Performance Analysis of Hash Table

# Time Execution Analysis:

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **100**  (in microsecond) | **1000**  (in microsecond) | **10000**  (in microsecond) |
| **Insert in Chaining** | 116 | 1207 | 11478 |
| **Delete in Chaining** | 68 | 350 | 2383 |
| **Search in Chaining** | 27 | 282 | 3499 |
| **Insert in Linear Probing** | 340 | 1285 | 11043 |
| **Delete in Linear Probing** | 37 | 386 | 3090 |
| **Search in Linear Probing** | 41 | 445 | 4194 |
| **Insert in Quadratic Probing** | 496 | 1471 | 18587 |
| **Delete in**  **Quadratic Probing** | 1101 | 2024 | 5308 |
| **Search in Quadratic Probing** | 775 | 1607 | 4860 |

# Questions:

# 1:

The Performance Changes as the number of Nodes Changes. The execution time for 100 Nodes to add in Hash Table is 116 microseconds. But for 10000 Number of Nodes, it is 11478 microseconds. Similarly for deleting and searching, there is the difference in the performance

# 2:

The Reason behind this change is as the number of Nodes increases in Hash Table. To add the new item, it has to either traverse through all the nodes in the chaining to add at the end of the chain in Case of Chaining. And in case of Linear and Quadratic Probing it has to traverse and find the Empty Place to insert the incoming Node. That`s why there is change in Performance for Search, delete and Add Operations.

# 3:

The Hash Table with chaining when we have a small table and the Linear and quadratic probing is preferable when we have a very large table and We can easily perform Linear and Quadratic Probing.