Part16-Bloom filters

Key videos	https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/riKfa/bloom-filters-the-basics https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/QSHNY/bloom-filters-heuristic-analysis
Note	Introduction to the implementation and performance of bloom filters, which are like hash tables except that they are insanely space-efficient and occasionally suffer from false positives.
Period	@2020/05/02

Note

▼ Bloom Filters: The Basics

- 优缺点：当数据量剧增时，但同时仅需作为过滤器的应用

  比哈希表空间复杂度更低，more space efficient

  缺点是**有一定的误识别率和删除困难**。

  ### Bloom Filters: Supported Operations

  Raison Dêtre: fast Inserts and Lookups.

  Comparison to Hash Tables:
  Pros: more space efficient.
  Cons:
  1) can’t store an associated object
  2) No deletions
  3) Small false positive probability
  (i.e., might say x has been inserted even though it hasn’t been)

- 应用：拼写检查，禁用密码序列等等

Original: early spellcheckers.

Canonical: list of forbidden passwords

Modern: network routers.

- Limited memory, need to be super-fast

- 组成部分与相关操作

  由bit组成的数列（其中|S|为每个对象大小）+k个哈希函数

  插入与查找功能 ⇒ 没有漏报，但是有一定概率的误报（就算之前没插入过，但可能因为hash函数对应的值相同，而有可能导致存在已有的假象）

  Ingredients: 1) array of n bits ($So \frac{n}{|S|} = \# \ of \ bits \ per \ object \ in \ data \ set \ S$)

  2) k hash functions $h_1, \ldots, h_k$ (k = small constant)

  Insert(x): for i = 1,2,...,k (whether or not bit already set ot 1)

  set $A[h_i(x)]=1$

  Lookup(x): return TRUE ⇔ $A[h_i(x)] = 1$ for every I = 1,2,....,k.

  Note: no false negatives. (if x was inserted, Lookup (x) guaranteed to succeed)

  But: false positive if all k $h_i(x)$'s already set to 1 by other insertions.

▼ Bloom Filters: Heuristic Analysis

- heuristic analysis 启发性分析

  在正确率与查询速度之间找到一个权衡

  Intuition: should be a trade-off between space and error (false positive) probability.

  Assume: [not justified] all $h_i(x)$'s uniformly random and independent (across different i's and x's).

  Setup: n bits, insert data set S into bloom filter.

  Note: for each bit of A, the probability it's been set to I is (under above assumption):

- 科学计算下的误报率

Under the heuristic assumption, what is the probability that a given bit of the bloom filter (the first bit, say) has been set to 1 after the data set S has been inserted?

- ○ $(1-1/n)^{k|S|}$
- ◉ $1-(1-1/n)^{k|S|}$

Correct

进一步化简

Intuition: should be a trade-off between space and error (false positive) probability.
Assume: [not justified] all $h_i(x)$'s uniformly random and independent (across different i's and x's).
Setup: n bits, insert data set S into bloom filter.
Note: for each bit of A, the probability it's been set to I is (under above assumption):

$$1-(1-\frac{1}{n})^{k|S|} \le 1-e^{-\frac{k|S|}{n}} = 1-e^{-\frac{k}{b}}$$

b = # of bits per object (n/|S|)

- 简化式推导

Heuristic Analysis (con'd)

Story so far: probability a given bit is 1 is $\le 1-e^{\frac{-k}{b}}$
So: under assumption, for x not in S, false positive probability is $\boxed{\le [1-e^{\frac{-k}{b}}]^k}$
where b = # of bits per object.

error rank ϵ

How to set k?: for fixed b, ϵ is minimized by setting

$k \approx (ln2).b \approx 0.693$

Plugging back in: $\epsilon \approx (\frac{1}{2})^{(ln2)b}$ or $b \approx 1.44 log_2 \frac{1}{\epsilon}$

(exponentially small in b)

Ex: with b = 8, choose k = 5 or 6 , error probability only approximately 2%.

Reference

- Bloom filters 布隆过滤器

布隆过滤器
布隆过滤器（英語：Bloom Filter）是1970年由布隆提出的。它实际上是一个很长的 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。它的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。 如果想判断一个元素是不是在一
https://zh.wikipedia.org/wiki/%E5%B8%83%E9%9A%86%E8%BF%87%E6%BB%A4%E5%99%A8

https://en.wikipedia.org/wiki/Bloom_filter

- 示例

Bloom Filters by Example
A Bloom filter is a data structure designed to tell you, rapidly and memory-efficiently, whether an element is present in a set. The price paid for this efficiency is that a
https://llimllib.github.io/bloomfilter-tutorial/