Data Structure

Trainer : Nisha Dingare

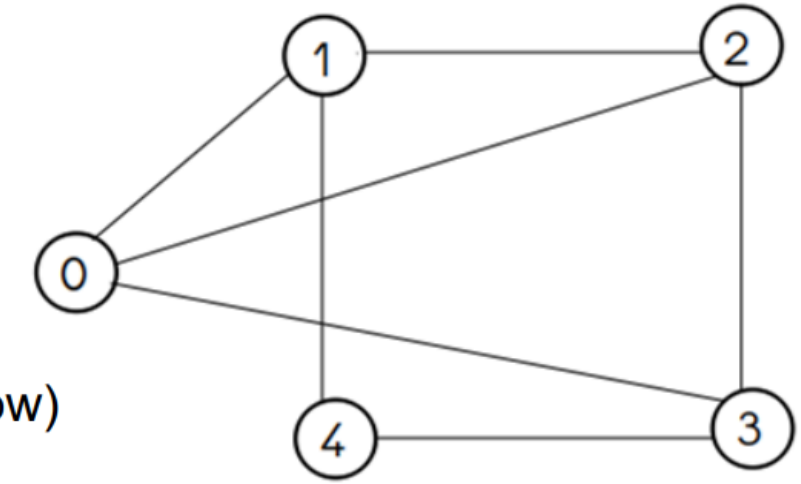
Email : nisha.dingare@sunbeaminfo.com

Graph Data Structure :

Graph: Graph is **non-linear** data structure, defined as set of vertices and edges.

- Vertices (or nodes) holds the data.
- Edges (or arcs) represent relation between vertices.
  - Edges may have direction and/or value assigned to them called as weight or cost.
- Applications of graph
  - Electronic circuits
  - Social media
  - Communication network
  - Road network
  - Flight/Train/Bus services
  - Bio-logical & Chemical experiments
  - Deep learning (Neural network, Tensor flow)
  - Graph databases (Neo4j)

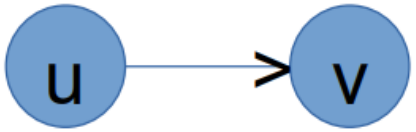
G(V,E): V={0,1,2,3,4}; E={ (0,1),(0,2),(0,3),(1,2),(1,4),(2,3),(3,4) }



Graph :

- If there exists a direct edge between two vertices then those vertices are referred as an **adjacent vertices** otherwise **non-adjacent**.

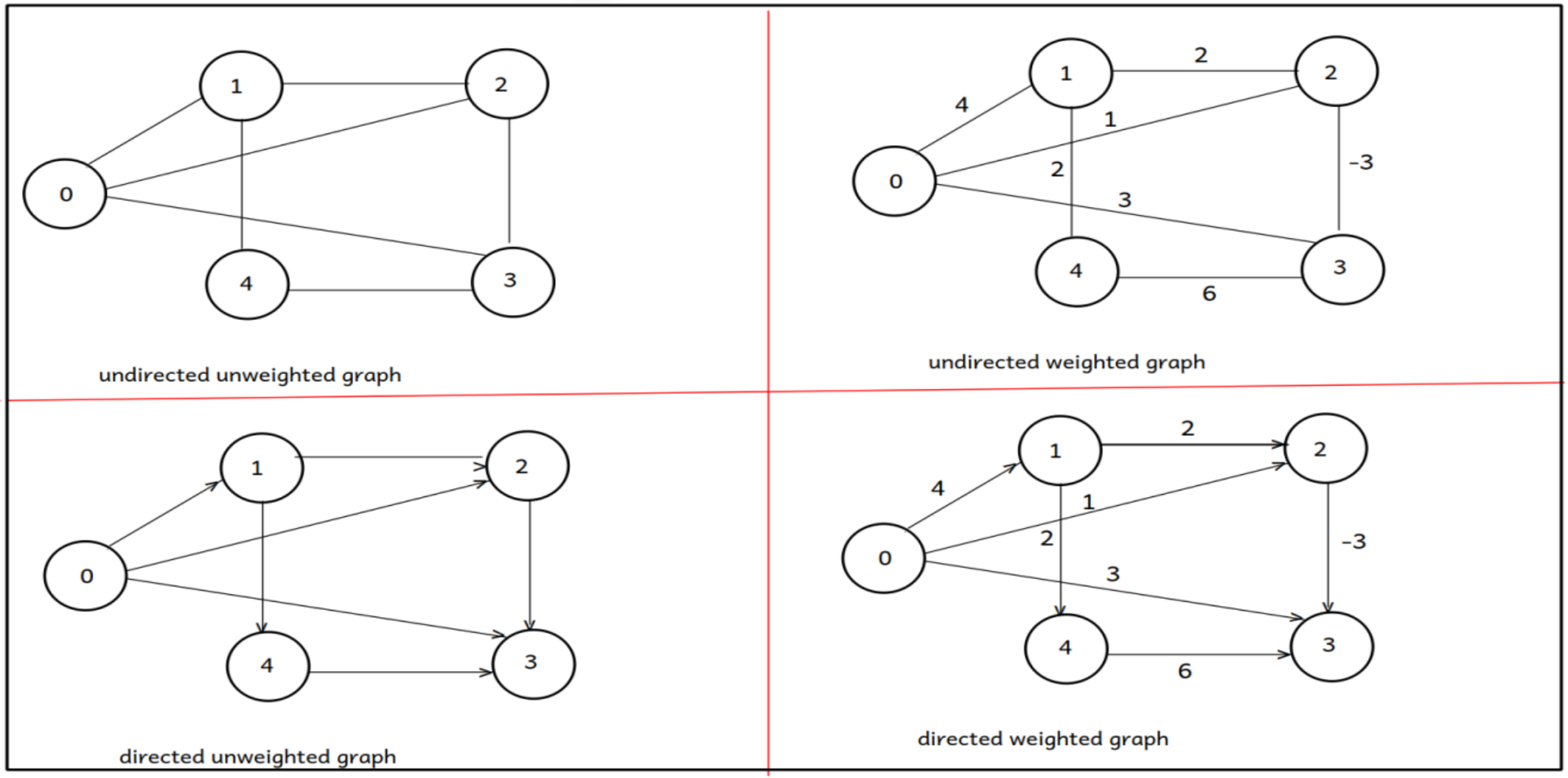
(u, v) == (v, u)



(u, v) != (v, u)

- If we can represent any edge either (u,v) OR (v,u) then it is referred as **unordered pair of vertices i.e. undirected edge.**
- **e.g. (u,v) == (v,u) => unordered pair of vertices => undirected edge => graph which contanis undirected edges referred as undirected graph.**
- If we cannot represent any edge either (u,v) OR (v,u) then it is referred as an **unordered pair of vertices** i.e. directed edge.
- **<u, v> != <v, u> => ordered pair of vertices => directed edge -> graph which contains set of directed edges referred as directed graph (di-graph).**

Graph :

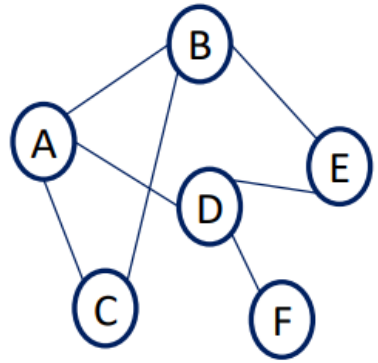

0
1
2
3
4
undirected unweighted graph
0
1
2
3
4
2
4
1
2
3
-3
6
undirected weighted graph
0
1
2
3
4
directed unweighted graph
0
1
2
3
4
2
4
1
2
3
-3
6
directed weighted graph

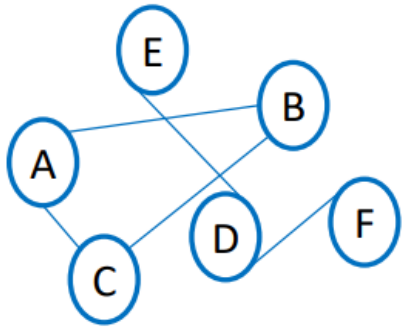
Graph :

- **Path:** Path is set of edges connecting two vertices.
- **Cycle:** in a given graph, if in any path starting vertex and end vertex are same, such a path is called as a cycle.
- **Loop:** if there is an edge from any vertex to that vertex itself, such edge is called as a loop. Loop is the smallest cycle.
- **Connected Vertices:** if there exists a direct/indirect path between two vertices then those two vertices are referred as a connected vertices otherwise not-connected.
  - Adjacent vertices are always connected but vice-versa is not true.

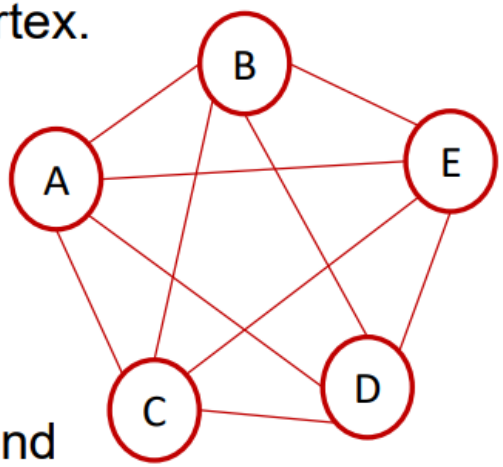
Graph :

- Connected graph
  - From each vertex some path exists for every other vertex.
  - Can traverse the entire graph starting from any vertex.

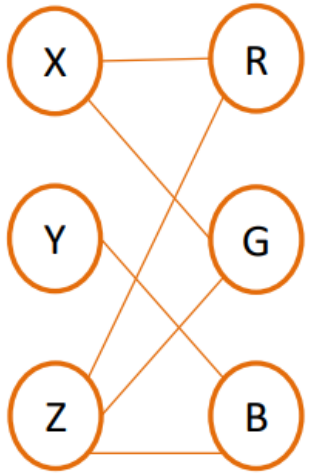




- Complete graph
  - Each vertex of a graph is adjacent to every other vertex.
  - Un-directed graph: Number of edges = n (n-1) / 2
  - Directed graph: Number of edges = n (n-1)

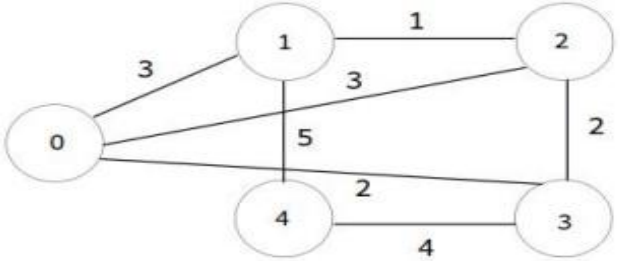


- Bi-partite graph
  - Vertices can be divided in two disjoint sets.
  - Vertices in first set are connected to vertices in second set.
  - Vertices in a set are not directly connected to each other.



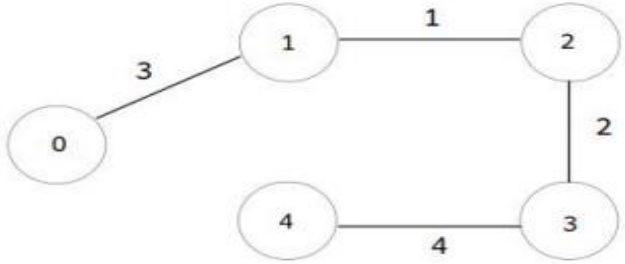
Spanning Tree :

Spanning Tree:



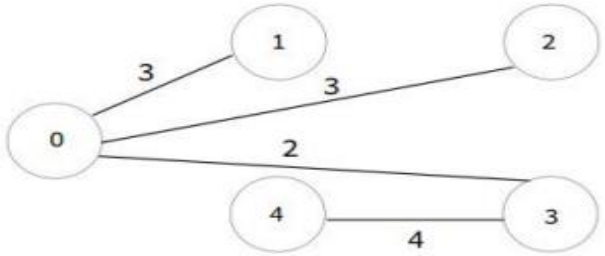
G1

Weight of a graph G1 = 20



G2

Weight of a graph G2 = 10



G3

Weight of a graph G2 = 12

Spanning Tree :

- **weight of a graph =** sum of weights of all its edges (applicable only in a weighted graph).

- **spanning tree :** it is a subgraph of a graph can be formed by removing one or more edges from it in such a way that it should remains connected and do not contains a cycle.

- **minimum spanning tree:** spanning tree of a given graph having min weight.

- one graph may has multiple spanning trees, **prim's** and **kruskal's** are 2 algorithms **to find minimum spanning tree of a given graph.**

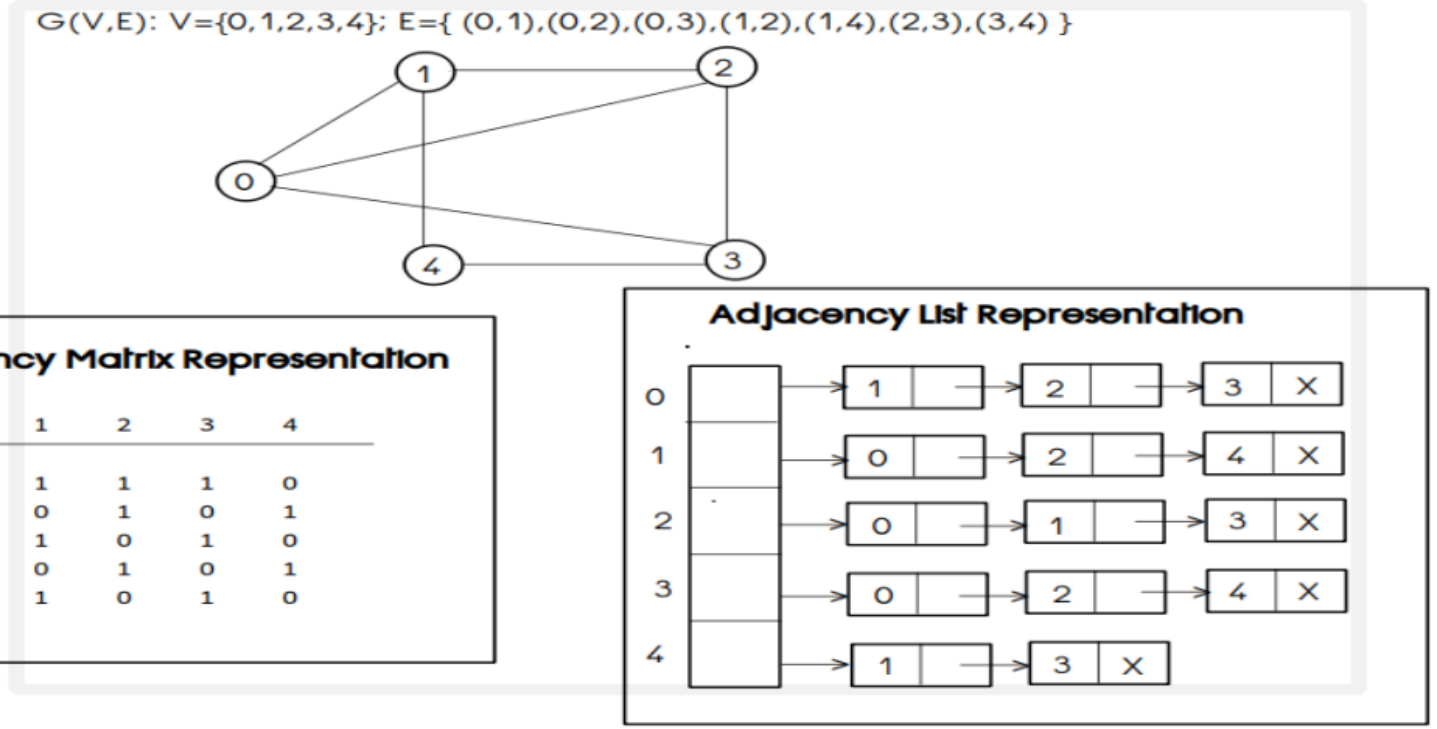
- **dijsktra's** and **bellman ford** are the algorithms to find **shortest distance of all vertices from given source vertex.**

Graph :

There are two graph representation methods:

1. Adjacency Matrix Representation (2-D Array)

2. Adjacency List Representation (Array of Linked Lists)



HashTable :

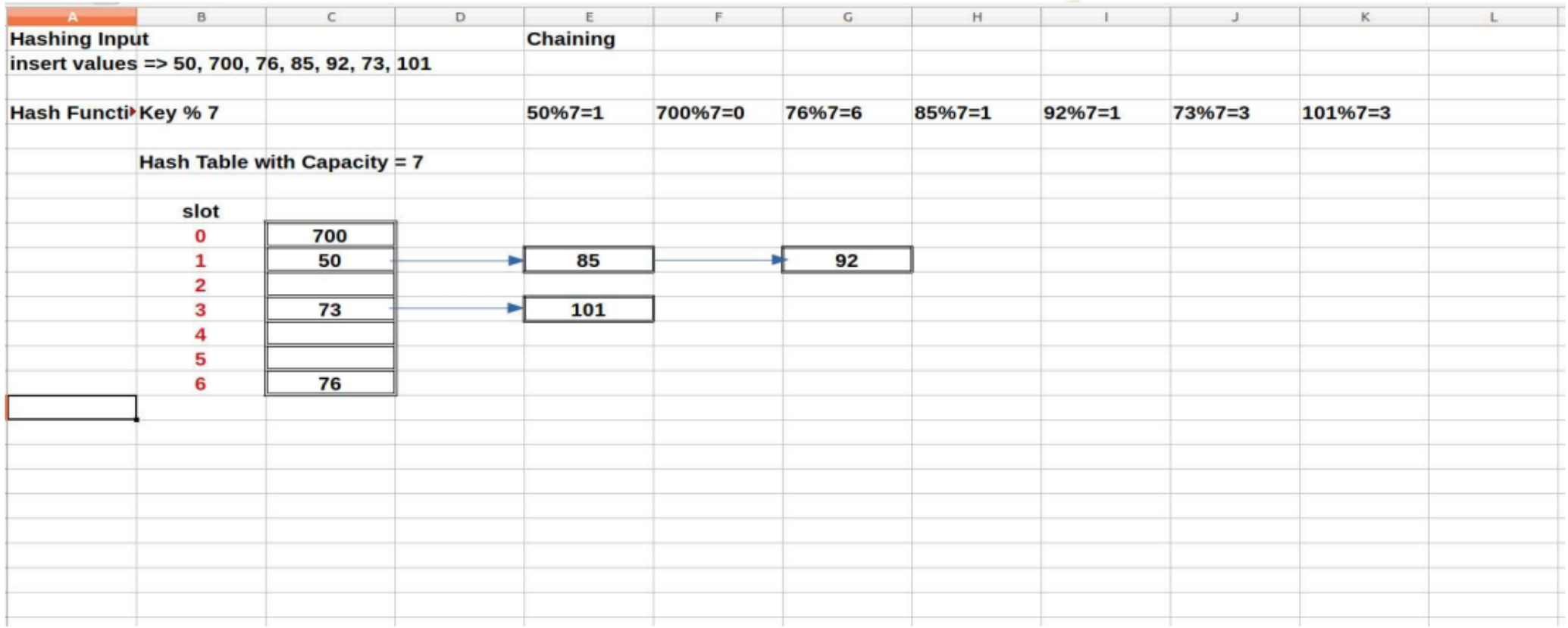
- **HashTable:** Hash table is an **associative** data structure in which data is stored in **key-value pairs** so that for the given key value can be searched in minimal possible time. Ideal time complexity is **O(1)**. Internally it is an array (table) where values can accessed by the index (slot) calculated from the key.
- **Hash Function:** It is mathematical function of the key that yields slot of the hash table where key-value is stored. Simplest example is: f(k) = k % size.
- **Collision:** There is possibility that two keys result in same slot. This is called collision and must be handled using some **collision handling technique.** It is handled by Open addressing or Chaining.

Hashing Input								
insert values => 50, 700, 76, 85, 92, 73, 101								
Hash Function	Key % 7		50%7=1	700%7=0	76%7=6	85%7=1	92%7=1	73%7=3	101%7=3
	Hash Table with Capacity = 7							
	slot							
	0	700						
	1	50	collision					
	2							
	3	73	collision					
	4							
	5							
	6	76						

Collision Handling Technique : chaining :

1. Chaining:

- The idea is to make each cell of hash table point to a linked list of records that have same hash function value. Chaining is simple, but requires additional memory outside the table.



Collision Handling Technique : Open Addressing :

2. Open Addressing:

- In open addressing, all elements are stored in the hash table itself. Each table entry contains either a record or NIL. When searching for an element, we one by one examine table slots until the desired element is found or it is clear that the element is not in the table.
- Open Addressing is done following ways:

A. Linear Probing: In linear probing, if collision occurs next free slot will be searched/probed linearly.

B. Quadratic Probing: In quadratic probing, if collision occurs next free slot will be searched/probed quadratically.

C. Double Hashing: In double hashing, if collision occurs next free slot will be searched/probed by using another hash function, so two hash functions can be use to find next/probe next free slot.

HashTable :

- **Load Factor = n / m**
  - n = Number of key-value pairs to be inserted in the hash table
  - m = Number of slots in the hash table
  - If n < m, then load factor < 1
  - If n = m, then load factor = 1
  - If n > m, then load factor > 1
- **Limitations of Open Addressing**
  - Open addressing requires more computation.
  - Cannot be used if load factor is greater than 1 (i.e. number of pairs are more than number of slots in the table).

Thank You