**Hash Function in Data Structure:**

There are almost 150 Zettabytes of data getting generated each day, which is equivalent to 150 trillion Gigabytes of data. With such an enormous speed of data growth, there emerges a need to store this data in an effective and efficient manner. By an effective and efficient manner of storage, we mean a way that will provide us the flexibility of retrieving the data in a minimal amount of time, because the more time required for an operation will directly increase the cost associated with that particular operation. So, in order to reduce the cost of an operation and do that task in an efficient manner, we need to reduce the retrieval time of the data for that particular task. And the solution for reducing the retrieval time is the Hash function or hash table. The hash function is used to map or bind the data to a particular hash value and then that hash value will be used as an index or a key to store that value in the hash table. The main benefit of storing data in the hash tables is that the retrieval time of the data stored in the hash tables is of unit time. That means the data that is stored in the hash table can be read with time complexity of O(1). So, in this way, the hash tables play an important role in drastically reducing the time required for reading the data from the hash tables. And for the working of the hash tables, it requires a hash function. Now let us see what is a hash function and how it works.

Hash Function can be defined as an algorithm or a function that is used to map or convert data of bigger size or length to a fixed or small index or hash value. In other words, a hash function can be defined as an algorithm that will be used to convert the data of higher length or size to data that is within a fixed range or size.

The input parameter that is passed to a hash function is the input data that needs to map to some hash data. And the output or result provided by a hash function depicts the hash value or the hashes that are associated with that input parameter value. The hash functions are associated with the hash tables that will actually store the data in the memory and the hash function is used only to map those values to the hash tables. The hash value returned by the hash function for the data item passed as an input parameter is then further used as an index to map or store that input data into the hash table. Or, we can say that the hash value returned by the hash function for the data item passed as an input parameter is used as a key for storing that data which will help in the easy and efficient retrieval of the stored data.

For an ideal hash function to work, it should satisfy two basic properties or conditions so that it can deliver optimal results within a specified span of computation period. These two basic properties or conditions for an efficient hash function to store data in the hash table are:

* Firstly, the hash function should be very fast in calculating and delivering the results. The speed is one of the main and crucial parameters that will affect the overall efficiency of a hash function. The relation between the speed of computation of the results and efficiency of a hash function is inversely proportional, which means more time required for computation and delivering the results lesser the efficiency of the hash function and vice-versa. So, ideally, it is required that the time span that is required for the calculations and delivering the results of a hash function should be as least as possible in order to maintain the efficiency of the hash function.
* Once the speed of creating the result is achieved, the next step is to deliver the results correctly and accurately. So, the second most important parameter is the accuracy of the results generated by the hash function. So, it is required that the result generated by the hash function should be unique and accurate because the hash values that are generated by the hash function acts as keys while sorting the data in the hash tables. So, the uniqueness of the hash value generated ensures that no two data should be mapped to the same key or hash value. That is why the accuracy and uniqueness of the result generated affects the efficiency of the whole hash function which in return affects the efficiency of storing data in the hash tables.

These are two main conditions that need to be satisfied by a hash function while generating the output data or hash values in order to maintain the efficiency of a hash function.

In conjunction with hash tables, hash functions are used to store and retrieve data items or data records. The hash function translates each datum or record associated with a key into a hash number which is used to index the hash table. When an item is to be added to the table, the hash code may index an empty slot (which is also called a bucket), in which case the item is added to the table there. The way input data is mapped to the indexes of the hash table using the hash function results in the creation of different types of hashing. In this article, we are going to see two main types of hashing that have their own benefits and drawbacks. The two main types of hashing types that we are going to understand are the chained hashing method and the open address hashing method.

In chained hashing, each slot that is present in the hash table acts as a head node for the input element that has got that index as a hash value output of the hash function. So, if that index's head node is empty then data is added at that head node otherwise if some data is already present at that index's head node, then the new incoming data is appended or added after that head node. In short, we can say that the indexes of the hash tables act as the head nodes of the linked list.

For instance, if we have ten indexes in a hash table starting from 0(zero) and ending at 9(nine). Then we have ten individual separate linked lists and the head nodes of all these ten different linked lists are stored at the indexes of this hash table. Then a hash function is used to map or store values in these different linked lists.