**Week 1 : Data Structures and Algorithms Solution**

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

**Asymptotic Notation:-**

Big O notation describes how the performance (runtime or space) of an algorithm **scales** with input size n. It helps compare algorithm efficiency **independently of hardware or implementation language**.

| **Notation** | **Meaning** | **Example** |
| --- | --- | --- |
| **O(1)** | **Constant time** | **Accessing an array element** |
| **O(n)** | **Linear time** | **Linear Search** |
| **O(log n)** | **Logarithmic time** | **Binary Search** |
| **O(n²)** | **Quadratic time** | **Bubble Sort** |

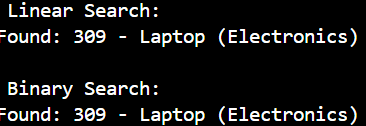
**Best, Average, and Worst Case (for Search)**

| **Scenario** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| **Best Case** | O(1) | O(1) |
| **Average** | O(n/2) ≈ O(n) | O(log n) |
| **Worst Case** | O(n) | O(log n) |

**Note:-**

**The implementation code is given in .java file**

**Output : -**

****

**Analysis :-**

**Time Complexity Comparison :-**

| **Search Type** | **Time Complexity** | **Requires Sorted Data?** |
| --- | --- | --- |
| Linear Search | O(n) | No |
| Binary Search | O(log n) | Yes |

**Which is Better?**

| **Situation** | **Recommended Search** |
| --- | --- |
| Small number of products | Linear Search |
| Large, sorted product catalog | Binary Search |
| Frequent insertions/deletions | Linear (unsorted) or indexed data structure |
| Performance-critical applications | Binary or use **search indexes** (e.g., DB or search engine like Elasticsearch) |