Bloom filters

Introduction

How to store a set, defined with following methods:

```
set.add(x)
set.remove(x)
set.contains(x)
```

Ideas:

1. Bit set (every bit for a specific property), efficient, can only store pre defined fixed numbers
2. Hash table (hash sets)
   Tree (balanced)
   - Fast O(1)
   - Require a lot of memory, objects are complex
3. Trie (Prefix trees, for strings)
4. Bloom filters
   if $x \in S$, return *yes*
   if $x \notin S$, return with $P_{no} \to 1$, $P_{yes} \to 0$
   - Do not support removing elements

Naive bloom filters

(Assumption: set is sparse, compared to element universe u.)

We want to store m elements, with k bits per element.
Let size $n = mk$ (total size in bits)

T: Bit set of size n (the underlying storage used to represent values)
h: hash function (from universal hash family / ROM), where $h : u \to \{0, \dots, n\}$

Define the following methods for set:

```
def add(x):
    id = h(x)
    T[id] = 1

def test(x):
    id = h(x)
    return T[id]
```

Performance definition

lemma:
Let $x_1, \dots, x_m$ be distinct elements inserted in the naive bloom filter, y is an element not in $\{x_1, \dots, x_m\}$)
then:

$$P\{test(y)=1\} <= 1/k$$

Proof

proof:

$$test(y)=1 \Leftrightarrow \exists i, h(x_i)=h(y)$$

then:

$$\begin{aligned} P\{test(y)=1\} &= P(\bigcup_i h(x_i)=h(y)) \\ &= \sum_{i=1}^{m} P\{h(x_i)=h(y)\} \\ &\le \frac{1}{n} * m \\ &= \frac{1}{m*k} * m \\ &= \frac{1}{k} \end{aligned}$$

Bloom filters

$h_1,\dots,h_k$: hash functions (ROM, independent from each other), where
$h_1,\dots,h_k : u \to \{0,\dots,n\}$
Let size $n=mk$

Define the following methods for set:

```
def add(x):
    for i = 1...k:
        T[h_i(x)] = 1

def test(x):
    return all(T[h_i(x)] = 1 for all i)
```

Performance definition

lemma:
Let $x_1,\dots,x_m$ be distinct elements inserted in the naive bloom filter, y is an element not in $\{x_1,\dots,x_m\}$)
then:

$$P\{test(y)=1\} \le (1-1/e)^k$$

Informal proof

Proof:

$$P\{test(y)=1\} = P\{\forall i,\ T[h_i(y)]=1\}$$

The RHS of this equation could be decomposed into:

$$
\begin{aligned}
&P\{\forall i,\ T[h_i(y)] = 1\} \\
&= \prod_{i=1}^{k} P\{T[h_i(y)] = 1\} \\
&= (P\{T[h(y)] = 1\})^k
\end{aligned}
$$

Note: The independence assumption of this transformation is **incorrect**, we will cover this part in the next section

Therefore, the probability of one of k hash values of y having no collision in table T is:

$$
\begin{aligned}
&P\{T[h(y)] = 0\} \\
&= P\{\forall i, j(i = 1...k, j = 1...m), \quad h_i(y)! = h_i(x_j)\} \\
&= (1 - \frac{1}{n})^{mk} \\
&= (1 - \frac{1}{mk})^{mk} \\
&= 1/e \qquad (\text{when} \lim_{mk \to \infty})
\end{aligned}
$$

With this result we can compute the value of probability $P\{\forall i,\ T[h_i(y)] = 1\}$:

$$
\begin{aligned}
&P\{\forall i,\ T[h_i(y)] = 1\} \\
&= \prod_{i=1}^{k} P\{T[h_i(y)] = 1\} \\
&= (P\{T[h(y)] = 1\})^k \\
\\
&\text{Note: O}(1) \text{ is a remainder, since } mk \text{ is not } \infty \\
&= (1 - \frac{1}{e} + O(1))^k
\end{aligned}
$$

Therefore the upper bound of the probability of a false positive (where all $T[h_i(y)] = 1$) is:

$$
P\{\text{false positive}\} \leq (1 - 1/e)^k
$$

Why informal proof is incorrect

More detailed analysis could be found in this paper: [On the false-positive rate of bloom filters]

Lets enumerate all positive conditions in a simple example, where $m = 1, k = 2, n = mk = 2$

Let A be the hash value of h_1, B be the hash value of h_2, we have the following combination table:

bits in T		bits of h(y)	
B	A	B	A
B	A	A	B
B	A	AB	-
B	A	-	AB
A	B	B	A
A	B	A	B
A	B	AB	-
A	B	-	AB
AB	-	B	A
AB	-	A	B
AB	-	AB	-
AB	-	-	AB
-	AB	B	A
-	AB	A	B
-	AB	AB	-
-	AB	-	AB

As we can see:

$$P\{h_1(y) = 1\} = P\{h_2(y) = 1\} = 12/16 = 3/4$$
$$P\{h_1(y) = 1, h_2(y) = 1\} = 10/16 = 5/8$$

These two probabilities are not independent in this case, because when table T is fully occupied, $P\{h_1(y) = 1, h_2(y) = 1\}$ becomes larger, in the upper 8 rows, this probability is 1, while in the lower eight rows, this probability is $1/4$, which is exactly $1/2 * 1/2$, and $P\{h_1(y) = 1\} = P\{h_2(y) = 1\} = 1/2$ in the lower eight rows, so independence only stands when the load of table T is not close to 1.

However, please take note that **dependence** only exists between $P\{h_i(y) = 1\}$ because we are **not informed** about how bit table T is occupied. When T is given, probabilities $P\{h_i(y) = 1|T\}$ are **independent**, because $h_i(y)$ are independent from each other, and they are also independent from T. We are going to use this independence to construct the correct, formal proof.

Formal proof

Let $n = 2mk$ in this case (In fact, 2 could be any number larger than $\frac{e}{e-1} \simeq 1.582$)

Theorem:

After inserting $x_1, \ldots, x_m$ in Bloom filter, the upper bound of false positive probability is constrained by:

$$P\{test(y)=1\} <= \frac{1}{2^k} = \frac{1}{2^{n/2m}} < (1-1/e)^{n/2m}$$

Proof:
Let $\mathcal{S}$ be the set of bits set after we insert $x_1, \ldots, x_m$, there is inequality:

$$|S| <= mk = n/2$$

For false positive probability there is:

$$\begin{aligned}
&P\{test(y)=1\} \\
&= P\{\forall i,\ T[h_i(y)] = 1\} \\
&= P\{h_1(y), h_2(y), \ldots h_k(y) \in \mathcal{S}\} \\
&= \mathbb{E}_S[P\{h_1(y), \ldots, h_k(y)\ in\ \mathcal{S}|\mathcal{S} = S\}] \quad \text{(expectation of probability when given S)} \\
\\
&\text{if the second term below, i.e. the conditional probability is proven} \leq 1/2^k \\
&= \sum_{S'} P\{S = S'\} * P\{h_1(y), \ldots, h_k(y) \in \mathcal{S}|\mathcal{S} = S\} \\
&\text{then:} \\
&\leq 1/2^k
\end{aligned}$$

Now goal becomes proving: $P\{h_1(y), \ldots, h_k(y) \in \mathcal{S}|\mathcal{S} = S\} \leq 1/2^k$

$$\begin{aligned}
&P\{h_i(y) \in \mathcal{S}, \forall i | \mathcal{S} = S\} \\
&\text{(Note: These probabilities are independent, as mentioned above)} \\
&= \prod_{i=1}^{k} P\{h_i(y) \in S|S\} \\
&= \prod_{i=1}^{k} \frac{|\mathcal{S}|}{n} \\
&\leq 1/2^k
\end{aligned}$$

This concludes our proof.