Performance of 5 Different Hash Functions

Muhammad Zaeem Shahzad

Date: 5th December, 2021

Introduction - Hash Functions’ Descriptions

Hash Functions used:

1) ASCII Value Sum Hash Function:

The Hash Code generated for a word is the Sum of the ASCII values of every character in the word.

2) Polynomial Shift Hash Function:

This function uses Horner’s Rule. It takes into account the order of arrangement of the characters in the word using a polynomial function. It returns the Sum of ASCII values of every character in a word taking the order of characters into account.

3) Cyclic Shift Hash Function:

This function shifts the integer value of the character of a word by 5 bits. In this way, this function returns the Hash Code for a word which is the sum of integer values of all the characters (shifted by 5 bits) in the word.

4) Squaring ASCII Value Hash Function:

This function sums the squares of the ASCII values of every character in a word to generate its Hash Code.

5) XOR Hash Function:

This is a function I created experimentally to give one of the fewest collisions from the list of Hash Functions I used. It is similar to the Cyclic Shift Hash Function in that it shifts bits (by 6 digits for this function). However, this function employs the XOR operator unlike the OR used in the Cyclic Shift Hash Function. The result of the shift is XOR-ed with the character value before adding it to the sum to ensure even-distribution of the Hash Codes generated.

*The hash codes were compressed in all Hash Functions by taking the modulo of the hash codes with capacity of the HashTable.

Results

Filename	Unique Word Count	Total Word Count	Collisions with Hash Functions numbered 1 to 5 (as in list above)				
			1	2	3	4	5
373-0.txt	11101	87342	9845	512	519	1197	536
877-0.txt	2254	8190	1512	227	203	251	225
6120-0.txt	11584	92499	10149	485	519	1212	542
6073-0.txt	8411	86735	7169	253	314	715	327
6040.txt	7221	41233	6039	482	450	728	466
5737-0.txt	10264	95977	8917	410	381	918	372
5592.txt	9960	102319	8700	353	367	941	370
3254.txt	55471	1631556	53408	661	738	13393	740
3181-0.txt	2422	9838	1607	199	216	255	221
2781-0.txt	3879	32515	2858	168	155	316	186
2550-0.txt	9498	68241	8156	484	472	840	484
2518.txt	6594	51462	5510	315	288	582	331
2429-0.txt	5721	44762	4681	260	236	494	258
2334-0.txt	24790	306796	23042	750	764	3872	734
2327-8.txt	6093	42348	4876	311	315	441	303
2305-0.txt	9679	96362	8433	333	347	899	380
1982-0.txt	936	3281	312	88	195	95	93
1944-0.txt	7647	77804	6478	269	275	625	289
1626-0.txt	11071	140197	9629	320	304	954	312
25035.txt	2246	10261	1451	198	174	249	167
24878-8.txt	10527	110743	9198	370	348	969	372
24558.txt	2595	11270	1747	214	222	235	198
24313-8.txt	8963	78773	7612	406	376	771	393

23942-8.txt	1829	7465	1101	159	165	161	150
23210-0.txt	2160	9161	1403	173	191	206	173
23099.txt	1264	4468	643	115	114	145	111
22897-8.txt	2537	10965	1696	188	194	248	191
22662-8.txt	2184	7418	1430	212	222	255	209
22522-8.txt	5209	29497	4188	338	346	497	314
22426-8.txt	3947	22201	3008	219	282	295	253
21782.txt	2210	8202	1417	196	219	226	214
18776-8.txt	9409	62047	8079	547	492	1008	508
17669-8.txt	13571	109491	12149	614	558	1562	588
15717-8.txt	8965	66161	7623	453	410	872	396
14744-8.txt	7749	58353	6523	392	380	687	369
13799.txt	10439	85770	9074	447	431	1045	436
10947-8.txt	13506	108058	12013	623	601	1437	589
9790-8.txt	13794	95966	12238	730	720	1512	733
9629-8.txt	6268	51512	5090	265	297	499	272
9205.txt	1684	5567	1025	178	180	184	168
8933-0.txt	9943	181749	8780	196	198	1120	196
8129-8.txt	4986	39791	4042	230	223	477	228
6696-8.txt	13806	98011	12400	687	746	1828	695
6168.txt	3895	34132	3022	169	168	317	152
pg4081.txt	9919	51398	8653	650	685	1176	746
59368.txt	1591	6032	899	132	162	163	156
59255.txt	2339	10205	1536	192	176	237	182
58995-8.txt	1569	6463	921	117	152	155	140
58991.txt	2209	7827	1440	238	204	235	222
58743.txt	1849	8730	1175	135	137	180	152
58735.txt	1755	6690	1041	148	151	195	163
58341-0.txt	11121	76453	9819	623	615	1151	614
57040-0.txt	11899	69482	10575	744	785	1255	735
57006-0.txt	6999	44182	5856	401	392	620	410
56870-8.txt	8362	67164	7098	364	412	715	374
55865-0.txt	4550	36863	3579	200	227	424	198

55514-0.txt	13056	72309	11704	853	924	1744	828
54183-0.txt	8109	60845	6842	393	429	820	368
51752.txt	2145	11336	1386	141	126	191	134
51699.txt	1968	8854	1251	155	168	182	143
51687.txt	2115	9610	1338	161	184	226	165
51603.txt	1519	6987	857	126	122	142	119
51498.txt	1564	5700	891	148	159	186	147
51493.txt	1771	6400	1065	158	171	199	171
51296.txt	1834	9064	1131	109	124	172	137
51268.txt	2457	10408	1682	192	209	257	215
51193.txt	2311	9859	1536	211	211	212	184
51129.txt	2028	8600	1284	181	158	216	161
51008.txt	1460	5668	812	140	154	151	142
50877.txt	1875	7173	1149	169	170	185	180
49598-8.txt	6964	54678	5766	326	343	554	306
42664.txt	1676	6426	1045	142	146	165	149
41562.txt	2014	8347	1301	173	184	193	166
40745-8.txt	8081	84030	6840	287	268	618	290
39706.txt	2914	28519	2097	106	84	182	119
38531-8.txt	12965	133104	11468	466	461	1331	428
32735.txt	2354	8216	1567	241	229	211	215
32347.txt	1210	4706	620	115	100	146	124
38172-8.txt	12605	113813	11150	550	546	1259	527
34766-0.txt	12857	130956	11439	460	554	1462	494
34313-8.txt	7041	56355	5747	335	337	542	298
32845-8.txt	14409	127232	12937	618	633	1698	589
32133.txt	1788	7858	1107	135	143	186	148
32104.txt	2039	7912	1287	173	195	204	182
32078.txt	2245	9288	1477	178	206	252	191
25035.txt	2246	10261	1451	198	174	249	167
26772.txt	2720	11445	1862	237	226	259	234
32077.txt	1903	7312	1169	171	191	232	175
32067.txt	2271	9979	1462	169	181	237	199
32046-8.txt	15440	241475	13979	376	358	1878	372

32040.txt	2167	9509	1353	157	138	220	172
31840.txt	1718	7008	1027	167	139	194	159
31217-8.txt	14752	167367	13256	465	470	1680	486
30044.txt	1348	5262	731	133	120	141	117
28062.txt	1811	8808	1127	118	141	149	154
30029-8.txt	1741	7317	1014	141	144	185	137
29750.txt	1601	6097	920	151	158	146	145
29618.txt	1507	5615	844	128	146	150	130
29503.txt	1619	7144	951	109	115	161	133
28726-8.txt	12679	105618	11267	547	542	1394	536
28698.txt	2213	10477	1474	158	161	179	166
28650.txt	1492	6148	854	118	140	150	119
Totals	637046	6359073	534382	30128	30695	73529	30257

The totals are used in the next page to evaluate the results and analyze the performance of each Hash Function used in the program.

Analysis

1) Methods:

Based on the results displayed above, the performance of the Hash Functions implemented in the program can be analyzed by calculating two values for each function: the average number of collisions per word inserted, and the average number of collisions per file. The second calculated value is valid because all the Hash Functions were run over the same set of files.

The formula used to calculate the average number of collisions per word inserted for a Hash Function is:

$$\frac{Sum\ of\ the\ Number\ of\ Collisions\ by\ the\ Hash\ Function\ over\ all\ Files}{Sum\ of\ Unique\ Word\ Count\ of\ all\ Files}$$

By definition, a collision is when a Hash Function generates a code for a key that is the same as the code for a different key. The HashTable used in the program only stores entries that have unique keys (the value of the entry is its frequency which determines total word count over all entries). Furthermore, a collision occurs only when a word is inserted which gets mapped to an index in the HashTable where a different word already exists. Thus, the average number of collisions should be representative of a Hash Function’s ability to minimize mapping the same Hash Code to different keys. This is why the Sum of Unique Word count, and not Total Word Count, is used in the above formula.

*The calculated value is an estimate of the number of collisions a Hash Function will generate if a word is inserted into the HashTable.

The formula used to calculate the average number of collisions per file for a Hash Function is:

$$\frac{Sum\ of\ the\ Number\ of\ Collisions\ by\ the\ Hash\ Function\ over\ all\ Files}{Total\ Number\ of\ Files}$$

*There were a total of 101 test files. And every Hash Function was run on each file. This is why this value is an estimate of the number of collisions a Hash Function generated per test file.

2) Conclusion

Hash Function	Average Collisions per Word Inserted	Average Collisions over all Files
ASCII Value Sum Hash Function	0.83884	5291
Polynomial Shift Hash Function	0.04729	298
Cyclic Shift Hash Function	0.04818	304
Squaring ASCII Value Hash Function	0.11542	728
XOR Hash Function	0.04749	300

The Polynomial Shift Hash Function (Function 2 in the list on the first page) has the best overall performance when compared with the other Hash Functions. It has the least average-collisions-per-word-inserted value of 0.04729 which means it generates the fewest collisions on average per word inserted. Furthermore, it has the least average number of collisions across all the test files, which means it generated the fewest collisions across the 101 test files when compared to the other Hash Functions. This is why it is chosen as the default Hash Function to be used in the system if a user does not decide which Hash Function to use.