Ashish Sahu
@ashsau

Top 6 Algorithm

Every Programmer Should Know

- bubble sort
- selection sort
- Insertion sort
- merge sort
- quick sort
- heap sort

Sorting Algorithms

- Linear search
- binary seach
- bfs
- dfs)

Searching Algorithm

Recursive Algorithm

Dijkstra's Algorithm

Dynamic Programming

Hashing Algorithms

NEXT ➡


Ashish Sahu
@ashsau

Sorting Algorithms

Sorting algorithms arrange data in a specific order, crucial for optimizing performance and efficiency.

Bubble Sort

- Concept: Repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order.
- Use Case: Simple datasets, educational purposes.

Selection Sort

- Concept: Divides the input list into two parts: a sorted and an unsorted region. Repeatedly selects the smallest (or largest) element from the unsorted part and moves it to the end of the sorted part.
- Use Case: Small datasets, where data movement cost is minimal.

Insertion Sort

- Concept: Builds the final sorted array one item at a time, with the assumption that the first item is already sorted.
- Use Case: Small or nearly sorted datasets.

NEXT


Ashish Sahu
@ashsau

Sorting Algorithms

Merge Sort

- Concept: Divides the unsorted list into n sublists until each contains one element, then merges sublists to produce new sorted sublists until only one remains.
- Use Case: Large datasets where stability is important.

Quick Sort

- Concept: Selects a 'pivot' element and partitions the other elements into two sub-arrays according to whether they are less than or greater than the pivot.
- Use Case: Efficient for large datasets.

Heap Sort

- Concept: Converts the list into a heap, repeatedly removes the largest element from the heap, and inserts it into the sorted array.
- Use Case: When memory usage is a concern.

NEXT


Ashish Sahu
@ashsau

Searching Algorithms

Linear Search

- Concept: Linear search is a simple search algorithm that sequentially checks each element in a list until the desired element is found or the end of the list is reached.
- Use Case: It is suitable for unsorted or small datasets where efficiency is not a concern. It is easy to implement and works well when the list is short or when the element being searched for is near the beginning.

Binary search

- Concept: Binary search is a divide-and-conquer search algorithm that works on sorted arrays. It repeatedly divides the search interval in half until the target value is found or the interval is empty.
- Use Case: It is highly efficient for searching in large sorted datasets, with a time complexity of O(log n). However, it requires the data to be sorted beforehand and is not suitable for unsorted arrays.

NEXT

Ashish Sahu
@ashsau

Searching Algorithms

Breadth-First Search (BFS)

- Concept: BFS is a graph traversal algorithm that explores all the neighboring nodes at the present depth level before moving on to nodes at the next depth level.
- Use Case: It is useful for finding the shortest path in an unweighted graph or for exploring nodes in a hierarchical structure level by level.

First Search (DFS)

- Concept: DFS is another graph traversal algorithm that explores as far as possible along each branch before backtracking.
- Use Case: It is suitable for tasks such as topological sorting, detecting cycles in graphs, or finding paths in mazes and puzzles.

NEXT


Ashish Sahu
@ashsau

Recursive Algorithm

Concept:

- A recursive algorithm calls itself with smaller instances of the same problem until it reaches a base case where the solution can be directly computed.
- Recursion involves breaking down a problem into smaller subproblems and solving each subproblem recursively.

Use Case:

- Recursive algorithms are suitable when a problem can be divided into smaller, similar subproblems.
- The solution to the larger problem depends on the solutions to the smaller subproblems.

Examples include:

- Recursive factorial computation (n! = n * (n-1) * ... * 1).
- Generating Fibonacci sequence (each number is the sum of the two preceding ones).
- Traversing hierarchical data structures like trees (e.g., tree traversal algorithms such as inorder, preorder, postorder).

NEXT


Ashish Sahu
@ashsau

Dijkstra's algorithm

Concept: Dijkstra's algorithm is a graph search algorithm.

- It finds the shortest path from a source vertex to all other vertices.
- Works on a weighted graph with non-negative edge weights.
- Uses a priority queue to expand vertices with the shortest known distance from the source.

Use Case:

- Commonly used in routing and navigation applications.
- Essential for finding the shortest path between nodes in networks (e.g., roads, distances).
- Ensures efficient route planning and optimization in transportation and logistics systems.

Examples include:

- Graph Representation: Cities A, B, C connected by roads with distances: A-B (5), A-C (3), B-C (2).
- Objective: Find the shortest path from city A to all other cities.
- Algorithm Steps: Start from A, update distances to B (5) and C (3), choose C next due to shorter distance, update distance to B (4) via C.
- Final Path: Shortest paths: A → C (3), A → B → C (4).

NEXT ➡

Ashish Sahu
@ashsau

Dynamic Programming

Concept: Dynamic programming (DP) is an optimization technique used to solve problems by breaking them down into overlapping subproblems and solving each subproblem only once, storing its solution to avoid redundant calculations. It typically applies when the problem exhibits optimal substructure and overlapping subproblems.

Use Case: DP is applied in problems where solutions to subproblems can be reused, improving efficiency. Examples include:

- Finding the longest common subsequence in strings
- Solving the knapsack problem
- Optimizing matrix chain multiplication

Examples:

- **Problem**: Given an array [10, 22, 9, 33, 21, 50, 41, 60], find the length of the longest increasing subsequence.
- **Algorithm Steps:** Start with each element having a minimum sequence length of 1. Compare each element with previous elements to determine the longest increasing subsequence length ending at that element.
- **Final Solution**: Longest increasing subsequence length is 5: [10, 22, 33, 50, 60]

NEXT


Ashish Sahu
@ashsau

Hashing Algorithms

Concept:

- Hashing algorithms convert data of any size into a fixed-size hash value.
- The hash value uniquely represents the original data, ensuring data integrity and efficient retrieval.

Use Case:

- Data Storage and Retrieval
- Data Integrity
- Security Applications
- File Integrity

Examples:

- **Problem**:
  - Securely store user passwords for a web application to prevent unauthorized access.
- **Algorithm Steps:**
  - Use bcrypt hashing algorithm with salt for password security.
  - Generate a unique salt for each user's password.
  - Hash the password combined with the salt using bcrypt.
  - Store the hashed password and the salt in the database.
- **Result**:
  - Passwords are securely hashed and stored, protecting user accounts from unauthorized access.