**Exercise 2: E-commerce Platform Search Function**

1. **Understand Asymptotic Notation:**

Big O expresses the upper bound (worst-case time complexity) of an algorithm. It helps evaluate how an algorithm's run-time or space requirement grows with input size (n).

| Algorithm | Best Case | Average Case | Worst Case |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n/2) => O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

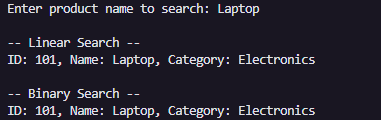
1. **Setup:**

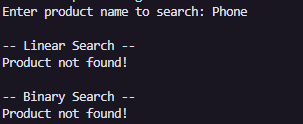


1. **Implementation:**



1. **Output:**





1. **Analysis:**

| Algorithm | Complexity | When to Use |
| --- | --- | --- |
| Linear Search | O(n) | Small or unsorted lists |
| Binary Search | O(log n) | Large, sorted product lists |

**Conclusion:**

* For e-commerce search, where the catalog is large and sorted, binary search is highly preferred.
* Linear search is okay for low-scale internal lookups or unsorted data.

**Exercise 7: Financial Forecasting**

1. **Understand Recursive Algorithms:**

* Recursion is when a function calls itself to solve smaller subproblems of the original problem.
* Recursion simplifies problems like:

Factorials,Fibonacci,Tree traversal and Financial growth over time

1. **Setup:**

Let’s assume we’re forecasting the future value of an investment with a fixed annual growth rate.

Formula:

FV(n)=PV\*(1+r)^n

Where:

FV(n) = future value after n years

PV = present value

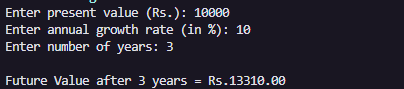
r = annual growth rate

n = number of years

1. **Implementation:**



**OUTPUT:**



1. **Analysis:**

| Step | Cost |
| --- | --- |
| Recursive Calls | O(n) |
| Each call | Constant time |
| Total | O(n) |

**How to Optimize?**

**We can optimise the recursive solution to avoid excessive computation by using an ITERATIVE approach.**

**public static double futureValueIterative(double presentValue, double rate, int years) {**

**double result = presentValue;**

**for (int i = 1; i <= years; i++) {**

**result \*= (1 + rate);**

**}**

**return result;**

**}**

**The time complexity is still O(n) but avoids stack overhead.**