DATA STRUCTURES & ALGORITHMS

Topics to cover :

Easy
- Searching Algorithms
- Sorting Algorithms
- Arrays
- Strings
- Stacks
- Queues
- HashMaps

Medium
- Linked Lists
- Trees
- Heaps
- Recursion
- Graphs
- Greedy Algorithms
- Dynamic Programming
- Back Tracking

SEARCHING ALGORITHMS

1. Linear Search
2. Binary Search
3. Ternary Search
4. Jump Search
5. Interpolation Search
6. Exponential Search
7. Fibonacci Search
8. Sublist Search
9. Depth First Search
10. Breadth First Search
11. Binary Search on Answer
12. Trie Search (Prefix Tree)

SEARCHING ALGORITHMS

Searching Algorithms are techniques used to locate a specific element or group of elements within a datastructure.

1. LINEAR SEARCH

- Linear Search is also known as sequential search.
- Simple algorithm to find a specific element in an array or a list.
- It checks element by element until it finds the target element or reaches the end of the list.

Working :-

1. Start at the Beginning :-
   - Algorithm begins with first element in the list.
2. Compare the Element :-
   - It compares the current element with the target element.
3. Check for a Match :-
   - If the current element matches the target :-

     The algorithm stops, returns the index (position) of the target element in the list.

- If there is no match :-

The algorithm moves to the next element in the list and repeats the comparison.

4. Continue Until Found or End :-

The process continues until either :

- The target element is found, or
- The end of the list is reached (meaning the element is not on the list).

5. Result :-

If the target element is found, the algorithm returns its position; if not, it typically returns -1 (indicating that the element was not found).

Example :-

Let's say we have a list of numbers : [5, 3, 8, 4, 2]

And we want to find the number 4.

1. Start with the first element (5):
   - Compare 5 with 4 → Not a match.
2. Next element (3):
   - Compare 3 with 4 → Not a match.
3. Next element (8):
   - Compare 8 with 4 → Not a Match.

4. Next element (4) :

- Compare 4 with 4 ⟶ Match found!

Result :- The index of 4 is 3.

Time Complexity :-

- Best Case : O(1) - The target is found at the first position.
- Worst Case : O(n) - The target element is not in the list or is at the last position. Here n is the no. of elements in the list.
- Average Case : O(n) - On average, it will take about half the elements to find the target.

Space Complexity :-

- O(1) - It requires a constant amount of space, regardless of input size.

Java Code :-

```
public class LinearSearch {
    public static int linearSearch(int arr[], int target){
        for (int i=0; i< arr.length; i++) {
            if (arr[i] == target){
                return i; // Return the index if found.
            }
        }
        return -1 // Return -1 if not found.
    }
    public static void main (String[] args) {
        int[] numbers = {5, 3, 8, 4, 2};
        int target = 4;
        int result = linearSearch(numbers, target);
        if (result != -1) {
            System.out.println("Element found at index:"
            + result);
        }
        else {
            System.out.println("Element not found.");
        }
    }
}
```

Real-World Problem Using Linear Search

Scenario :- Finding a Book in an Unsorted Stack.

Imagine you work in a small library where books are stacked randomly on a shelf. Each book is labeled with a unique number (let's say the ISBN number). Your task is to find a specific book by its ISBN number.

Since the books are not arranged in any order, you cannot use a more efficient search method like binary search. The only way to find the book is to check each book one by one until you find the one with the matching ISBN number.

Steps :-

1. You start from the first book in the stack.
2. Check the ISBN number of that book.
3. If it matches the ISBN of the book you're looking for, you stop and pick the book.
4. If it doesn't match, you move to the next book and repeat the process until you find the book or reach the end of the stack.

IMPLEMENTATION IN JAVA.

```
Public class Library {
    public static int findBook (int[] Books, int targetISBN){
        for (int i=0; i< Books.length ; i++){
            if (books [i] == targetISBN) {
                return i;   // Book found at index
            }
        }
        return -1;
    }
    public static void main (String[] args) {
        int [] bookstack = {301, 245, 124, 789, 432, 567};
        int targetISBN = 432;
        int result = findBook (BookStack, targetISBN);
        if (result != -1) {
            System.out.println (" Book found at index: "+result);
        }
        else {
            System.out.println (" Book not found.");
        }
    }
}
```