Hash tables

Introduction

A Hash table is defined as a data structure used to insert, look up, and remove key-value pairs quickly. It operates on the hashing concept, where each key is translated by a hash function into a distinct index in an array. The index functions as a storage location for the matching value. In simple words, it maps the keys with the value.

What is Load factor?

A hash table's load factor is determined by how many elements are kept there in relation to how big the table is. The table may be cluttered and have longer search times and collisions if the load factor is high. An ideal load factor can be maintained with the use of a good hash function and proper table resizing.

What is a Hash function?

A Function that translates keys to array indices is known as a hash function. The keys should be evenly distributed across the array via a decent hash function to reduce collisions and ensure quick lookup speeds.

Collision resolution techniques:

Collisions happen when two or more keys point to the same array index. Chaining, open addressing, and double hashing are a few techniques for resolving collisions.

- **Open addressing:** collisions are handled by looking for the following empty space in the table. If the first slot is already taken, the hash function is applied to the subsequent slots until one is left empty. There are various ways to use this approach, including double hashing, linear probing, and quadratic probing.
- **Robin Hood hashing:** To reduce the length of the chain, collisions in Robin Hood hashing are addressed by switching off keys. The algorithm compares the distance between the slot and the occupied slot of the two keys if a new key hashes to an already-occupied slot. The existing key gets swapped out with the new one if it is closer to its ideal slot. This brings the existing key closer to its ideal slot. This method has a tendency to cut down on collisions and average chain length.

Applications of Hash:

- Hash is used in disk-based data structures.
- In some programming languages like Python, JavaScript hash is used to implement objects.

- Hash tables are commonly used to implement caching systems
- Used in various cryptographic algorithms.
- Hash tables are used to implement various data structures.

Some advantages of Hash:

- Fast lookup
- Efficient insertion and deletion
- Space efficiency

Some disadvantages of Hash:

- Hash is inefficient when there are many collisions.
- Hash collisions are practically not be avoided for large set of possible keys.
- Hash does not allow null values.
- Hash tables have a limited capacity and will eventually fill up.

Graphs

Introduction

Graph Data Structure is a non-linear data structure consisting of vertices and edges. It is useful in fields such as social network analysis, recommendation systems, and computer networks.

Components of Graph Data Structure

- **Vertices:** Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.
- **Edges:** Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labelled/unlabelled.

Some types Of Graphs:

1. **Null Graph:** A graph is known as a null graph if there are no edges in the graph.
2. **Trivial Graph:** Graph having only a single vertex, it is also the smallest graph possible.

3. Undirected Graph: A graph in which edges do not have any direction. That is the nodes are unordered pairs in the definition of every edge.
4. Directed Graph: A graph in which edge has direction. That is the nodes are ordered pairs in the definition of every edge.

Representation of Graph Data Structure:

There are multiple ways to store a graph example:

- Adjacency Matrix
- Adjacency List

Advantages of Graph Data Structure:

- Graph Data Structure used to represent a wide range of relationships as we do not have any restrictions.
- They can be used to model and solve a wide range of problems.
- Graph Data Structure can be used to represent complex data structures in a simple and intuitive way, making them easier to understand and analyze.

Disadvantages of Graph Data Structure:

- Graph Data Structure can be complex and difficult to understand.
- Creating and manipulating graphs can be computationally expensive, especially for very large or complex graphs.

- Graph algorithms can be difficult to design and implement correctly, and can be prone to bugs and errors.
- Graph Data Structure can be difficult to visualize and analyze, especially for very large or complex graphs.