Data Structures

Bloom Filters

Design and Analysis
of Algorithms I

Bloom Filters: Supported Operations

Raison Dêtre: fast Inserts and Lookups. → Same as hash tables

Comparison to Hash Tables:

doesn't matter whether hash table has been implemented via chaining or open addressing
Still better than hash table

Pros: more space efficient. requires less space than the object itself

Cons: for applications where you just want to remember what values you have seen. you are not trying to store key value pairs

aka hash set

1) can't store an associated object Only remembers what kind of values you have seen
Doesn't store objects, or even pointers to objects

2) No deletions vanilla bloom filters described here don't have deletion,
but better bloom filters exist which can support deletion.
In general, deletion similar to deletion in hash tables with open addressing

3) Small false positive probability false negative probability is hard 0

(i.e., might say x has been inserted even though it hasn't been)

Bloom Filters: Applications

Original: early spellcheckers.

Canonical: list of forbidden passwords

Modern: network routers.
- Limited memory, need to be super-fast

assume this ratio to be 8 for now.
The object itself might be large.
For eg. an IP address is 32 bits wide

Bloom Filter: Under the Hood

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

In pratice, can pick just 2 hash functions and then create their k copies

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 $A[h_i(x)]=1$

for every new object x, run the hash function k times;
and set those k bits in array A to 1
(irrespective of whether that bit was previously 0 or 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.

For lookup
We evaluate h_i(x) for all k
All of them must be 1 for possibility
of x to be in bloom filter

Heuristic Analysis

more space—> less error

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):

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?

- $(1 - 1/n)^{k|S|}$ — prob 1st bit = 0
- $1 - (1 - 1/n)^{k|S|}$ — prob 1st bit = 1
- $(1/n)^{|S|}$
- $(1 - 1/n)^{|S|}$

Heuristic Analysis

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):

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

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

Heuristic Analysis (con'd)

Story so far: probability a given bit is 1 is $\leq 1 - e^{\frac{-k}{b}}$

So: under assumption, for x not in S, false positive probability is $\leq [1 - e^{\frac{-k}{b}}]^k$

where b = # of bits per object.

error rank ϵ

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

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

(exponentially small in b)

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

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