HashTable:- A Hash Table is a data structure designed to be fast to work with.

The reason Hash Tables are sometimes preferred instead of arrays or linked lists is because searching for, adding, and deleting data can be done really quickly, even for large amounts of data.

Uses of Hash Tables

Hash Tables are great for:

- Checking if something is in a collection (like finding a book in a library).
- Storing unique items and quickly finding them (like storing phone numbers).
- Connecting values to keys (like linking names to phone numbers).

The most important reason why Hash Tables are great for these things is that Hash Tables are very fast compared Arrays and Linked Lists, especially for large sets. Arrays and Linked Lists have time complexity

O(n) for search and delete, while Hash Tables have just O(1) on average

Hash Tables Summarized

Hash Table elements are stored in storage containers called **buckets**.

Every Hash Table element has a part that is unique that is called the **key**.

A **hash function** takes the key of an element to generate a **hash code**.

The hash code says what bucket the element belongs to, so now we can go directly to that Hash Table element: to modify it, or to delete it, or just to check if it exists. Specific hash functions are explained in detail on the next two pages.

A **collision** happens when two Hash Table elements have the same hash code, because that means they belong to the same **bucket**. A collision can be solved in two ways.

Chaining is the way collisions are solved in this tutorial, by using arrays or linked lists to allow more than one element in the same bucket.

Open Addressing is another way to solve collisions. With open addressing, if we want to store an element but there is already an element in that bucket, the element is stored in the next available bucket. This can be done in many different ways, but we will not explain open addressing any further here.

Building A Hash Table from Scratch

To get the idea of what a Hash Table is, let's try to build one from scratch, to store unique first names inside it.

We will build the Hash Set in 5 steps:

1. Starting with an array.
2. Storing names using a hash function.
3. Looking up an element using a hash function.
4. Handling collisions.
5. The basic Hash Set code example and simulation.