学习笔记

位运算

异或操作的一些特点

x^0 = x

x^1s = ~x // 注意1s = ~0

x^(~x)=1s

x^x = 0

c = a^b => a^c = b => b^c = a // 交换两个数

a^b^c = a^(b^c) = (a^b)^c

常用的位运算

- 判断奇偶
- 除以2
- 清零最低位的1，n & (n-1) 快速做**2的幂**的方法，n>0 and n&n-1 == 0
- 得到最低位的1 n & (-n)
- x&~x = 0

注意位运算在运算符中的优先级较低，经常要用括号包起来

例题：

N皇后终极解法，cols, pie, na

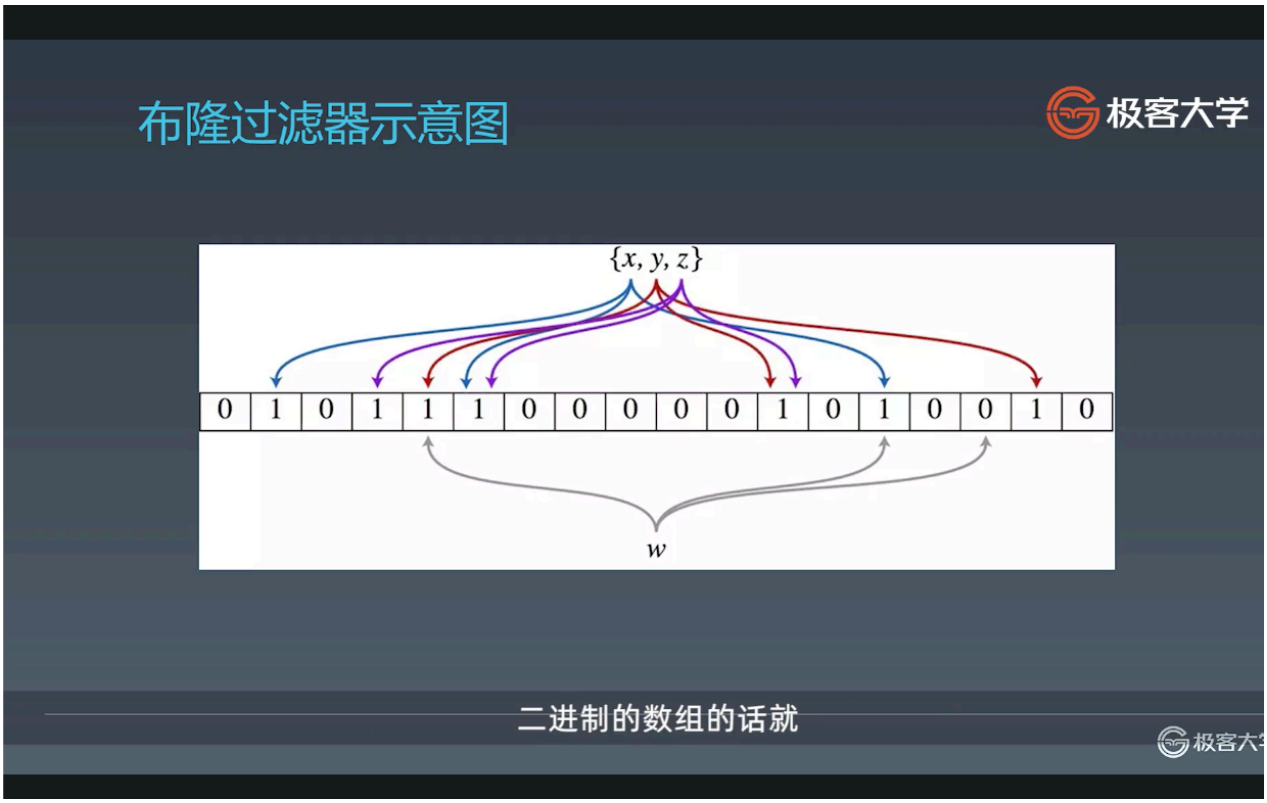
布隆过滤器

bloom filter vs hash table

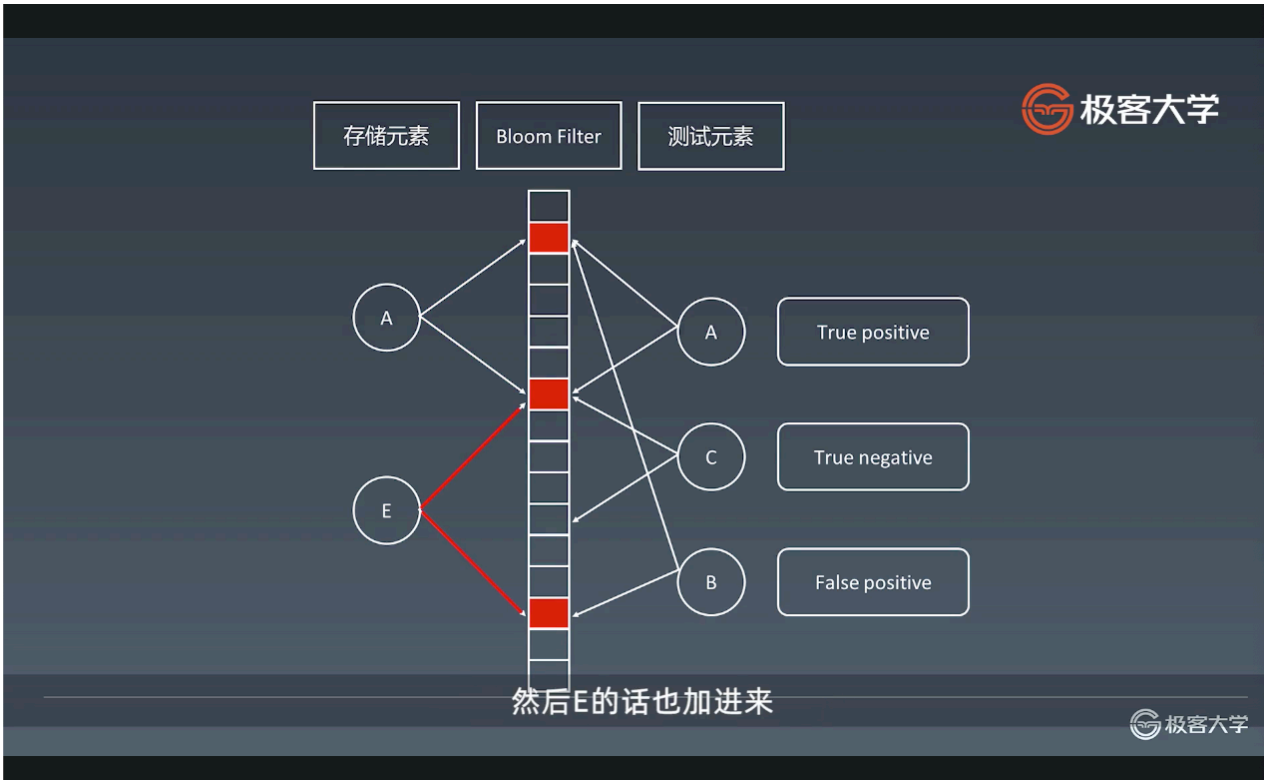
一个很长的**二进制**向量和一系列**随机映射函数**。布隆过滤器可以用于检索一个元素是否在一个集合中。

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**。



一定的误识别率



对于测试元素，当它验证这个元素所对应的二进制位是1的时候，那么它可能存在在布隆过滤器里面，当它验证这个元素所对应的二进制位只要有一个不为1的话，那么我们可以百分之百肯定它不在。

那么接下来要怎么判断它到底是否存在？布隆过滤器只是放在最外面当一个缓存使的，当一个很快速的判断使的。当B查到了之后，布隆过滤器里面是存在的，那么B会继续到这台机器的DB里面去查。C就不用查了。

案例

1. 比特币网络
2. 分布式系统（Map-Reduce）- Hadoop、search engine
    1. 搜索引擎，经常需要把大量的网页信息，图片信息存到整个服务器里面，一般来说，不同的网页是存在不同的集群里面的。那么就先去这个集群的布隆过滤器里面查一下。
3. Redis缓存
4. 垃圾邮件、评论等的过滤

LRU Cache

排序

初级排序-O(n^2)

1. 选择排序（Selection Sort)

   每次找最小值，然后放到待排序数组的起始位置。

2. 插入排序（Insertion Sort）

   从前往后逐步构建有序序列；对于未排序数据，在已排序序列中从后往前扫描，找打相应位置并插入。

3. 冒泡排序（Bubble Sort）

   嵌套循环，每次查看相邻的元素，如果逆序，则交换。

高级排序-O(nlogn)

- 快速排序（Quick Sort）
    - 数组取标杆pivot，将小元素放picot左边，大元素放右侧，然后依次对左边和右边的子数组继续快排；已达到整个序列有序。

      > 注意：正常情况下数组的prepend的操作的时间复杂度是O(n)，但是可以进行特殊化到O(1)。采用的方式是申请稍大一些的内存空间，然后在数组的最开始预留一部分空间，然后prepend的操作则是把头下标前移一个位置即可。

- 归并排序（Merge Sort） - 分治
    1. 把长度为n的输入序列分成两个长度为n/2的子序列；
    2. 对这两个子序列分别采用归并排序；
    3. 将两个排序好的子序列合并成一个最终的排序序列。

归并 和 快排 具有相似性，但步骤顺序相反

归并：先排序左右子数组，然后合并两个有序子数组

快排：先调配出左右子数组，然后对于左右子数组进行排序

- 堆排序（Heap Sort）- 堆插入O(logN），取最大/小值O(1)
    1. 数组元素一次建立小顶堆
    2. 依次取堆顶元素，并删除

特殊排序

- 计数排序（Counting Sort）

  计数排序要求输入的数据必须是有确定范围的整数。将输入的数据值转化为键存储在额外开辟的数组空间中；然后依次把计数大于1的填充回原数组

- 桶排序（Bucket Sort）

  桶排序的工作原理：假设数据服从均匀分布，将数据分到有限数量的桶里，每个桶再分别排序（有可能再使用别的排序算法或是以递归方式继续使用桶排序进行排序）。

- 基数排序（Radix Sort）

  基数排序是按照低位先排序，然后收集；再按照高位排序，然后再收集；以此类推，直到最高位。有时候有些属性是有优先级顺序的，先按低优先级排序，再按高优先级排序。