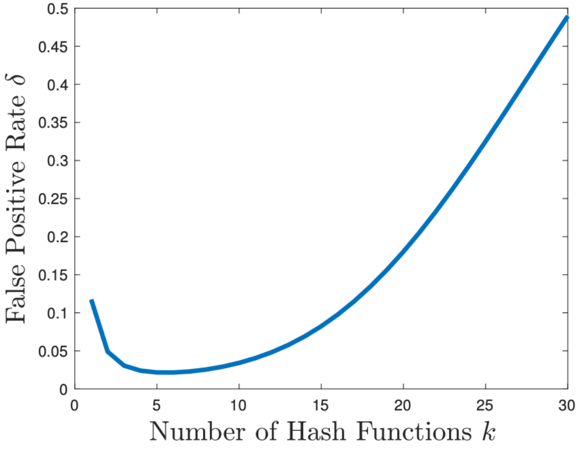
回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$\left(1-e^{-\frac{kn}{m}}\right)^k$，令其小于等于$\varepsilon$

先选择k，最小化$\left(1-e^{-\frac{kn}{m}}\right)^k$

根据先减后增的趋势，导数为0处取最优的k值

最优k值为$\frac{m}{n}\ln 2$，相应概率为$2^{-\frac{m}{n}\ln 2}$



布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$2^{-\frac{m}{n}\ln 2}$，令其小于等于ε

m的取值应该满足：$m \geq \frac{\log_2 \frac{1}{\varepsilon}}{\ln 2} n$

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

元素x对应的k个哈希值在过滤器的相应位置取值都为1的概率约为$2^{-\frac{m}{n}\ln 2}$，令其小于等于ε

m的取值应该满足：$m \geq \frac{\log_2 \frac{1}{\varepsilon}}{\ln 2} n$

若用k个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{\log_2 \frac{1}{\varepsilon}}{\ln 2} n$比特

布隆过滤器

（1）通过哈希函数将网站/文件名进行映射

（2）利用多个（k个）哈希函数

如果目标是当$x \notin S$时，出现误报的概率小于等于ε，应该如何设置m?

回顾：若仅用一个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{n}{\ln\left(\frac{1}{1-\varepsilon}\right)}$比特

当ε为0.01时，该值约$99.5n$

若用k个哈希函数，为使误报率小于等于ε，所需存储空间（即m大小）约$\frac{\log_2\frac{1}{\varepsilon}}{\ln 2}n$比特

当ε为0.01时，该值约$9.6n$

m值的计算：https://hur.st/bloomfilter/?n=4000&p=1.0E-7&m=&k=

布隆过滤器

实验效果

缓存策略回顾

一种简单的缓存策略：当用户请求访问某文件，先判断该文件是否是第一次被请求：

（1）如果是，不缓存；

（2）如果不是，缓存。

* 传输策略：如果用户请求的文件已被缓存，则从本级服务器提取文件进行传输；否则，从上级服务器中查找/提取文件进行传输

内容分发网络

回顾

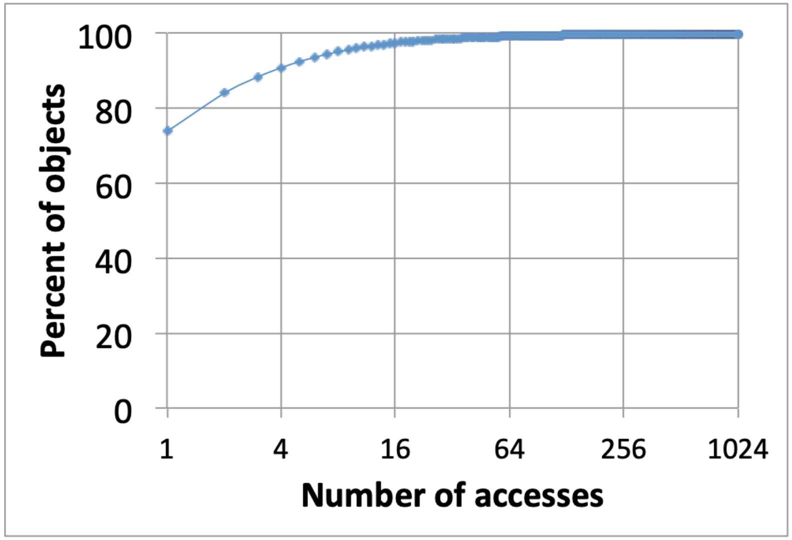


Figure 5: On a typical CDN server cluster serving web traffic over two days, 74% of the roughly 400 million objects in cache were accessed only once and 90% were accessed less than four times.

（2015年的论文）分析了Akamai的一个有45台服务器的集群，统计在两天时间内4亿份文件被用户访问的频次

图片引自B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery>

内容分发网络

在一个有47台服务器（47×8个硬盘）的集群上测试布隆过滤器的效果（关闭 — 打开 — 关闭）

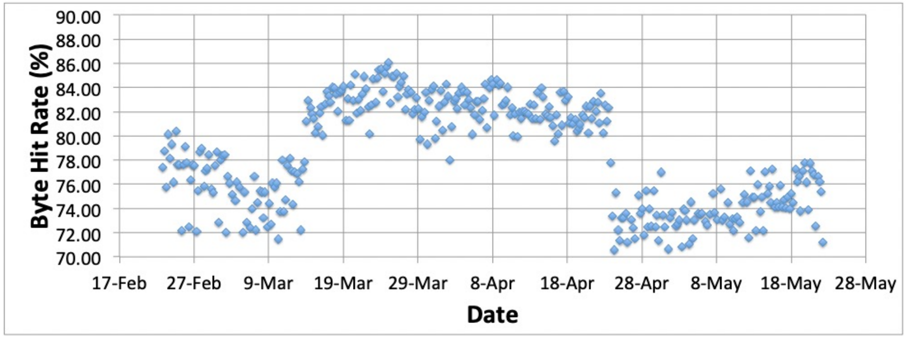


Figure 6: Byte hit rates increased when cache filtering was turned on between March 14th and April 24th because not caching objects that are accessed only once leaves more disk space to store more popular objects.

Byte hit rate: 用户请求文件时文件已被缓存的比例

比如75%说明对每100个比特的网络流量，其中有75个比特被缓存/25个比特通过从上级服务器获取

实验引自B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery>

内容分发网络

在一个有47台服务器（47×8个硬盘）的集群上测试布隆过滤器的效果（关闭 — 打开 — 关闭）

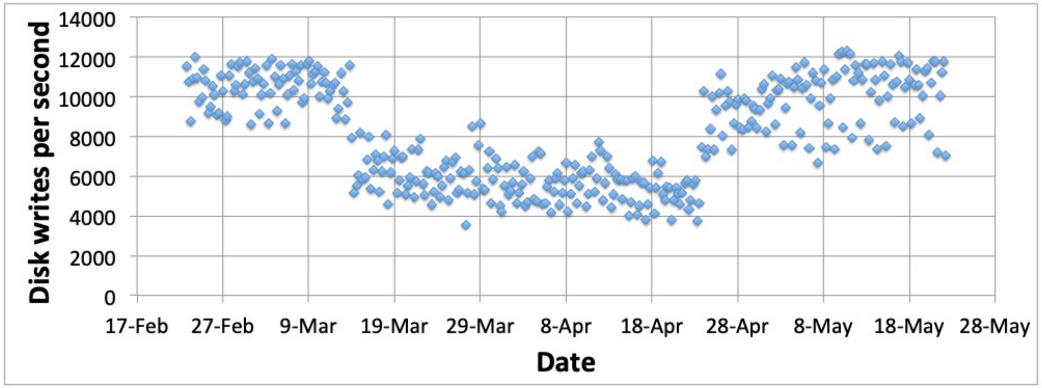


Figure 7: Turning on cache filtering decreases the rate of disk writes by nearly one half because objects accessed only once are not written to disk.

硬盘写入频率的变化

实验引自B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery>

内容分发网络

在一个有47台服务器（47×8个硬盘）的集群上测试布隆过滤器的效果（关闭 — 打开 — 关闭）

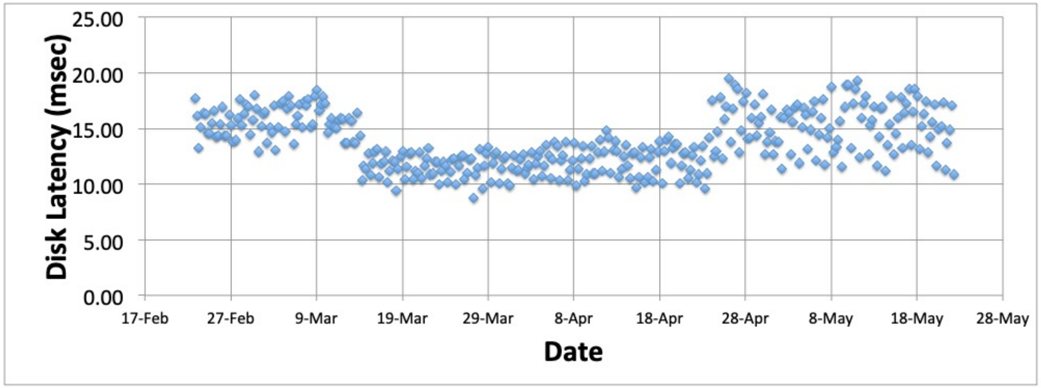


Figure 8: The latency of reading content from disk decreases when cache filtering is turned on, since there are fewer competing disk writes.

文件读取时延的变化

实验引自B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery>

本讲小结

- 从属问题
- 布隆过滤器的方法

主要参考资料

Tim Roughgarden and Gregory Valiant <CS 168 - The Modern Algorithmic Toolbox> Lecture Notes

Cameron Musco <COMPSCI 514 - Algorithms for Data Science> Slides

B.M. Maggs and R.K. Sitaraman <Algorithmic Nuggets in Content Delivery> Paper

Keith Schwarz <Stanford CS166 - Data Structures> Slides

谢谢！