中国科学技术大学

读书报告

布隆过滤器及其衍生的新型数据结构

作者姓名：　柳枫

学科专业：　网络空间安全

导师姓名：　薛开平教授

完成时间：　二〇二四年八月二十五日

目　录

符 号 说 明

a	The number of angels per unit area
N	The number of angels per needle point
A	The area of the needle point
σ	The total mass of angels per unit area
m	The mass of one angel
$\sum_{i=1}^{n} a_i$	The sum of a_i

第 1 章　简　　介

1.1　布隆过滤器

在构造网络安全协议时，我们通常会使用到许多不同类型的数据结构。其中，布隆过滤器作为一种经典的数据结构，在诸如隐私集合求交、可搜索加密、隐私信息检索等密码学协议中有着广泛的应用。布隆过滤器是一种用于快速判断元素是否存在于某一集合的数据结构，它具有空间效率高、判断速度快的特点。以大小为 n 的集合 S 为例，对应的布隆过滤器构造只需要 $O(n)$ 的存储开销以及 $O(1)$ 的判断复杂度。布隆过滤器的构造如下图所示，它是使用 k 个哈希函数 $\{h_1, \cdots, h_k\}$ 构造的哈希表结构。布隆过滤器上分为插入（Insert）和查找（Lookup）两个算法，在构造过程中，对于每个在集合 S 中的元素 x，首先使用这 k 个哈希函数计算出 k 个位置，然后对过滤器中该位置上的比特置为 1。在判断元素是否属于集合 S 时，只需要通过哈希函数计算该元素的 k 个位置，然后检查过滤器上这 k 个位置上的比特是否全为 1。如果是，返回 True，否则返回 False。从布隆过滤器的构造可以看出，

文献[1]：

布隆过滤器（Bloom filter，BF）是一种空间效率高的概率型数据结构，它可以用来快速判断元素是否属于某一集合。布隆过滤器是由 0-1 比特组成的比特向量结构，包含插入（Insert）和查找（Lookup）两个基本算法。以包含 n 个元素的集合 $S = \{x_1, x_2, \cdots, x_n\}$ 为例，假设构造的布隆过滤器长度为 m，使用的哈希函数为 $\{h_1, h_2, \cdots, h_k\}$，其中每个哈希函数 $h_i : \{0,1\}^* \to [0, m-1]$ 为任意长度的输入到布隆过滤器上某一位置的映射。首先我们将布隆过滤器 m 个位置上的比特都置为 0，然后再插入集合 S 中的每一个元素。在插入元素 x 时，需要使用 k 个哈希函数计算出 k 个位置信息，即 $\{h_1(x), h_2(x), \cdots, h_k(x)\}$。再将布隆过滤器上这 k 个位置上的比特都置为 1。在判断某个元素是否属于集合 S 时，只需要计算该元素对应的 k 个位置，然后检查这 k 个位置上的比特是否都为 1。只要有一个位置上出现了 0，那么判断结果就是不属于；否则，布隆过滤器认为该元素属于集合 S。

从布隆过滤器的构造和判断算法中可以看出，如果一个元素属于集合 S，那么判断结果一定是正确的；但是如果一个元素不属于集合 S（False 的情况），布隆过滤器也有可能会认为该元素属于 S（输出结果为 Positive），此时判断错误的概率也称为假阳性率（False Positive Rate）。尽管布隆过滤器存在误判的问题，但在实际应用场景中，只要将误判率控制在较小的值，一般认为以一定的误判换取低空间开销和高效判断是值得的。

布隆过滤器的误判率 f_r 由布隆过滤器的长度 m，使用的哈希函数个数 k 和集合中的元素个数 n 所决定。理论上，误判率 f_r 与它们的关系如公式 (1.1)所示：

$$f_r = \left[1 - \left(1 - \frac{1}{m}\right)^{nk}\right]^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k, \tag{1.1}$$

其中，$(1 - 1/m)^{nk}$ 近似成 $e^{-kn/m}$ 的形式。为了尽可能降低误判率 f_r，那么就需要尽可能降低 $e^{-kn/m}$ 的值，这样一来，k 的最优取值为：

$$k_{opt} = \frac{m}{n} \ln 2 \approx \frac{9m}{13n}. \tag{1.2}$$

此时，误判率大约为 $0.5^k \approx 0.6185^{m/n}$。通常在实际应用中的误判率要比理论分析上的更高。也有一些工作对误判率做了更精确的刻画。

布隆过滤器本身是不支持删除的，因为如果是简单将需要删除元素所对应位置的比特置为 0，那么就会对其他元素的判断造成影响。

布隆过滤器的高效体现在两个方面：

- 空间利用率：布隆过滤器的大小与元素的大小无关，只与集合中元素的数量有关。比如，当给定 m 与 n 的比值为 5 时，根据公式 (1.2) 可以计算出需要的哈希函数数量为 3 或 4。因为布隆过滤器中每个位置上存储的都是比特，所以整体长度也就是 m 比特。
- 恒定时间的查询时间：因为只需要检查 k 个位置上的比特是否全为 1，因此检查一个元素的时间复杂度为 $O(k)$。相比于树形结构的查询效率（$O(\log(n))$）或列表结构的查询效率（$O(n)$）都要高。而对于具体的布隆过滤器实例来说，k 的值在初始化阶段就是常数，因此插入和查询的复杂度都为 $O(1)$。
- 无漏判：尽管布隆过滤器在查询时会存在误判的情况，但是它不会出现漏判（假阴性）的情况。也就是只要是布隆过滤器判断元素 x 不属于 S，那么该论断一定是正确的。

布隆过滤器的局限性：

- 误判，如果减少误判带来的影响
- 实现，实际实现中需要考虑访问的优化，哈希计算的优化
- 灵活性，哈希函数不能变，集合一开始就是确定的
- 功能有限，只能做成员存在性测试

降低误判率的方法（思路）：首先，

1.2　分类

在介绍布隆过滤器相关衍生的数据结构之前，我们首先需要对这些数据结构进行分类。对此，我们对这些数据结构进行了如下统一的定义。

定义 1.1　令 $\mathcal{U}$ 表示元素的集合，$\mathcal{H}$ 为哈希函数的集合。过滤器一般包含以下两个算法：

- Construct$(S,\mathcal{H}) \to F/\perp$：输入集合 $S \subseteq \mathcal{U}$ 和预先给定的哈希函数集合 $\mathcal{H}$，输出构造的过滤器 F（或者以可忽略的概率输出错误指示符 $\perp$）。
- Evaluate$(x,\mathcal{H},F) \to$ True/False：输入元素 x，预先给定的哈希函数集合 $\mathcal{H}$，输出结果 True 或者 False。

正确性：对于任意的 $S \subseteq \mathcal{U}$，都有：1）构建过程中，输出 $\perp$ 的概率是可忽略的；2）如果 $F \leftarrow$ Construct$(S,\mathcal{H})$，且 $F \neq \perp$，那么在判断过程中，对于任意的 $x \in S$，Pr[Evaluate$(x,\mathcal{H},F)$ = True] = 1；对于任意 $x' \notin S$，Pr[Evaluate$(x,\mathcal{H},F)$ = True] 为可忽略的。

从以上定义中可以看出，如果一个元素在原本输入的集合中，那么过滤器在判断过程中一定能返回正确的结果，即过滤器中不存在假阴性的情况；反之，如果一个元素不存在于输入的集合中，过滤器会大概率返回正确的结果，即过滤器中会存在一定的假阳性率。

在判断过程中，需要使用 $\mathcal{H}$ 中的哈希函数计算出元素在 F 中对应的位置，再对这些位置上记录的结果进行计算，最后根据计算结果与事先定义的 $f(x)$ 进行比较返回 Ture 或者 False。计算过程也被称为探测（probing），文献[2]将探测方式分为以下四种类型：

- AND 型：在通过哈希函数计算出的位置中，如果所有位置上结果都与 $f(x)$ 相等，那么就输出 True；
- OR 型：在通过哈希函数计算出的位置中，如果至少有一个位置上的值与 $f(x)$ 相等，那么就输出 True，否则输出 False。该类型的过滤器典型代表是布谷鸟过滤器[3]，这种构造可以方便
- XOR 型：在通过

这里引用文献[4]

参 考 文 献

[1] LUO L, GUO D, MA R T B, et al. Optimizing Bloom filter: Challenges, solutions, and comparisons[J/OL]. IEEE Communications Surveys & Tutorials, 2019, 21(2): 1912-1949. DOI: 10.1109/COMST.2018.2889329.

[2] DILLINGER P C, WALZER S. Ribbon filter: Practically smaller than Bloom and Xor: arXiv:2103.02515[M/OL]. arXiv, 2021. DOI: 10.48550/arXiv.2103.02515.

[3] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo filter: Practically better than bloom [C/OL]//Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CONEXT). ACM, 2014: 75-88. DOI: 10.1145/2674005.2674994.

[4] 张响鸽, 张聪, 刘巍然, 等. 隐私集合运算中的关键数据结构研究[J/OL]. 密码学报, 2024, 11(2): 263-281. DOI: 10.13868/j.cnki.jcr.000679.