Advanced Algorithms and Data Structures

Assignment 2

https://github.com/StephenMcGlynn0/AlgorithmsAndDataStructuresAssignment2

Stephen Mc Glynn   L00176430

Q1. Define the following terms and concepts.

1.1 Hash Function:

A hash function is a function that takes an input(like a string or integer) and maps it to fixed-length integer values.

1.2 Hash Table:

A hash table is a data structure that holds key-value pairs and uses a hash function to create an index where every key should be stored. This allows for very fast average-case performance for insertion, deletion, and search operations, normally O(1).

1.3 Collision in a hash table:

A collision happens when two different keys produce the same hash value and that causes them to map to the same index in the hash table. Since multiple keys cant have the same slot, collision resolution techniques such as open addressing or chaining are used to handle this situation.

1.4 Load Factor:

A hash table load factor is the ratio of stored elements (keys) to available slots. Higher load factors result in increased collisions and sluggish operations and eventually lead to resizing when the load factor crosses a certain threshold (usually 0.75).

1.5 Open Addressing:

Open addressing is collision resolution strategy where the algorithm seeks another empty slot in the hash table instead of external structures when collision occurs. Common methods of probing are linear probing, quadratic probing, and double hashing. All elements are stored directly within the table array.

Q2. Compare hashing with tree based structures.

2.1 Discuss how hash-based data structures (like hash tables) differ from tree-based data structures (like red-black trees or AVL trees):

Hash based data structures such as hash tables, store data using a hash function which maps the data to specific positions using a hash code computed from the key, there is no inherent order for items to be placed, the key must be unique.

Tree based structures like red black trees organise data in an ordered way, every node has a relationship with other nodes and rebalances itself as needed to facilitate being ordered. Allows data to be looked at in sorted form

Hash tables are better for looking up specific data, due to quick lookup times if you already know the key you are looking for, red black trees are good for looking at data when the order of the data matters. e.g. you have employees salaries stored in a tree, you can easily find lowest salary at the far left, and the highest at the far right.

2.2 What are the advantages and disadvantages of each in terms of time complexity for operations such as insert, delete, and search?

Hash tables usually perform well for insert, delete and search operations, the usual would be O(1), this makes them very effective when working with large amounts of data, and when fast access is required. But this performance relies on having a good hash function and keeping collisions low, in the worst case when many keys collide, performance can degrade to O(n). and they don't support ordered operations.

Tree based structures generally perform insert, search and delete operations at O(log n), slow compared to a hash table for the average case but the performance is consistent, trees keep their data sorted so that makes finding minimum and maximum values, searching ranges, and stuff like that very efficient. A downside is that tree operations have more overhead due to rebalancing when adding things.

Q3.

https://github.com/StephenMcGlynn0/AlgorithmsAndDataStructuresAssignment2/blob/main/HashTableDivideAndConquer.java

Q4.

https://github.com/StephenMcGlynn0/AlgorithmsAndDataStructuresAssignment2/blob/main/LRUCache.java

Q5.

To support concurrency the LRU cache must be made thread safe so multiple threads can call get and put at the same time without destroying the caches internal state. Since the cache updates both Hashmap and the doubly linked list at the same time, race conditions must not occur during these operations.

One approach would be locking with a synchronized block, to prevent more than one thread changing the cache at a time, for example the get and put methods may be synchronized so that they only happen as one single uninterrupted step. Synchronised keyword ensures only one thread can run the method at a time

The HashMap could also be replaced with ConcurrentHashMap for safe concurrent lookups, but because the linked list still needs to be updated when items are accessed or evicted, locking is still needed on the list operations.