MIVA
OPEN UNIVERSITY

ALGORITHMS AND COMPLEXITY ANALYSIS

Search Algorithms

02

School of Computing

Fundamental Search Algorithms

Search Algorithms

Search algorithms are vital techniques in computing for efficiently finding specific elements in data structures like arrays, linked lists, or hash tables.

Linear Search

Linear Search is a basic search algorithm that checks each element sequentially until the target is found or the list ends, with a time complexity of O(n) in the worst case.

Binary Search

Binary Search operates on sorted data by dividing the search space in half repeatedly, offering a significant improvement in efficiency, with a time complexity of O(log n) in the worst case.

Linear Search is suitable for small or unsorted datasets, while Binary Search is ideal for large sorted datasets due to its logarithmic efficiency.

A hash table stores data in key-value pairs and uses a hash function to compute an index, allowing for constant-time performance (O(1)) on average for search, insertion, and deletion operations.

The hash function maps keys to indices in a table, ensuring efficient searching, provided the function is deterministic, uniform, and efficient.

Hash Table Implementation and Performance

Collision handling techniques such as chaining, open addressing, and double hashing are used to resolve conflicts when two keys hash to the same index.

01

Chaining

Chaining involves storing colliding elements in linked lists, while open addressing probes for the next available slot, using methods like linear probing, quadratic probing, or double hashing.

02

Load Factor Impact

Hash table performance depends on the load factor, with low load factors leading to faster operations and high load factors leading to more collisions and slower performance.

03

Real-World Applications

Hash tables are widely used in databases, compilers, programming languages (for maps/dictionaries), and networking for tasks like routing and DNS resolution.

The efficiency of a hash table depends on maintaining an appropriate table size and performing rehashing when necessary, ensuring continued performance near O(1) time complexity.

Linear Search and Binary Search remain useful for small-scale problems or specific use cases, but Hash Tables offer more scalability and speed when handling large datasets with constant-time operations.

Reflection Questions and Practice

Reflection Questions:

How does the time complexity of linear search compare to that of binary search in large datasets?

What are the practical advantages of using hash tables over linear or binary searches in real-world applications like databases or networking?
How would you handle collisions in a hash table? Which collision handling technique would you prefer for a high-load system?
In what cases might you still prefer linear search despite its inefficiency with larger datasets?
When would you choose binary search over hash table search, considering factors like data size and sorting?

Call to Action:

> Try implementing both linear search and binary search algorithms on different data structures to compare their performance in terms of speed and efficiency. For practice, you can also implement a hash table with collision handling and test its performance with various datasets.

Questions:

What are the key differences in terms of performance between linear search and binary search?
How does the hash function ensure efficient searching in hash tables?
What are the advantages of open addressing over chaining in hash table collision handling?
How do you determine the load factor of a hash table, and why is it important for performance?
Can you think of a scenario where binary search would not be effective, even on sorted data?

Skills and Competencies

Algorithm Understanding
Understanding of search algorithms (linear, binary, and hash table searches).

Time Complexity Analysis
Time complexity analysis for linear search, binary search, and hash table operations.

Collision Handling
Collision handling techniques in hash tables (chaining, open addressing, double hashing).

Practical Application
Practical application of hash tables in real-world systems (databases, compilers, networking).

Optimisation Skills
Optimisation skills for large-scale data retrieval using hash tables.

Algorithm Implementation
Algorithm implementation in programming languages with an emphasis on efficiency.

Performance Evaluation
Evaluation of algorithm performance and scalability in varying datasets.

Problem-Solving
Problem-solving skills in choosing the right search algorithm based on data structure and use case.