Belajaritma

Algoritma dan Pemrograman

Sesi VIII: Implementasi Algoritma Sederhana (PDF Example)

Definitions and Components of Algorithms

An algorithm is defined as a set of instructions that, when followed, lead to a specific outcome.

Some essential components and concepts of an algorithm include:

- Input: The data provided to the algorithm to solve a problem.
- Output: The result obtained after executing the algorithm.
- Step-by-step procedure: A sequence of operations performed on the input data based on the algorithm's design.
- Control structure: Logical constructs used to manipulate the flow of execution in the algorithm, such as conditional statements, loops, and branching statements.
- Data structures: Structures used to store and organize data, such as arrays, linked lists, stacks, and queues.

Sorting Algorithms

Sorting algorithm arranges data in a specific order, either ascending or descending, based on a particular key.

These are some of the examples of sorting algorithms:

- Bubble Sort

  First Pass:

  (5 1 4 2 8) –> (1 5 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1

  (1 5 4 2 8) –> (1 4 5 2 8), Swap since 5 > 4

  (1 4 5 2 8) –> (1 4 2 5 8), Swap since 5 > 2

  (1 4 2 5 8) –> (1 4 2 5 8), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

  Second Pass:

  (1 4 2 5 8) –> (1 4 2 5 8)

  (1 4 2 5 8) –> (1 2 4 5 8), Swap since 4 > 2

  (1 2 4 5 8) –> (1 2 4 5 8)

  (1 2 4 5 8) –> (1 2 4 5 8)

  Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

  Third Pass:

  (1 2 4 5 8) –> (1 2 4 5 8)

  (1 2 4 5 8) –> (1 2 4 5 8)

  (1 2 4 5 8) –> (1 2 4 5 8)

  (1 2 4 5 8) –> (1 2 4 5 8)

  It is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. Time complexity : $O(n^2)$.

- Selection Sort,

array[] = 64 25 12 22 11

Find the minimum element in array [0 - 4], then place it at the beginning array [0]

11 25 12 22 64

Find the minimum element in array [1-4], then place it at array [1]

11 12 25 22 64

Find the minimum element in array [2-4], then place it at array [2]

11 12 22 25 64

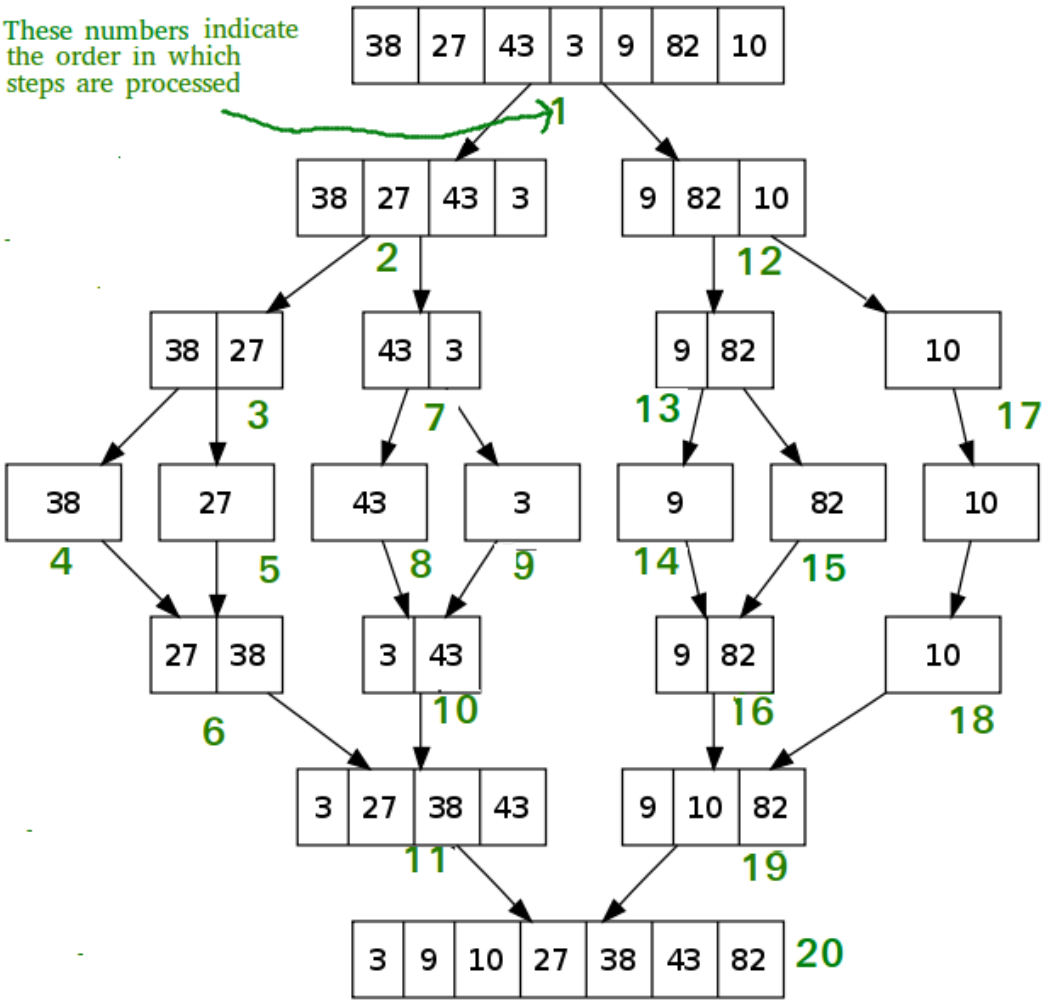
Find the minimum element in array [3-4], then place it at array [3]

11 12 22 25 64

Final sorted array = {11,12,22,25,64}.

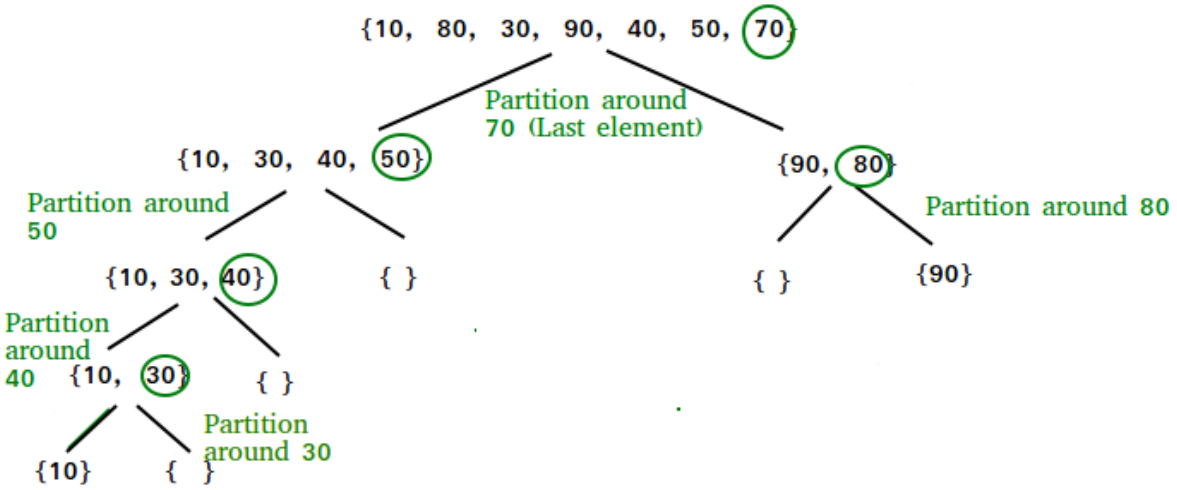
This sort works by repeatedly finding the minimum element (either ascending or descending) from the unsorted part and putting it at the correct location. Time complexity : O(n^2).

- Merge Sort



Merge sort works by recursively dividing the list into two halves, sorts each half, and then merges them back together in the correct order. Time complexity : O($n * log(n)$).

- Quick Sort

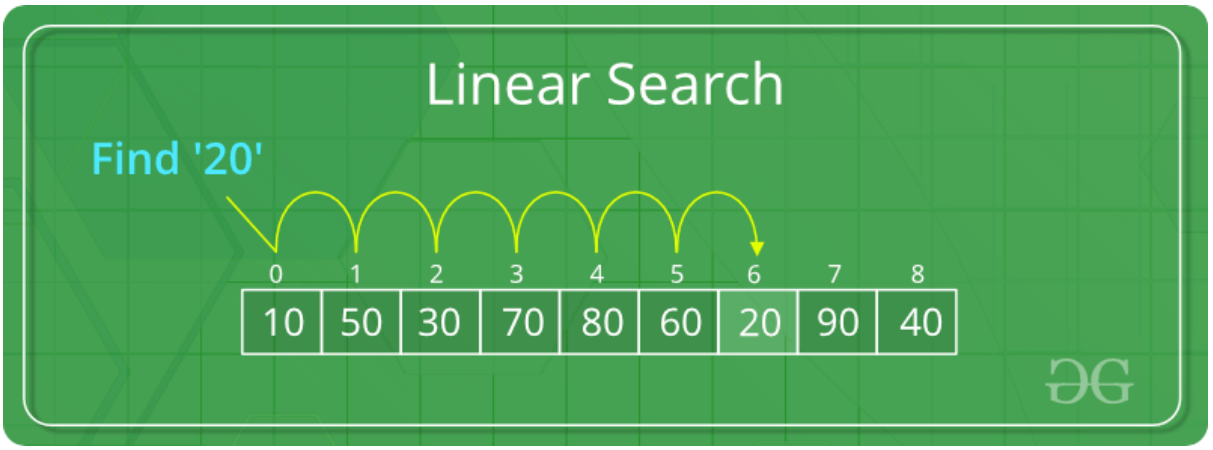


Quick sort works by picking an element as pivot and partitions the given list around the picked pivot to less and greater than. Time complexity : O($n * log(n)$).

Search Algorithms

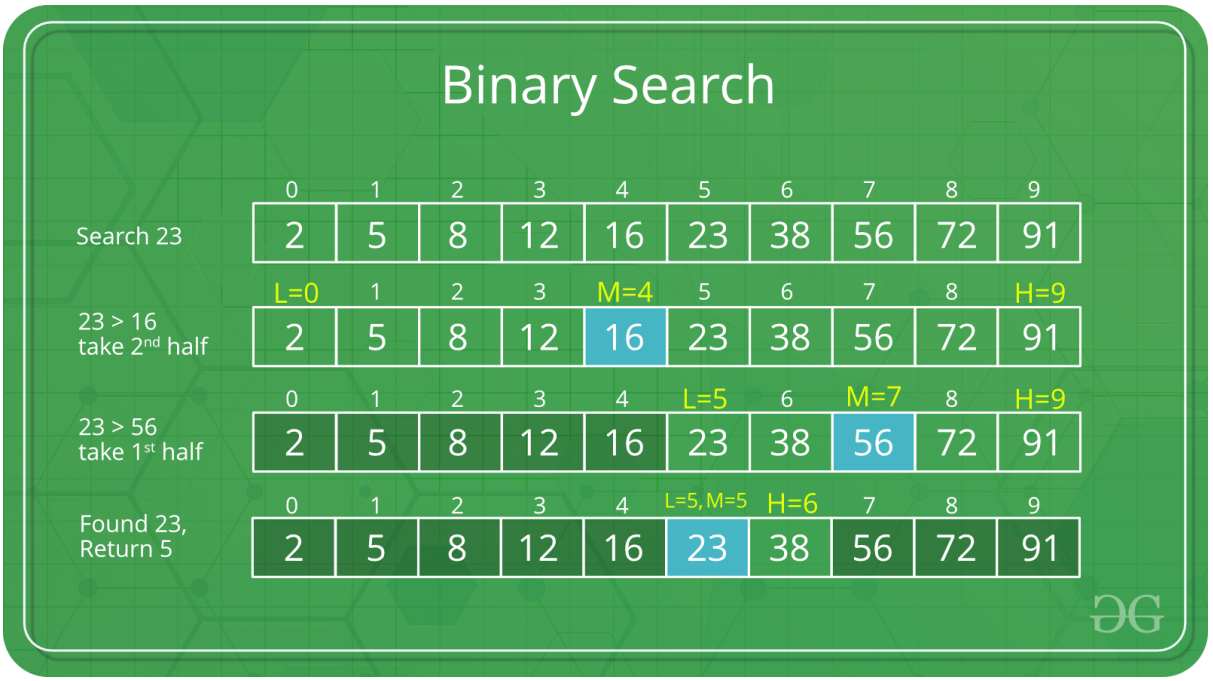
These are algorithms that are commonly used to search specific data in a database.

- Linear Search



A linear search works by searching the entire data list from the leftmost element. If it matches to the target value, it will return the index, thereby confirming that the data exists. Time complexity : O(n).

- Binary Search



First of all, the data list must be sorted first. After that, binary search works by dividing the list into two halves and searches each until either the target value is found or the entire list has been searched. Time complexity : O($log(n)$).