TDT4225
Storage and Management of Large Data Volumes

Chapter 7: SPECIAL ACCESS METHODS

Bloom filter (0)

- Stored records set traces (bits) in bit vector
- x independent hash functions set x bits for each key (record) which is stored.
-
- At search time for key:
- Record is stored only if all x bits for key are set

Bloom filter (1)

- Stored records set traces (bits) in bit vector
- x independent hash functions set x bits for each key (record) which is stored.
-
- Choice of x, how many bits should be set?
- Large x value, the filter is filled with many bits, the probability for hitting 1-bit increases.
- Large x, many tests increase the probability for one of them to hit a 0-bit

Bloom filter (2) – finding the optimal number of probes

Bloom filter is a bit vector,
stored records put their signature in the bit vector

After N records are placed in file, number of bits will still be 0. This share is $p_0 = \left(1 - \frac{1}{V}\right)^{Nx}$. For record not belonging to file to sneak through, it must hit x at positions when crawling x hash functions. Probability it will succeed is: $p_1(x) = (1 - p_0)^x$ or substituted:

$$p_1(x) = \left[1 - \left(1 - \frac{1}{V}\right)^{Nx}\right]^x \qquad (7.2)$$

Bloom filter (3)

Answer can be found by differentiating equation 7.2 with respect to x. To facilitate derivation, one rewrites equation 7.2:

$$\begin{aligned} p_1(x) &= e^{x \ln\left[1 - e^{Nx \ln\left(1-\frac{1}{V}\right)}\right]} \\ p_1(x) &= e^{x \ln(1-e^{rx})} \end{aligned} \tag{7.3}$$

after substituting: $r = N \ln\left(1 - \frac{1}{V}\right)$. Differentiation gives:

$$\frac{d(p_1(x))}{dx} = \ln(1 - e^{rx}) \times e^{x \ln(1-e^{rx})} \times \left[\frac{d}{dx}\left(x \ln\left(1 - e^{rx}\right)\right)\right] = 0 \tag{7.4}$$

Zero only if last term equal zero!!

Bloom filter (4)

$$\begin{aligned} \frac{d}{dx}\left(x \ln\left(1 - e^{rx}\right)\right) &= 0 \\ \ln\left(1 - e^{rx}\right) - \frac{rxe^{rx}}{1 - e^{rx}} &= 0, \text{ we insert: } z = rx \\ \ln\left(1 - e^{z}\right) - \frac{ze^{z}}{1 - e^{z}} &= 0, \text{ or rewritten:} \\ \ln\left(1 - e^{z}\right)\left(1 - e^{z}\right) &= ze^{z}, \text{ we set: } u = \ln(1 - e^{z}), \text{ which gives:} \\ ue^{u} &= ze^{z} \end{aligned}$$

Bloom filter (5)

This follows that $u = z$. It is inserted into expression for u and one gets:

$$\begin{aligned} z &= \ln(1 - e^z) \\ e^z &= 1 - e^z \\ 2e^z &= 1 \\ z &= -\ln 2 \end{aligned}$$

An interesting feature of solution is there is a value for z, independent of all other parameters. [2] By back substitution, one gets:

$$x = \frac{z}{r} = -\frac{\ln 2}{N \ln\left(1 - \frac{1}{V}\right)} \qquad (7.5)$$

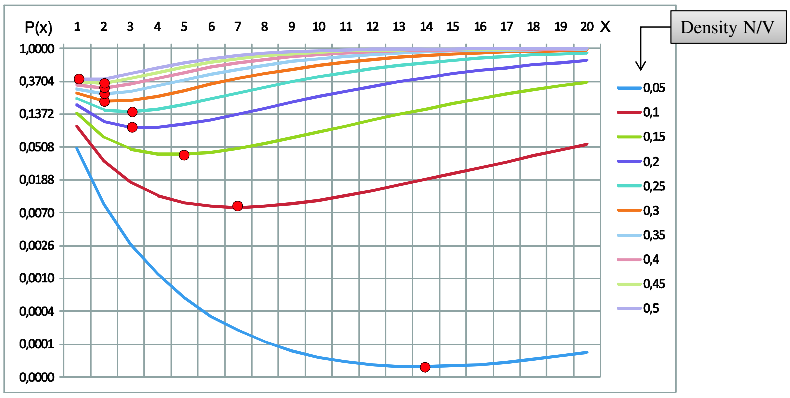
Matematisk løsning funnet av daværenede student – nå PhD Martin Thorsen Ranang

Bloom filter (6)

x_{opt}	$\delta = \frac{N}{V}$	N	x=actual number of hash functions							
			1	2	3	4	5	6	7	8
14	0.050	10 000	0.0488	0.0091	0.0027	0.0011	0.00053	0.0003	0.0002	0.0001
7	0.100	20 000	0.0952	0.0329	0.0174	0.0118	0.00943	0.0084	**0.0082**	0.0085
5	0.150	30 000	0.1393	0.0672	0.0476	0.0414	**0.0409**	0.04367	0.0491	0.0569
3	0.200	40 000	0.1813	0.1087	**0.0919**	0.0920	0.1009	0.11645	0.1378	0.1646
3	0.250	50 000	0.2212	0.1548	**0.1469**	0.1597	0.1849	0.21983	0.2628	0.3125
2	0.300	60 000	0.2592	**0.2036**	0.2090	0.2385	0.2830	0.33821	0.4008	0.4673
2	0.350	70 000	0.2953	**0.2534**	0.2747	0.3222	0.3850	0.45668	0.5317	0.6054
2	0.400	80 000	0.3297	**0.3032**	0.3413	0.4057	0.4833	0.56519	0.6446	0.7168
2	0.450	90 000	0.3624	**0.3522**	0.4065	0.4854	0.5730	0.65874	0.7360	0.8012
1	0.500	100 000	**0.3935**	0.3996	0.4689	0.5590	0.6517	0.73608	0.8068	0.8625

Filter size: 200 000 bits
N: number of records stored
x: independent hash functions
Values: probability of false match

Bloom filter (7) –discrimination capacity



x-axis is number of independent hash functions
Lower value is better.
Red dot denotes minimum

Bloom filter (8), various properties

- The filter is built during inserts
- A filter may not be updated by deleting bits
- After many deletes or changed key values the filter will contain too many false bits
- A *worn-out* filter should be recreated

Bloom filter (9), distributed filters

- Multiple filters in tandem
- X independent filters may be tested in random order
- Do not need all filters in RAM at the same time
- Keys which have passed may be tested on new filter sets (read from disk)

Bloom filter (10), applications

- Should be used:
- All places where the search key has a possibility of not being present:
  Early elimination of non-candidates
  - False/invalid credit card numbers
  - Overflow storage
  - Duplicate control
  - Relational algebra operations (matching keys)
  - …