**Assignment-5 Hashing using Linear Probing with Dynamic Resizing**

Hashing Performed using three collision Probing techniques linear probing, quadratic probing, Double Hashing for different set of values such as 1000,5000,10000 and checked for the load factor constraint on the hash table.

Attached is a statistical report for the different collision probing technique with respect to different input values.

Below is the example of linear probing behavior of hash table with the different load factor values for the input = 5000

|  |  |
| --- | --- |
| **Row Labels** | **Sum of Linear** |
| 0.1 | 8050177 |
| 0.2 | 7861538 |
| 0.3 | 5074724 |
| 0.4 | 5270180 |
| 0.5 | 3082230 |
| 0.6 | 4203649 |
| 0.7 | 5661093 |
| 0.8 | 5834237 |
| 0.9 | 6020769 |
| **Grand Total** | **51058597** |

Based on the empirical analysis and statistical report the load factor value between 0.5 and 0.6 possesses least time of execution.

Initially the array was being resized for every load factor value.

Based on the empirical analysis and statistical report found out that the load factor of 0.5 or 0.6 should be the case when the hash table should be resized.

In the second part based on the statistical report 0.5 is load factor set when the hash table is resized. The values in the new hash table are stored incrementally based on the old key hash table and the remaining entry keys into the new hash table. To perform this task incrementally, made use of thread to perform the task of insertion of old hash values into the new one hash table in background.

Sample output

Linear Probing

Time taken for Storing Keys in Hash Table with load factor 0.5 is 2795739

Linear Collision Count 2

Hash Table

-1 -1 -1 33 -1 -1 -1 -1 8 -1 70 -1 -1 73 74 -1 -1 -1 -1 49 20 21 -1 -1 54 -1 -1 -1 -1 89

Also the search operation performed to search the key in the hash table is done using the hash search technique which uses the hash function to search for key in the hash table rather than performing sequential or serial search.