Hash Tables

-Hash table *(CHAINING)*

- ***Space Complexity O(n).***

- **Average Time Complexity.**
  - **If the spread function is uniform for the data set used.**
  - Each key has the **same probability** of being assigned to any of the (m) indices.
  - The **average length of the lists** is (n/m) , equal to the load factor.
  - **Successful searches**: (1 + L/2) accesses on average. **O(1)**
    - In a list of length (L) the average is (L/2) accesses.
    - Deletion of a key is equivalent to a successful search.

  - **Failed searches**: (1 + L) accesses on average. **O(1)**
    (m) failed lookups, one for each table index, traverse the sum of lengths of lists: ((m + n)/m = 1 + L).
  - **Insertion**: 1 access best case, **O(1)** in amortized time.
  - The **restructuring** is **O(n),** but it guarantees **(n) insertions into O(1).**

- **Worst case Time Complexity.**
  - **The worst case occurs when the hash function is extremely non-uniform: Assigns the same position to all the keys in the data set.**
  - The table contains **a single list** with the **n elements**.
  - Under normal circumstances the **probability of falling in the worst case is negligible (1/m!)**
  - But given a hash function, it is always possible to design a data set that causes the worst case (attack by efficiency degradation).
    - **Search, delete**: **O(n)** accesses on average.
    - **Insertion**: Remains **O(1)** in amortized time.

-Hash table *(OPEN ADDRESSING)*

- ***Space Complexity O(n).***

- **Average Time Complexity.**
  - **If the spread function is uniform for the data set used.**
  - **Successful searches**: (ln(1/(1+L))/L) accesses on average. **O(1)**
    - Deletion of a key is equivalent to a successful search.
  - **Failed searches**: (1/(1+L))accesses on average. **O(1)**
  - **Insertion**: 1/(1+L), **O(1)** in amortized time.
  - The **restructuring** is **O(n),** but it guarantees **(n) insertions into O(1).**