BOSSCODER ACADEMY

ALGORITHMS

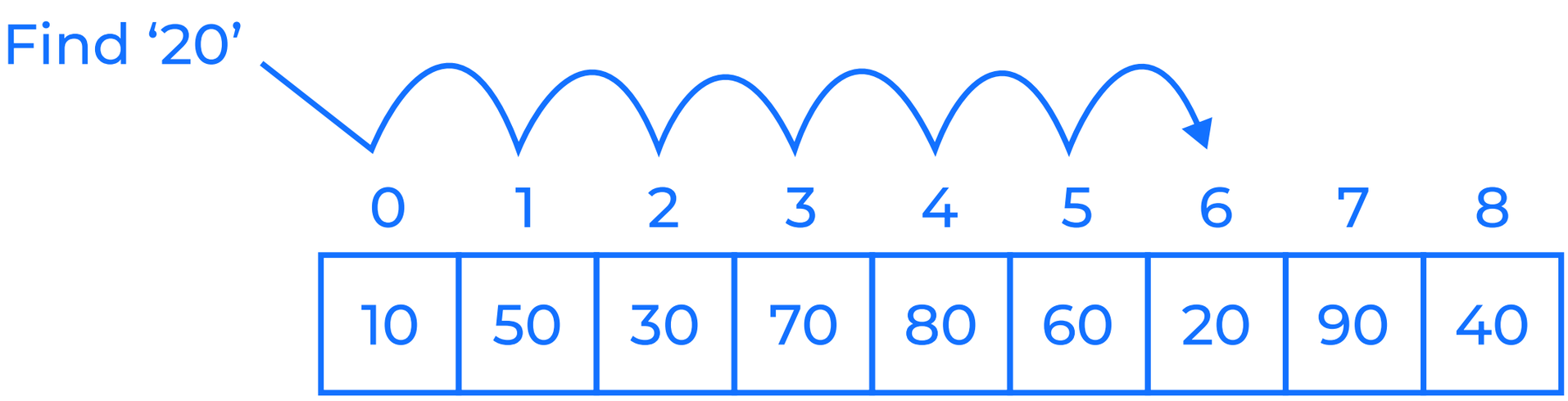
BASIC → ADVANCE

Unsorted

170	45	75	90	802	24	2	66

Sorted

2	24	45	66	75	90	170	802



Every Programmer Should Know

1.

Searching Algorithms

Linear Search

Search each and every Of the array till you find the required element

Time Complexity: O(n)

Binary Search

Searches for the element by comparing it with the middle item of the sorted array. If a match occurs, index is returned, else the searching area is reduced appropriately to either the upper half or lower half of the array

Time Complexity: $O(\log_2 n)$

2.

Sorting Algorithms

Bubble Sort

Works by swapping adjacent elements in repeated passes, if they are not in correct order. High time complexity and not suitable for large datasets

Time Complexity: O(n²))

Insertion Sort

The array is split into sorted and unsorted parts. Unsorted elements are picked and placed at their correct position in the sorted part

Time Complexity: O(n²))

Selection Sort

The smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array.

Time Complexity: O(n²))

Heap Sort

Uses the property of max and min heaps having largest and smallest elements at the root level It is an in place sorting algorithm.

Time Complexity: O(nlogn))

Merge Sort

Repeatedly divide the array into half, sort the halves and then combine them. It is a divide and conquer algorithm.

Time Complexity: O(nlog(n))

Quick Sort

A pivot element is picked and the partitions made around it are again recursively partitioned and sorted. It is a divide and conquer algorithm.

Time Complexity: O(nlog (n))
Worst Case Time Complexity: O(n2)

3.

Basic Math Algorithms

Euclid's Algorithm for GCD

Works by recursively dividing the bigger number with smaller number until the remainder is zero to get the greatest common divisor.

Sieve of Eratosthenes

Used for finding all prime numbers up to a given number by iteratively marking and removing the multiples of composite numbers.

Bit Manipulations

Perform operations at the bit-level or to manipulate bits in different ways by using bitwise operators AND, OR, NOT, XOR

4.

Breadth First Search and Depth First Search

Breadth First Search is implemented using a queue and starts at one given vertex and all its adjacent vertices are visited first before going to the next.

Depth First Search is implemented using a stack and starts at one given vertex and continues its search through adjacent vertices until there are none left.

Time complexity for both is O (V + E)

Dijkstra's Algorithm

Used to find the shortest path between two vertices in a graph. It is a greedy algorithm

5.

Tree Algorithms

Inorder Traversal

Traverse the left subtree, visit the root node and then the right subtree.

Preorder Traversal

Visit the root node, traverse the left subtree and then the right subtree.

Postorder Traversal

Traverse the left subtree, then the right subtree and then visit the root node.

Time Complexity: O(n)

Kruskal's Algorithm

Used for finding the minimum spanning tree, by sorting the edges in descending order and adding the smallest edge not added yet to form a tree with all the nodes.

6.

Dynamic Programming

Dynamic Programming works by storing the result of subproblems to access when needed without recalculation It uses memoization which is a top down approach and tabulation which is a bottom up approach.

Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm is dynamic programming based.

7.

Backtracking Algorithms

Solving problems by trying to build a solution one piece at a time, removing those solutions that fail to satisfy the constraints of the problem.

Standard questions for backtracking include. The N-queens problem, Sum of Subsets problem, Graph Colouring and Hamiltonian cycles.

8.

Huffman Coding Compression Algorithm

It is a technique of compressing data to reduce its size without losing any of the details. Generally useful to compress data with frequently occurring characters.

Involves two major parts

- Building a Huffman tree
- Traversing the tree and assigning codes to characters based on their frequency of occurrence.

Follow for more useful content

Telegram @codehype

Join Telegram for FREE Coding Resources

CodeHype