Bloom Filter Description and Derivation

A Bloom Filter is a Statistical Data Structure

It provides space and time efficient indexing of list membership.

It replaces linear, binary, or tree searches. However, it is statistical. Occasionally, the look-up process can fail and attribute membership in the list when the item is not in the list. This is a false alarm. The developer can control this by designing the data structure tailored to the need or a specification.

The major pieces of the Bloom Filter are [L,H,X] as described below with the dimensions being, [m,k,n]. There are m elements in the List, k Hash functions performed on every element in the List and n bits in the X bit vector.

$$
\begin{aligned}
&L \rightarrow \textit{List, normally a string array} \\
&\quad L_l \rightarrow l^{th}\ \textit{element} \in \textit{List} \\
&H \rightarrow \textit{Hash function } H() \\
&\quad H_h \rightarrow h^{th}\ \textit{function hash} \\
&X \rightarrow \textit{Bit Vector dimension } \underline{n} \\
&\quad X_i \rightarrow i^{th}\ \textit{element} \in \textit{Bit Vector Array}
\end{aligned}
$$

$$X_i = \{0 \vee 1\}$$

$$X_i = \begin{Bmatrix} X[H_h(L_l)] = 1, \\ X[\overline{H_h(L_l)}] = 0 \end{Bmatrix}$$

$$
\begin{aligned}
&\textit{where } H, h, l, i \in Z, \\
&\quad 0 \leq H_h(L_l) \leq n-1, \\
&\quad 1 \leq h \leq \underline{k}, \\
&\quad 1 \leq l \leq \underline{m},
\end{aligned}
$$

The development process is relatively easy.

1) Calculate the Expected value of the Bit Vector Sum as you Hash the List.
2) Determine the biased coin probability of a position being hashed
3) Calculate the biased coin probability of a false alarm

Use this to design your Bloom Filter

1) Given the Probability of False Alarm and the length of your list,
2) Calculate n as a function of k
3) Choose the optimal value

When you implement your design, then you can calculate the Final Probability of False Alarm by Replacing the expected theoretical Bit Vector Sum with the actual.

1) Calculate the expected value of the bit vector sum.

$$E\left(\left(\sum_{i=0}^{N-1} X_i\right) / H_{km}\right) = E_{km}$$

Recursively,

$$E_{km} = E_{km-1} + X_i \, P(H_i)$$

$$= E_{km-1} + \frac{N - E_{km-1}}{N}$$

← Empty positions Left / Total positions in vector

$$= (1 - 1/N) E_{km-1} + 1$$

Let $\underline{\underline{c = (1 - 1/N)}}$ + $E_1 = 1$

$$E_2 = c + 1$$

$$E_3 = c(c+1) + 1 = c^2 + c + 1$$

$$E_4 = c(c^2 + c + 1) + 1 = c^3 + c^2 + c + 1$$

$$\vdots$$

$$E_{km} = \sum_{i=0}^{km-1} c^i$$

← We solved this in class

$$= \frac{1 - c^{km}}{1 - c}$$

2) The biased coin probability of a position being hashed

$$P_{km} = \frac{E_{km}}{N}$$

$$= \frac{1 - C^{km}}{N(1-C)} = \frac{(1 - C^{km})}{N(1-(1-1/N))}$$

$$= \underline{\underline{1 - C^{km}}}$$

3) The probability of false alarm is just a biased coin thrown k times

$$P_{fa} = (P_{km})^k = (1 - C^{km})^k$$

$$= (1 - (1-1/N)^{km})^k$$

4) Solve for N given m and k. Choose the desired P_{fa}

$$N = (1 - (1 - P_{fa}^{1/k})^{1/km})^{-1}$$

↑ This is size of the bit vector

5) The actual false alarm

$$P_{fa}' = \left(\frac{\sum_{i=0}^{N-1} x_i}{N} \right)^k$$

P_{fa} calculated $\neq P_{fa}'$

Find Optimal N given K and Pfa

5.000000% = Pfa → Probability of False Alarm
10 = m → Size of the List

k # Hashes	Pfa^1/k	(1-I)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element
1	0.050000	0.994884	195	19.55	2.44
2	0.223607	0.987425	80	7.95	0.99
3	0.368403	0.984800	66	6.58	0.82
4	0.472871	0.984120	63	6.30	0.79
5	0.549280	0.984188	63	6.32	0.79
6	0.606962	0.984556	65	6.48	0.81
7	0.651836	0.985040	67	6.68	0.84
8	0.687656	0.985560	69	6.93	0.87
9	0.716871	0.986077	72	7.18	0.90
10	0.741134	0.986576	74	7.45	0.93
11	0.761596	0.987050	77	7.72	0.97
12	0.779078	0.987496	80	8.00	1.00
13	0.794183	0.987914	83	8.27	1.03
14	0.807364	0.988305	86	8.55	1.07
15	0.818964	0.988671	88	8.83	1.10
16	0.829250	0.989014	91	9.10	1.14
17	0.838434	0.989335	94	9.38	1.17
18	0.846682	0.989636	96	9.65	1.21
19	0.854131	0.989919	99	9.92	1.24
20	0.860892	0.990186	102	10.19	1.27
21	0.867054	0.990437	105	10.46	1.31
22	0.872695	0.990675	107	10.72	1.34
23	0.877877	0.990899	110	10.99	1.37
24	0.882654	0.991112	113	11.25	1.41
25	0.887072	0.991314	115	11.51	1.44
26	0.891170	0.991506	118	11.77	1.47
27	0.894981	0.991688	120	12.03	1.50
28	0.898534	0.991862	123	12.29	1.54
29	0.901855	0.992027	125	12.54	1.57
30	0.904966	0.992186	128	12.80	1.60
31	0.907886	0.992337	130	13.05	1.63
32	0.910632	0.992482	133	13.30	1.66

1 Byte

Optimal k

Find Optimal N given K and Pfa

0.100000% = Pfa → Probability of False Alarm
10 = m → Size of the List

k # Hashes	Pfa^1/k	(1-l)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element	
1	0.001000	0.999900	9995	999.55	124.94	
2	0.031623	0.998395	623	62.29	7.79	
3	0.100000	0.996494	285	28.52	3.57	
4	0.177828	0.995117	205	20.48	2.56	
5	0.251189	0.994231	173	17.34	2.17	
6	0.316228	0.993685	158	15.83	1.98	2 bytes
7	0.372759	0.993359	151	15.06	1.88	
8	0.421697	0.993178	147	14.66	1.83	
9	0.464159	0.993092	145	14.48	1.81	
10	0.501187	0.993069	144	14.43	1.80	Optimal k
11	0.533670	0.993089	145	14.47	1.81	
12	0.562341	0.993138	146	14.57	1.82	
13	0.587802	0.993206	147	14.72	1.84	
14	0.610540	0.993287	149	14.90	1.86	
15	0.630957	0.993376	151	15.10	1.89	
16	0.649382	0.993471	153	15.32	1.91	
17	0.666085	0.993569	155	15.55	1.94	
18	0.681292	0.993667	158	15.79	1.97	
19	0.695193	0.993766	160	16.04	2.01	
20	0.707946	0.993865	163	16.30	2.04	
21	0.719686	0.993962	166	16.56	2.07	
22	0.730527	0.994057	168	16.83	2.10	
23	0.740568	0.994151	171	17.10	2.14	
24	0.749894	0.994242	174	17.37	2.17	
25	0.758578	0.994331	176	17.64	2.21	
26	0.766682	0.994418	179	17.92	2.24	
27	0.774264	0.994503	182	18.19	2.27	
28	0.781371	0.994585	185	18.47	2.31	
29	0.788046	0.994665	187	18.74	2.34	
30	0.794328	0.994742	190	19.02	2.38	
31	0.800250	0.994818	193	19.30	2.41	
32	0.805842	0.994891	196	19.57	2.45	

Find Optimal N given K and Pfa

0.000100% = Pfa → Probability of False Alarm
10 = m → Size of the List

k # Hashes	Pfa^1/k	(1-l)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element	
1	0.000001	1.000000	9999995	999999.55	124999.94	
2	0.001000	0.999950	19990	1999.05	249.88	
3	0.010000	0.999665	2985	298.55	37.32	
4	0.031623	0.999197	1245	124.53	15.57	
5	0.063096	0.998697	768	76.77	9.60	
6	0.100000	0.998246	570	57.00	7.12	
7	0.138950	0.997865	468	46.84	5.86	
8	0.177828	0.997555	409	40.91	5.11	
9	0.215443	0.997308	371	37.14	4.64	
10	0.251189	0.997111	346	34.62	4.33	
11	0.284804	0.996957	329	32.87	4.11	
12	0.316228	0.996837	316	31.62	3.95	4 Bytes
13	0.345511	0.996745	307	30.72	3.84	
14	0.372759	0.996674	301	30.07	3.76	
15	0.398107	0.996621	296	29.60	3.70	
16	0.421697	0.996583	293	29.27	3.66	
17	0.443669	0.996557	290	29.04	3.63	
18	0.464159	0.996540	289	28.90	3.61	
19	0.483293	0.996531	288	28.83	3.60	
20	0.501187	0.996528	288	28.81	3.60	optimal k
21	0.517947	0.996531	288	28.83	3.60	
22	0.533670	0.996538	289	28.89	3.61	
23	0.548442	0.996549	290	28.98	3.62	
24	0.562341	0.996563	291	29.09	3.64	
25	0.575440	0.996579	292	29.23	3.65	
26	0.587802	0.996597	294	29.39	3.67	
27	0.599484	0.996617	296	29.56	3.69	
28	0.610540	0.996638	297	29.74	3.72	
29	0.621017	0.996660	299	29.94	3.74	
30	0.630957	0.996683	301	30.15	3.77	
31	0.640400	0.996706	304	30.36	3.80	
32	0.649382	0.996730	306	30.58	3.82	

Find Optimal N given K and Pfa

5.000000% = Pfa → Probability of False Alarm
1000 = m → Size of the List

k # Hashes	Pfa^1/k	(1-I)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element
1	0.050000	0.999949	19496	19.50	2.44
2	0.223607	0.999873	7903	7.90	0.99
3	0.368403	0.999847	6529	6.53	0.82
4	0.472871	0.999840	6247	6.25	0.78
5	0.549280	0.999841	6275	6.27	0.78
6	0.606962	0.999844	6426	6.43	0.80
7	0.651836	0.999849	6635	6.64	0.83
8	0.687656	0.999855	6875	6.88	0.86
9	0.716871	0.999860	7133	7.13	0.89
10	0.741134	0.999865	7400	7.40	0.92
11	0.761596	0.999870	7672	7.67	0.96
12	0.779078	0.999874	7948	7.95	0.99
13	0.794183	0.999878	8224	8.22	1.03
14	0.807364	0.999882	8501	8.50	1.06
15	0.818964	0.999886	8777	8.78	1.10
16	0.829250	0.999890	9053	9.05	1.13
17	0.838434	0.999893	9327	9.33	1.17
18	0.846682	0.999896	9599	9.60	1.20
19	0.854131	0.999899	9870	9.87	1.23
20	0.860892	0.999901	10140	10.14	1.27
21	0.867054	0.999904	10408	10.41	1.30
22	0.872695	0.999906	10674	10.67	1.33
23	0.877877	0.999909	10939	10.94	1.37
24	0.882654	0.999911	11202	11.20	1.40
25	0.887072	0.999913	11463	11.46	1.43
26	0.891170	0.999915	11723	11.72	1.47
27	0.894981	0.999917	11981	11.98	1.50
28	0.898534	0.999918	12238	12.24	1.53
29	0.901855	0.999920	12493	12.49	1.56
30	0.904966	0.999922	12747	12.75	1.59
31	0.907886	0.999923	13000	13.00	1.62
32	0.910632	0.999925	13251	13.25	1.66

1 Byte

optimal k

Find Optimal N given K and Pfa

0.100000% = Pfa → Probability of False Alarm
1000 = m → Size of the List

k # Hashes	Pfa^1/k	(1-l)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element	
1	0.001000	0.999999	999500	999.50	124.94	
2	0.031623	0.999984	62241	62.24	7.78	
3	0.100000	0.999965	28474	28.47	3.56	
4	0.177828	0.999951	20429	20.43	2.55	
5	0.251189	0.999942	17285	17.29	2.16	
6	0.316228	0.999937	15785	15.78	1.97	2 Bytes
7	0.372759	0.999933	15008	15.01	1.88	
8	0.421697	0.999932	14608	14.61	1.83	
9	0.464159	0.999931	14425	14.43	1.80	
10	0.501187	0.999930	14378	14.38	1.80	Optimal k
11	0.533670	0.999931	14420	14.42	1.80	
12	0.562341	0.999931	14523	14.52	1.82	
13	0.587802	0.999932	14669	14.67	1.83	
14	0.610540	0.999933	14847	14.85	1.86	
15	0.630957	0.999934	15048	15.05	1.88	
16	0.649382	0.999934	15267	15.27	1.91	
17	0.666085	0.999935	15499	15.50	1.94	
18	0.681292	0.999936	15742	15.74	1.97	
19	0.695193	0.999937	15993	15.99	2.00	
20	0.707946	0.999938	16250	16.25	2.03	
21	0.719686	0.999939	16512	16.51	2.06	
22	0.730527	0.999940	16778	16.78	2.10	
23	0.740568	0.999941	17047	17.05	2.13	
24	0.749894	0.999942	17318	17.32	2.16	
25	0.758578	0.999943	17591	17.59	2.20	
26	0.766682	0.999944	17866	17.87	2.23	
27	0.774264	0.999945	18141	18.14	2.27	
28	0.781371	0.999946	18417	18.42	2.30	
29	0.788046	0.999947	18693	18.69	2.34	
30	0.794328	0.999947	18970	18.97	2.37	
31	0.800250	0.999948	19247	19.25	2.41	
32	0.805842	0.999949	19524	19.52	2.44	

Find Optimal N given K and Pfa

0.000100% = Pfa → Probability of False Alarm
1000 = m → Size of the List

k # Hashes	Pfa^1/k	(1-l)^1/KM	Bit Vector Size → n	n/m Bits per element	n/m Bytes per element	
1	0.000001	1.000000	999999473	999999.47	124999.93	
2	0.001000	0.999999	1999000	1999.00	249.88	
3	0.010000	0.999997	298498	298.50	37.31	
4	0.031623	0.999992	124481	124.48	15.56	
5	0.063096	0.999987	76718	76.72	9.59	
6	0.100000	0.999982	56948	56.95	7.12	
7	0.138950	0.999979	46791	46.79	5.85	
8	0.177828	0.999976	40857	40.86	5.11	
9	0.215443	0.999973	37093	37.09	4.64	
10	0.251189	0.999971	34570	34.57	4.32	
11	0.284804	0.999970	32817	32.82	4.10	
12	0.316228	0.999968	31569	31.57	3.95	4 Bytes
13	0.345511	0.999967	30668	30.67	3.83	
14	0.372759	0.999967	30016	30.02	3.75	
15	0.398107	0.999966	29547	29.55	3.69	
16	0.421697	0.999966	29216	29.22	3.65	
17	0.443669	0.999966	28991	28.99	3.62	
18	0.464159	0.999965	28850	28.85	3.61	
19	0.483293	0.999965	28776	28.78	3.60	
20	0.501187	0.999965	28756	28.76	3.59	Optimal Q
21	0.517947	0.999965	28779	28.78	3.60	
22	0.533670	0.999965	28839	28.84	3.60	
23	0.548442	0.999965	28929	28.93	3.62	
24	0.562341	0.999966	29045	29.05	3.63	
25	0.575440	0.999966	29182	29.18	3.65	
26	0.587802	0.999966	29338	29.34	3.67	
27	0.599484	0.999966	29509	29.51	3.69	
28	0.610540	0.999966	29693	29.69	3.71	
29	0.621017	0.999967	29889	29.89	3.74	
30	0.630957	0.999967	30096	30.10	3.76	
31	0.640400	0.999967	30311	30.31	3.79	
32	0.649382	0.999967	30533	30.53	3.82	