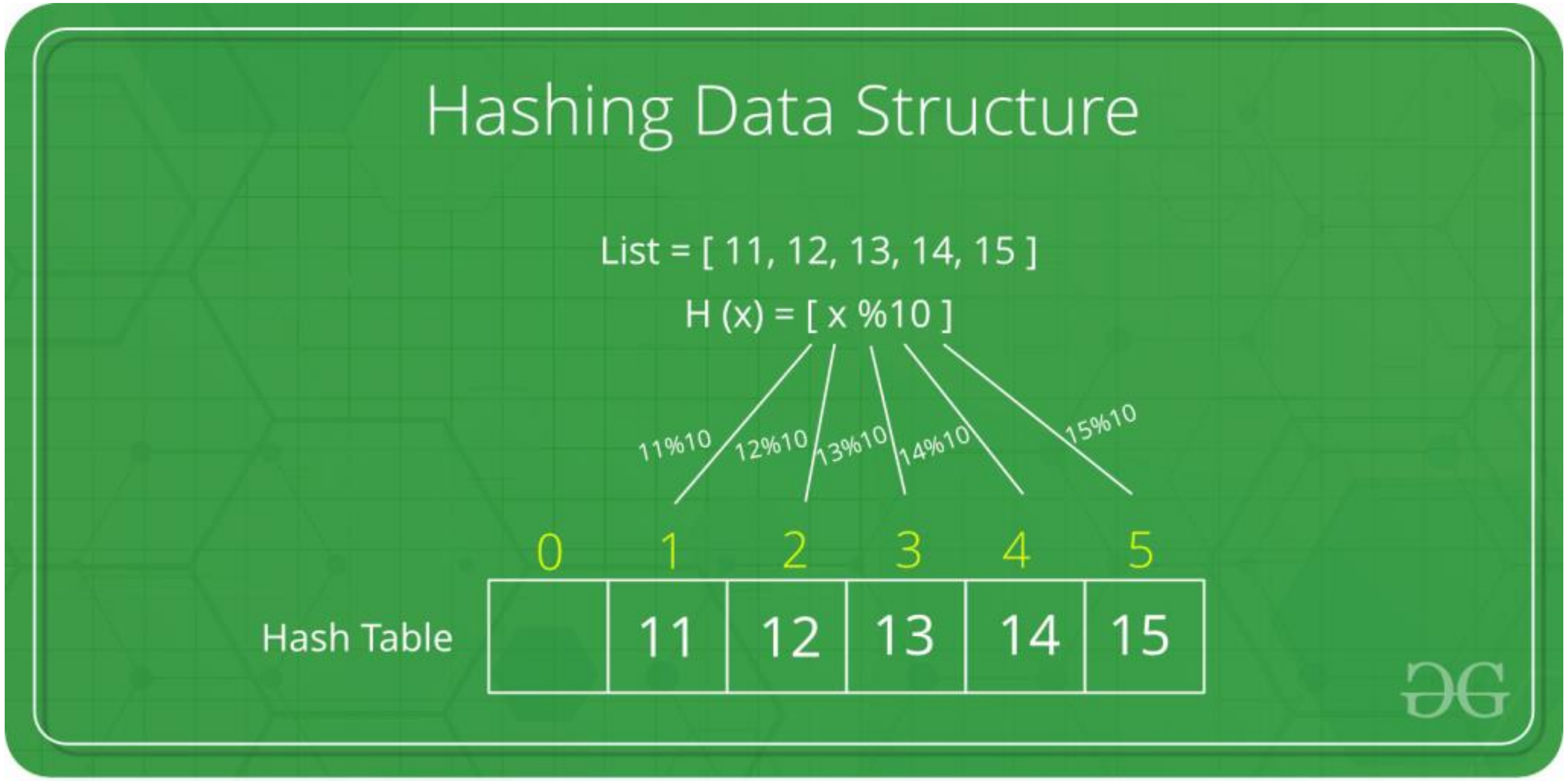
# Hashing Data Structure

## Hashing

- Hashing is a technique or process of mapping keys, values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.

Let a hash function H(x) maps the value at the index **x%10** in an Array.
For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.


Hashing Data Structure
List = [ 11, 12, 13, 14, 15 ]
H (x) = [ x %10 ]
11%10
12%10
13%10
14%10
15%10
0
1
2
3
4
5
Hash Table
11
12
13
14
15

# Key Pair

- Given an array **Arr** of **N** positive integers and another number **X**. Determine whether or not there exist two elements in **Arr** whose sum is exactly **X**.

**Example 1:**
**Input:** N = 6, X = 16 Arr[] = {1, 4, 45, 6, 10, 8}
**Output:** Yes
**Explanation:** Arr[3] + Arr[4] = 6 + 10 = 16

**Example 2:**
**Input:** N = 5, X = 10 Arr[] = {1, 2, 4, 3, 6} **Output:** Yes

- Hashing
- Recursion and Backtracking
- DnC -> quick sort, merge sort, binary search
- Greedy
- Tree
- Graph
- Dynamic Program

# Length of the longest subarray with zero Sum

**Problem Statement:** Given an array containing both positive and negative integers, we have to find the length of the longest subarray with the sum of all elements equal to zero.

**Example 1:**
**`Input Format`**`: N = 6, array[] = {9, -3, 3, -1, 6, -5}`
**`Result`**`: 5`

**Example 2:**
**`Input Format`**`: N = 8, array[] = {6, -2, 2, -8, 1, 7, 4, -10}`
**`Result`**`: 8 Subarrays with sum 0 : {-2, 2}, {-8, 1, 7}, {-2, 2, -8, 1, 7}, {6, -2, 2, -8, 1, 7, 4, -10} Length of longest subarray = 8`

**Example 3:**
**`Input Format`**`: N = 3, array[] = {1, 0, -5}` **`Result`**`: 1 Subarray : {0} Length of longest subarray = 1`

## Two Sum : Check if a pair with given sum exists in Array

**Problem Statement:** Given an array of integers nums[] and an integer target, return *indices of the two numbers such that their sum is equal to the target*.
**Note**: Assume that there is ***exactly* one solution**, and you are not allowed to use the *same* element twice. Example: If target is equal to 6 and num[1] = 3, then nums[1] + nums[1] = target is not a solution.
**Example** 1:
**`Input:`** `nums = [2,7,11,15], target = 9` **`Output:`** `[0,1]` **`Explanation:`** `Because nums[0] + nums[1] == 9, which is the required target, we return indexes [0,1]. (0-based indexing)`
**Example 2**:
**`Input Format:`** `nums = [3,2,4,6], target = 6` **`Output:`** `[1,2]` **`Explanation:`** `Because nums[1] + nums[2] == 6, which is the required target, we return indexes [1,2].`

# 4 Sum | Find Quads that add up to a target value

**Problem Statement:** Given an array of N integers, your task is to find unique quads that add up to give a target value. In short, you need to return *an array of all the unique quadruplets* [arr[a], arr[b], arr[c], arr[d]] such that their sum is equal to a given *target*.

**Pre-req:** Binary Search and 2-sum problem

**Note**:

- 0 <= a, b, c, d < n
- a, b, c, and d are distinct.
- arr[a] + arr[b] + arr[c] + arr[d] == target

**Example 1**:

**Input Format:** arr[] = [1,0,-1,0,-2,2], target = 0 **Result:** [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]] **Explanation:** We have to find unique quadruplets from the array such that the sum of those elements is equal to the target sum given that is 0. The result obtained is such that the sum of the quadruplets yields 0.

**Example 2**:

**Input Format:** arr[] = [4,3,3,4,4,2,1,2,1,1], target = 9 **Result:** [[1,1,3,4],[1,2,2,4],[1,2,3,3]]

## Longest Consecutive Sequence in an Array

**Problem Statement:** You are given an array of ‘N’ integers. You need to find the length of the longest sequence which contains the consecutive elements.

**Examples:**

```
Example 1: Input: [100, 200, 1, 3, 2, 4] Output: 4
Explanation: The longest consecutive subsequence is 1, 2, 3, and 4.

Input: [3, 8, 5, 7, 6] Output: 4
Explanation: The longest consecutive subsequence is 5, 6, 7, and 8.
```

## Length of Longest Substring without any Repeating Character

**Problem Statement:** Given a String, find the length of longest substring without any repeating character.

**Examples:**

**Example 1:**
**Input:** s = ”abcabcbb”
**Output:** 3
**Explanation:** The answer is abc with length of 3.

**Example 2:**
**Input:** s = ”bbbbb”
**Output:** 1 **Explanation:** The answer is b with length of 1 units.