布隆过滤器 Bloom Filter

不仅可以判断它是否在集合中
还可以存很多其余的信息
(元素本身+元素的各种额外信息)

布隆过滤器 与 哈希表 类似

Bloom Filter vs Hash Table

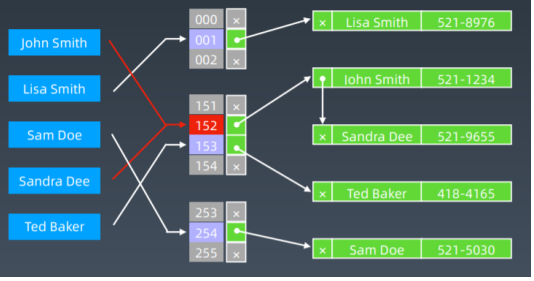
一个很长的二进制向量和一系列随机映射函数。布隆过滤器可以用于检索一个元素是否在一个集合中。

不能存储额外信息

优点是空间效率和查询时间都远远超过一般的算法，

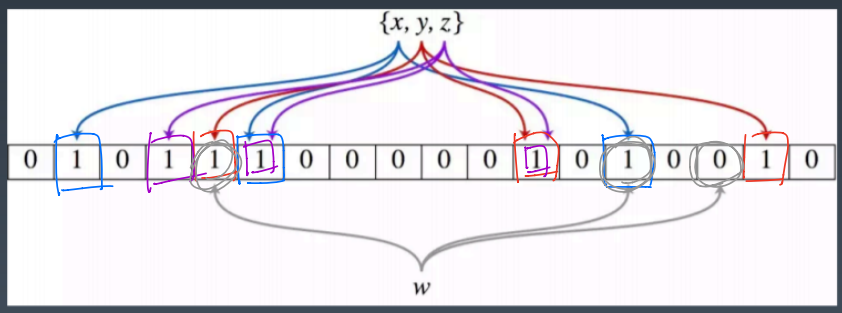
缺点是有一定的误识别率和删除困难。

HashTable + 拉链存储重复元素



↑ 用了二进制向量来表示
& 是一种模糊的查询方式

布隆过滤器示意图



原理：对于任何一个角色，这里假设要存三个元素依次存进来，

x, y, z 一个一个依次往里面灌

每一个元素，它会分配到一系列的二进制位中，

假设 x 分配到三个二进制位

x插入到布隆过滤器的话，就表示把x对应的这三个位置置为1就可以了。

其他 (y、z) 同理也置为1

这个二进制数组，用来表示所有的已经存入的 x, y, z 是否已经在索引里面了

这时候重新给你一个元素 x 的的话会对应于这些色的，始终 x 会对应于这三个蓝色的二进制位

(去表里面查，查到三个皆为1，所以可以认为x是存在的)

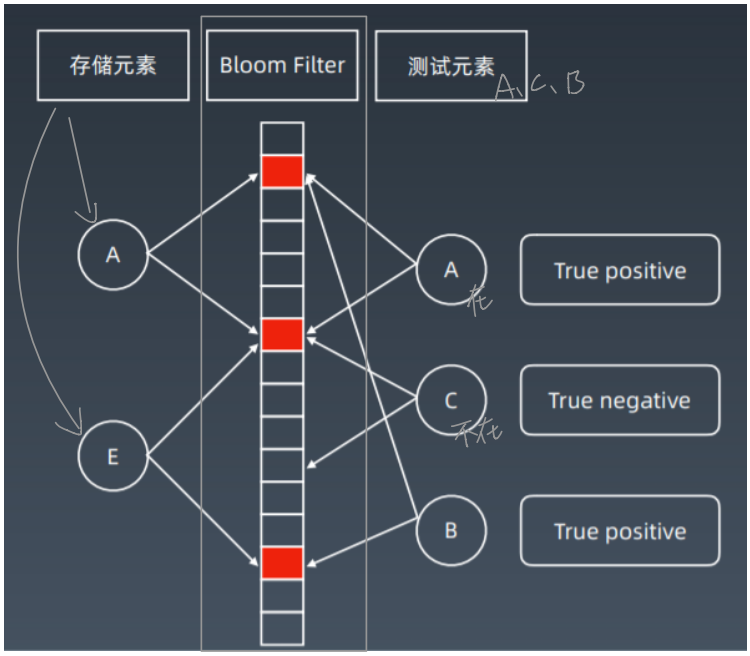
W一个陌生的元素进来，把它分配给通过布隆过滤器的三进制索引的函数的函数的话，
W就得到灰色的这三个三进制位 "1 1 0"

有一个三进制位为0

→ 说明：W未在索引里面

特点 如果一个元素它所对应的三进制位只要有一个为0，
就说明不在布隆过滤器的索引里面，且肯定它不在的

但是当一个元素9，刚好分配的三个三进制位都为1的话，
不一定能说是在的



对B的判断有误差的

结论 1、当布隆过滤器把元素都插入完了之后，
对于测试元素（就是对于新来的一个元素）要来验证它是否存在的话，

2、当它验证这个元素所对应的三进制位是1的时候，我们只能说它可能存在在布隆过滤器里面
但是当这个元素所对应的三进制位只要有一个不为1的话，
那我们可以肯定它不在

布隆过滤器：只是放在最外面当一个缓存使的（快速判断）

真的要确定元素一定存在的话，它必须再访问这个机器里面的一个完整的存储数据结构（DB）

案例

1. 比特币网络
2. 分布式系统（Map-Reduce）— Hadoop、search engine
3. Redis 缓存
4. 垃圾邮件、评论等的过滤

科普：https://www.cnblogs.com/cpselvis/p/6265825.html
https://blog.csdn.net/tianyaleixiaowu/article/details/74721877