Name: Alaa Ibrahim Elshahawy
Track:Digital IC Design

Data Structures Report
(Hash table & Graphs)

==========

Introduction

hash tables and graphs and explore some graph algorithms. The hash table data structure uses hash functions to generate indexes where keys are stored or being accessed in an array. They find a wide use including implementation of associative arrays, database indexes, caches and object representation in programming languages.

A graph data structure is represented as a collection of finite sets of vertices or nodes (V) and edges (E). Edges are ordered pairs such as (u, v) where (u, v) indicates that there is an edge from vertex u to vertex v. Edges may contain cost, weight or length. Graphs are widely used in modeling networks including computer networks, social networks or some abstract phenomena of interrelated conceptual objects. Graph algorithms can be used to solve most problems occurring in situations modeled as networks.

1. Hash Table

To store a key/value pair we can use a simple array like data structure where keys directly can be used as index to store values. But in case when keys are large and can't be directly used as index to store value, we can use a technique of hashing. In hashing, large keys are converted into small ones, by using hash functions and then the values are stored in data structures called hash tables.

The goal of hashing is to distribute the entries (key / value pairs) across an array uniformly. Each element is assigned a key (converted one) and using that key we can access the element in O(1) time. Given a key, the algorithm (hash function) computes an index that suggests where the entry can be found or inserted.

Hashing is often implemented in two steps:

- The element is converted into an integer, using a hash function, and can be used as an index to store the original element, which falls into the hash table.

- The element is stored in the hash table, where it can be quickly retrieved using a hashed key. That is,

hash = hashfunc(key)

index = hash % array_size

In this method, the hash is independent of the array size, and it is then reduced to an index (a number between 0 and array_size − 1), using modulo operator (%).

A hash table data structure attracts a wide use including:

- Associative arrays - Hash tables are commonly used to implement many types of in-memory tables. They are used to implement associative arrays (arrays whose indices are arbitrary strings or other complicated objects).
- Database indexing - Hash tables may also be used as disk-based data structures and database indices (such as in dbm).
- Caches - Hash tables can be used to implement caches, auxiliary data tables that are used to speed up the access to data that is primarily stored in slower media.
- Object representation - Several dynamic languages, such as Perl, Python, JavaScript, and Ruby, use hash tables to implement objects.

Hash functions are used in various algorithms to make their computing faster.

2. Graphs

Consider a social network such as facebook which is an enormous network of persons including you, your friends, family, their friends and their friends, etc. This is referred to as a social graph. In this graph, every person is considered as a node of the graph and the edges are the links between two people. In facebook, friendship is a bidirectional relationship. That is if A is B's friend it implies also that B is A's friend, thus the associated graph is an undirected graph.

A graph is defined as a collection of finite sets of vertices or nodes (V) and edges (E). Edges are represented as ordered pairs (u, v), where (u, v) indicates that there is an edge from vertex u to vertex v. Edges may contain cost, weight or length. The degree of a vertex is the number of edges that connect to it. A path is a sequence of vertices in which every consecutive vertex pair is connected by an edge and no vertex is repeated in the sequence except possibly the start vertex. There are two types of graphs:

Undirected graph - a graph in which all the edges are bidirectional, essentially the edges don't point in a specific direction. Figure 1 depicts an undirected graph on four vertices and four edges.

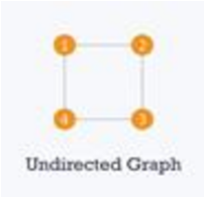

Fig.1

Directed graph (digraph) - a graph in which all the edges are unidirectional. Figure 2 depicts a digraph on four vertices and five directed edges (arcs).

Fig.2

A weighted graph has each edge assigned with a weight or cost. Consider a graph of 4 nodes depicted in Figure 4.2.3 . Observe that each edge has a weight/cost assigned to it. Suppose we need to traverse from vertex 1 to vertex 3. There are 3 possible paths: 1 -> 2 -> 3, 1 -> 3 and 1 -> 4 -> 3. The total cost of path 1 -> 2 -> 3 is (1 + 2) = 3 units, the total cost of path 1 -> 3 is 1 units and the total cost of path 1 -> 4 -> 3 is (3 + 2) = 5 units.

Fig.3

A graph is called cyclic if there is a path in the graph which starts from a vertex and ends at the same vertex, and this path is called a cycle. An acyclic graph is a graph which has no cycle. A tree is an undirected graph in which any two vertices are connected by only one path. A tree is acyclic graph and has N - 1 edges where N is the number of vertices. Figure 4.2.4 shows a tree on 5 nodes.

Fig.4

Graph Implementation outlines:

- Represent using adjacency matrices (dense graphs) or adjacency lists (sparse graphs).
- Implement algorithms like Topological Sorting or Shortest Paths

2.1 Shortest Path Algorithms

These algorithms determine the minimum path cost between two vertices in a graph. Common algorithms include:

Dijkstra's Algorithm: Finds the shortest path from a source node to all other nodes in a graph with non-negative weights.

Bellman-Ford Algorithm: Handles graphs with negative weights, though slower than Dijkstra's.

Floyd-Warshall Algorithm: Computes shortest paths between all pairs of nodes (used in dense graphs).

A Algorithm*: Heuristic-based algorithm often used in pathfinding and graph traversal problems like games and maps.

2.2 Topological Sorting Algorithms

Topological sorting is used to linearize the order of vertices in a Directed Acyclic Graph (DAG), respecting edge directions. Key algorithms include:

Kahn's Algorithm: Iteratively removes nodes with no incoming edges, creating a linear order.
DFS-Based Algorithm: Uses recursion to order nodes based on dependencies.

Applications:
- Scheduling tasks (e.g., build systems, course prerequisites)
- Dependency resolution (e.g., package installations)
- Workflow organization

3. Comparison

Feature	Hash Tables	Graphs
Structure	Key-value pair storage	Nodes connected by edges
Access Efficiency	O(1)	O(V+E) traversal
Applications	Data indexing, lookups	Network analysis, dependency graphs
Challenges	Collision handling	Computational complexity in large graphs