FIT1008 Introduction to Computer Science

Solo Prac 3

Name: Vihaan Philip

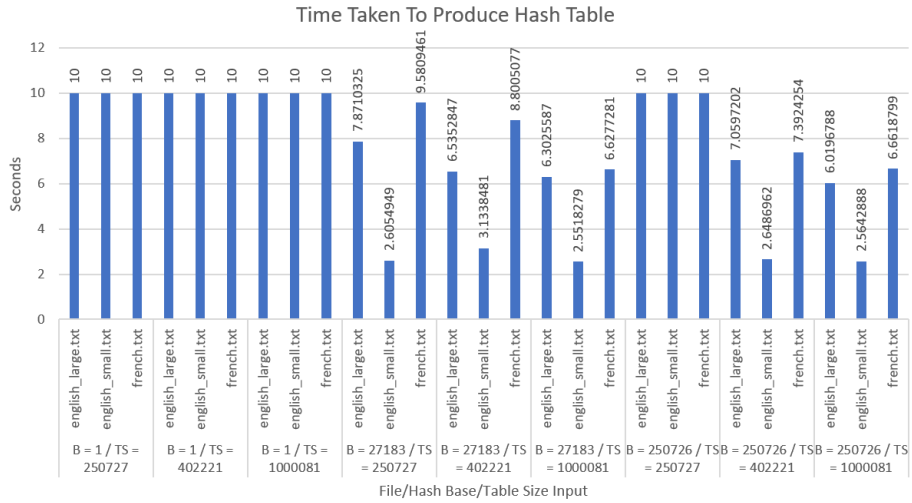
Student ID: 31862306

Due Date: 9th October 2021

The generated data formed:

	A	B	C	D	E	F	G	H	I	J
1	Hash Base	Table Size	Label	Filename	Word Count	Time	Conflicts	Probe Count	Probe Max	Rehash Count
2	1	250727	B = 1 / TS = 250727	english_large.txt	2906	10	2578	2644948	2786	0
3	1	250727		english_small.txt	2943	10	2583	2719659	2769	0
4	1	250727		french.txt	2978	10	2552	2647838	2669	0
5	1	402221	B = 1 / TS = 402221	english_large.txt	2931	10	2603	2699707	2827	0
6	1	402221		english_small.txt	2932	10	2572	2696380	2769	0
7	1	402221		french.txt	3018	10	2590	2730350	2669	0
8	1	1000081	B = 1 / TS = 1000081	english_large.txt	2963	10	2635	2769883	2827	0
9	1	1000081		english_small.txt	2922	10	2562	2676965	2747	0
10	1	1000081		french.txt	2962	10	2536	2611520	2669	0
11	27183	250727	B = 27183 / TS = 250727	english_large.txt	194433	7.8710325	76623	420606	278	0
12	27183	250727		english_small.txt	84097	2.6054949	14264	22580	23	0
13	27183	250727		french.txt	202358	9.5809461	85022	565917	299	0
14	27183	402221	B = 27183 / TS = 402221	english_large.txt	194433	6.5352847	47798	100107	43	0
15	27183	402221		english_small.txt	84097	3.1338481	9000	11736	13	0
16	27183	402221		french.txt	202358	8.8005077	53376	124793	57	0
17	27183	1000081	B = 27183 / TS = 1000081	english_large.txt	194433	6.3025587	19039	24442	20	0
18	27183	1000081		english_small.txt	84097	2.5518279	3501	3936	15	0
19	27183	1000081		french.txt	202358	6.6277281	22045	30701	21	0
20	250726	250727	B = 250726 / TS = 250727	english_large.txt	2445	10	2351	2707357	2372	0
21	250726	250727		english_small.txt	2414	10	2326	2659616	2360	0
22	250726	250727		french.txt	2852	10	2536	2651027	2639	0
23	250726	402221	B = 250726 / TS = 402221	english_large.txt	194433	7.0597202	47901	101110	51	0
24	250726	402221		english_small.txt	84097	2.6486962	9010	11798	10	0
25	250726	402221		french.txt	202358	7.3924254	53160	122422	49	0
26	250726	1000081	B = 250726 / TS = 1000081	english_large.txt	194433	6.0196788	19047	24487	13	0
27	250726	1000081		english_small.txt	84097	2.5642888	3606	4047	8	0
28	250726	1000081		french.txt	202358	6.6618799	21709	29956	22	0

Graph of time taken to produce the hash tables:



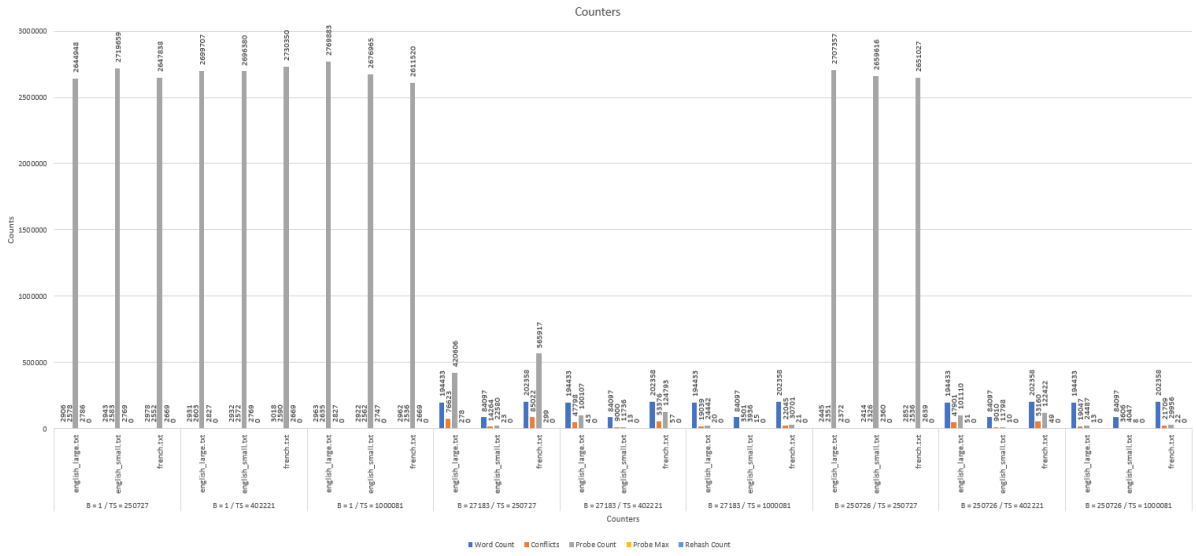
Analysis:

- Based on the above graph, all the hash tables produced with a hash base of 1 all exceeded the max time of 10 seconds to read in the files and fill the hash table with its data. This is because a hash base of 1 is small, thus it causes an uneven spread within the hash table because many of the hash values will be the same, causing many collisions to occur. This in turn causes the linear probe function to probe many spaces to find an empty space to store the value, which increases complexity, thus causing it to take a long time to fill the hash table.
- Besides that, it can also be seen that using a hash base value which is close to the size of the hash table, in this case when hash base = 250726 and table size = 250727 also causes the process of filling the hash table to perform poorly in my run as it exceeded the max time of 10 seconds for all the three files. This is because multiplying such a

hash base value will cause most of the values to be stored towards the end of the hash table, which also causes an uneven spread in the hash table, causing many collisions to occur as well. This also increases the complexity of the code thus causing it to take a long time to complete executing.

- Furthermore, it can also be noticed that the fastest time taken obtain to produce the hash table is when the hash base value is not similar to the value of table size, and the table size is very large. In our case, the fastest performance obtained is when the hash base is 27183 and the table size is 1000081. A hash base value further away from the table size value creates a more even spread in the hash table, and a larger table size also allows more spaces for keys to be entered into, which further reduces any collisions from occurring.

Graph of counters:



Analysis:

- Based on this graph and applying the analysis on the previous graph, the probe count is very high for the instances where the code exceeded the max time of 10 seconds. This is mainly because the spread of the data in the hash table when the hash base value is small or is close to the value table size is uneven, causing many conflicts to occur, and thus causing many linear probes to occur.
- The longest probe chain obtained from executing the code is 2827, and this occurred when the hash base is 1. Having long probe chains is bad, as having a longer probe chain increases complexity. This will cause the O(1) complexity promised by hash tables to be broken, as long probe chains takes more than O(1) time complexity to be executed.
- Lastly, the rehash count of all the runs is 0. This is because after executing each and every run, the hash tables still have empty spaces to accommodate more data, thus there is no need to rehash the table in these runs.