千腾网
www.specl.com
网聚全球特产

# Bloom Filter

# 换个角度思考

- 有1000瓶药水，但是其中有一瓶是有毒的，小白鼠吃了一个星期以后就会死掉，请问，在一个星期内找出有毒的药物，最少需要多少只小白鼠？

- 1 0000000001
- 2 0000000010
- 3 0000000011
- 4 0000000100
- 5 0000000101

…

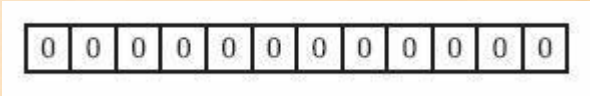
- 1000 1111101000

# 简介

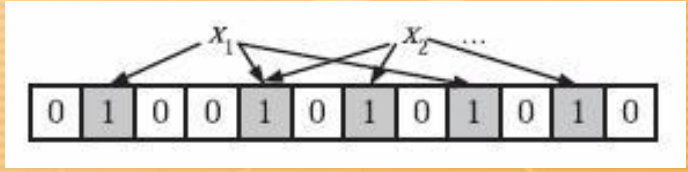
- Bloom Filter是一种空间效率很高的随机数据结构，它利用位数组很简洁地表示一个集合，并能判断一个元素是否属于这个集合。

# 集合表示

- 下面我们具体来看Bloom Filter是如何用位数组表示集合的。初始状态时，Bloom Filter是一个包含m位的位数组，每一位都置为0。

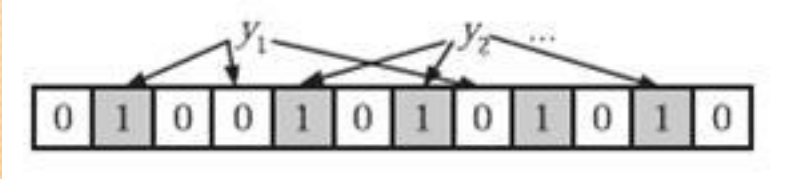



- 为了表达S={x1, x2,…,xn}这样一个n个元素的集合，Bloom Filter使用k个相互独立的哈希函数（Hash Function），它们分别将集合中的每个元素映射到{1,…,m}的范围中。对任意一个元素x，第i个哈希函数映射的位置hi(x)就会被置为1（1≤i≤k）。注意，如果一个位置多次被置为1，那么只有第一次会起作用，后面几次将没有任何效果。在下图中，k=3，且有两个哈希函数选中同一个位置（从左边数第五位）。



# 查询

- 在判断y是否属于这个集合时，我们对y应用k次哈希函数，如果所有$h_i(y)$的位置都是1（$1 \le i \le k$），那么我们就认为y是集合中的元素，否则就认为y不是集合中的元素。下图中$y_1$就不是集合中的元素。$y_2$或者属于这个集合，或者刚好是一个false positive。



# 适用场合

- Bloom Filter的这种高效是有一定代价的：在判断一个元素是否属于某个集合时，有可能会把不属于这个集合的元素误认为属于这个集合（False Position）。因此，Bloom Filter不适合那些“零错误”的应用场合。而在能容忍低错误率的应用场合下，Bloom Filter通过极少的错误换取了存储空间的极大节省。

# 最优的哈希函数个数

- 既然Bloom Filter要靠多个哈希函数将集合映射到位数组中，那么应该选择几个哈希函数才能使元素查询时的错误率降到最低呢？这里有两个互斥的理由：如果哈希函数的个数多，那么在对一个不属于集合的元素进行查询时得到0的概率就大；但另一方面，如果哈希函数的个数少，那么位数组中的0就多。为了得到最优的哈希函数个数，我们需要根据上一小节中的错误率公式进行计算。
- 先用p和f进行计算。注意到$f = \exp(k \ln(1 - e^{-kn/m}))$，我们令$g = k \ln(1 - e^{-kn/m})$，只要让g取到最小，f自然也取到最小。由于$p = e^{-kn/m}$，我们可以将g写成

$$g = -\frac{m}{n}\ln(p)\ln(1-p),$$

- 根据对称性法则可以很容易看出当p = 1/2，也就是**k = ln2· (m/n)**时，g取得最小值。在这种情况下，最小错误率f等于$(1/2)^k \approx (0.6185)^{m/n}$。另外，注意到p是位数组中某一位仍是0的概率，所以p = 1/2对应着位数组中0和1各一半。换句话说，要想保持错误率低，最好让位数组有一半还空着。
- 需要强调的一点是，p = 1/2时错误率最小这个结果并不依赖于近似值p和f。同样对于$f' = \exp(k \ln(1 - (1 - 1/m)^{kn}))$，$g' = k \ln(1 - (1 - 1/m)^{kn})$，$p' = (1 - 1/m)^{kn}$，我们可以将g'写成

$$g' = \frac{1}{n\ln(1-1/m)}\ln(p')\ln(1-p'),$$

- 同样根据对称性法则可以得到当p' = 1/2时，g'取得最小值。

# 位数组的大小

- 在错误率不大于ϵ的情况下，m至少要等于n $\log_2(1/\epsilon)$才能表示任意n个元素的集合。
- 上面我们曾算出当$k = \ln 2 \cdot (m/n)$时错误率f最小，这时$f = (1/2)^k = (1/2)^{m\ln 2 / n}$。现在令f≤ϵ，可以推出 $m \geq n\frac{\log_2(1/\epsilon)}{\ln 2} = n \log_2 e \cdot \log_2(1/\epsilon).$

- 这个结果比前面我们算得的下界n $\log_2(1/\epsilon)$大了$\log_2 e \approx 1.44$倍。这说明在哈希函数的个数取到最优时，要让错误率不超过ϵ，m至少需要取到最小值的**1.44**倍。

# 错误概率

| m/n | k | k=1 | k=2 | k=3 | k=4 | k=5 | k=6 | k=7 | k=8 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 1.39 | 0.393 | 0.400 | | | | | | |
| 3 | 2.08 | 0.283 | 0.237 | 0.253 | | | | | |
| 4 | 2.77 | 0.221 | 0.155 | 0.147 | 0.160 | | | | |
| 5 | 3.46 | 0.181 | 0.109 | 0.092 | 0.092 | 0.101 | | | |
| 6 | 4.16 | 0.154 | 0.0804 | 0.0609 | 0.0561 | 0.0578 | 0.0638 | | |
| 7 | 4.85 | 0.133 | 0.0618 | 0.0423 | 0.0359 | 0.0347 | 0.0364 | | |
| 8 | 5.55 | 0.118 | 0.0489 | 0.0306 | 0.024 | 0.0217 | 0.0216 | 0.0229 | |
| 9 | 6.24 | 0.105 | 0.0397 | 0.0228 | 0.0166 | 0.0141 | 0.0133 | 0.0135 | 0.0145 |
| 10 | 6.93 | 0.0952 | 0.0329 | 0.0174 | 0.0118 | 0.00943 | 0.00844 | 0.00819 | 0.00846 |

# 扩展

- 同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。
- Counting bloom filter（CBF）将位数组中的每一位扩展为一个counter，从而支持了元素的删除操作。
- Spectral Bloom Filter（SBF）将其与集合元素的出现次数关联。SBF采用counter中的最小值来近似表示元素的出现频率。

# 应用

- 加速查询

适用于一些key-value存储系统，当values存在硬盘时，查询就是件费时的事。

将Storage的数据都插入Filter，在Filter中查询都不存在时，那就不需要去Storage查询了。

当False Position出现时，只是会导致一次多余的Storage查询。

- 网络应用

P2P网络中查找资源操作，可以对每条网络通路保存Bloom Filter，当命中时，则选择该通路访问。

广播消息时，可以检测某个IP是否已发包。

检测广播消息包的环路，将Bloom Filter保存在包里，每个节点将自己添加入Bloom Filter。

信息队列管理，使用Counter Bloom Filter管理信息流量。

- **数据字典**

**垃圾邮件地址过滤**

来自于Google黑板报的例子。

**英文的拼写检查**

将词库建成一个bloomfilter

谢谢!

千腾网

www.specl.com

网聚全球特产