**Time Complexity**

|  |  |  |
| --- | --- | --- |
|  | Ordered Map | Unorderd Map |
| Storing | O(logn) (best, avg, worst) | O(1) (best) O(n) (worst) |
| Fetching | O(logn) (best, avg, worst) | O(1) (best) O(n) (worst) |

n is number of elements **Hashing methods**

- division

- folding

- mid square

**Division Method**

(**step 1 Modulous** + **step 2 sorted linear chaining**) =(to generate an sorted linkedlist)

**Unordered Map**Time complexity

Best | Average case => O(1)

Worst case => O(n) (in case of internal collision)

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 28 | 98 | 68 |

**STEP 1: Modulous**

8%10 = 8

28%10 = 8

98%10 = 8

68%10 = 8

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 2: Sorted linear chaining**  
all values comes in index 8 = 8 > 28 > 68 > 98

**Collision Handling Methods**

- separate chaning   
- open addressing

**Separate Chaning**- Each has bucket (position in the table) contains a list of entris and when collision occurs, new entry are added to the list

**Open Addressing**- If a collison occur, the algorithm search for the next available sloth in the table