Hash Table Analysis

Our Prediction

The hash function is a crucial part of the entire hash table implementation. This function decides where new key-value pairs are placed within the hash table. The hash function is important as its efficiency decides the efficiency of the overall use of the hash table. Hash functions use a hash base to calculate the hash value, which is essentially the index where the key value pairs are to be inserted into the hash table. As a result, if the hash base is chosen poorly, it might not map different keys to unique hash values efficiently enough. As a result, this will cause many conflicts. This in turn causes the total probe length of the hash table to increase as every time a key value pair is inserted into the table, there is a high chance that the hash value it gets has been used by a previous entry to the hash table. On the other hand, if an appropriate hash base is used for the hash function, it is more likely that fewer conflicts will occur and less probing will be done in the process. However, this might be dependent on the hash table size.

A larger sized table will allow for a lesser number of conflicts or, in other cases, a lesser number of probes as there is plenty more space left for key value pairs to be inserted when a conflict arises. When the conflict arises, the next available space for the key value pair is close by, resulting in a smaller probe length throughout the hash table and smaller probe chains in total. This, however, might depend on a good hash base used for the hashing function. Despite having a large table size, a poor hash base and function will result in approximately the same number of probing and conflict resolutions. Having a poor hash base might undo the merits that a large table size brings to the table. These merits that come with using a large table size for small data storage will come at the expense of more memory storage. If the table size is about the size of the key value pairs needed, these merits perish.

Analysis using data and hash table statistics

To analyse the hash tables and validate or invalidate our predictions, we will use the fake data provided. We have three test data sets, each containing all the Indian cities, Australian cities, and US cities, respectively, in different text files. Different tables will be created using different hash bases, table sizes, and input data as per the table below.

Hash Base	Table Size	FileName
1	20021	indian_cities.txt (491 items)
9929	402221	aust_cities.txt (1035 items)
250726	1000081	us_cities.txt (19,084 items)

From the table above, 27 combinations can be made, and hence, for the purpose of analysing the hash table, we will create 27 hash tables that meet all the combinations to better understand the behaviour of the hash table with different attributes. The hash table used for this analysis will have a hashing function that uses a static base. The city names in the files will be used as key value pairs in the tables created.
(**Note:** For reference purposes, the three text files can be found in the submission folder along with the python file called analysis.py that has all the code related to producing the tables and its statistics)

The following table shows all the 27 tables created with their attributes and data used,

Table Number	Hash Base	Table Size	Data File Used
Table 1	1	20021	indian_cities.txt
Table 2	1	402221	indian_cities.txt
Table 3	1	1000081	indian_cities.txt
Table 4	9929	20021	indian_cities.txt
Table 5	9929	402221	indian_cities.txt
Table 6	9929	1000081	indian_cities.txt
Table 7	250726	20021	indian_cities.txt
Table 8	250726	402221	indian_cities.txt
Table 9	250726	1000081	indian_cities.txt
Table 10	1	20021	aust_cities.txt
Table 11	1	402221	aust_cities.txt
Table 12	1	1000081	aust_cities.txt
Table 13	9929	20021	aust_cities.txt
Table 14	9929	402221	aust_cities.txt
Table 15	9929	1000081	aust_cities.txt
Table 16	250726	20021	aust_cities.txt
Table 17	250726	402221	aust_cities.txt
Table 18	250726	1000081	aust_cities.txt
Table 19	1	20021	us_cities.txt
Table 20	1	402221	us_cities.txt
Table 21	1	1000081	us_cities.txt
Table 22	9929	20021	us_cities.txt
Table 23	9929	402221	us_cities.txt
Table 24	9929	1000081	us_cities.txt
Table 25	250726	20021	us_cities.txt
Table 26	250726	402221	us_cities.txt
Table 27	250726	1000081	us_cities.txt

The results of the statistics for the 27 tables are as follows in a tabular format:
(**Note:** Refer to the previous table to understand the attributes and data used for each table if the need arises.)

Table Number	Conflict Count	Distance Probed	Longest Probe	Rehash Count
Table 1	212	3249	67	0
Table 2	212	3249	67	0
Table 3	212	3249	67	0
Table 4	10	10	1	0
Table 5	0	0	0	0
Table 6	0	0	0	0
Table 7	3	3	1	0
Table 8	0	0	0	0
Table 9	0	0	0	0
Table 10	654	129359	888	0
Table 11	654	129359	888	0
Table 12	654	129359	888	0
Table 13	25	27	2	0
Table 14	0	0	0	0
Table 15	0	0	0	0
Table 16	31	31	1	0
Table 17	2	2	1	0
Table 18	1	1	1	0
Table 19	28252	213305667	18969	1
Table 20	18656	169529021	18969	0
Table 21	18656	169529021	18969	0
Table 22	7130	13824	25	1
Table 23	450	475	2	0
Table 24	160	160	1	0
Table 25	7073	13394	18	1
Table 26	482	504	2	0
Table 27	191	196	2	0

Now let us analyse the statistics of the 27 tables based on the attributes and data they hold. Looking at the first three tables (Table 1 to Table 3), it can be seen that all the statistics are exactly the same and are rather huge. All these three tables have the same hashing base but increasingly different table sizes. This tells us that despite the table sizes being different, all the statistics are poor and the same because of the choice of hash base, which is 1. Even though we are using Indian cities for this table, whose number of cities is much less than the available table sizes, there are a large number of conflicts and probing in the hash table. Comparing the first 3 tables to the next 6 tables, it can be inferred that the change in the hash base to another value which is more suitable yields better results, as can be seen from the table. By using a more appropriate hash value, the conflict count and distance probed drastically changed. From 212 conflicts to 0 conflicts and a total distance probed from 3249 to 0 just by changing the hash base from 1 to 9929. The same goes for changing it to a hash base of 250726. All of this insight is regarding the first 9 tables that use the Indian cities as the data.

If we refer to Table 4 and Table 7, we understand that despite having the same table size, a different hash base can make a significant difference. As is observed for these 2 tables, the total conflicts found were reduced from 10 to 3 just by changing the hash value from 9929 to 250726. For the next steps in table sizes, there is no difference as all the statistics are a 0, indicating an efficient and desirable case and also points to the fact that a larger table size could alleviate some of the conflicts.

The same trends can be observed in a similar fashion for tables that use the other two text files, which are the Australian cities and the US cities. However, the statistics for them are larger than the tables that use the Indian cities, as these two text files hold more city data, especially the US cities. It houses around 19,000 elements or cities, and as a result, for a particular table size attribute of less than 19,000 in size, a single rehash was done. Hence, the rehash count of 1 for Tables 19, 22, and 25. To avoid repetition, we will not go over these two files in detail like we did for the first text file. But the takeaway is that it follows very similar behaviour and nature.

Based on these findings, it can be concluded that our analysis strongly resonates with the prediction of the hash table results mentioned previously.