**Analysis of Double Hashing:**

The load factor α is always going to be ≤ 1.

We assume that every probe looks at a random location in the table. 1 – α fraction of the table is empty.

€ (Expected no. of probes required to find an empty location) => 1/ (1 – α)

Unsuccessful search

Where α = n / m ….n => no of elements

….m => size

Avg (no. of probes for successful search) = Avg (no. of probes required to insert all the elements.)

To insert an element we need to find an empty location.

|  |  |  |
| --- | --- | --- |
| inserting | Avg mo. Of probes | Total no of probes |
| First m/2 | <= 2 | m |
| Next m/4 | <= 4 | m |
| Next m/8 | <= 8 | m |

Probes required to insert (m/2 + m/4 + m/8+…..+m/2i) => probes required to leave 2i empty

elements

=> m \* i

Probes required to leave 1 – α table empty => -m log (1 – α)

Avg probes required to insert n elements => -(m/n) log (1 – α) = -(1 – α) log (1 – α)

**Comparison with chaining:**

|  |  |  |
| --- | --- | --- |
|  | **un**successful | successful |
| Chaining | O (1 – α) | O (1 – α) |
| Probing | O (1 / 1 – α) | O( ) |

**Memory Analysis:**

Most efficient O (n) as no extra memory is required.

We achieve this by using Open addressing technique and putting the keys in the same table rather than linked list.