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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Steps:**

1. **Understand Asymptotic Notation:**
   * **Explain Big O notation and how it helps in analyzing algorithms.**

**Describe the best, average, and worst-case scenarios for search operations.**

Big O Notation

Big O notation is a mathematical concept used to describe the upper limit of an algorithm's running time or space requirements in terms of input size n. It helps in analyzing the efficiency of algorithms by providing an abstract measure of their performance as the input size grows. The notation focuses on the dominant term and ignores constants and lower-order terms.

* O(1): Constant time. The algorithm's performance is not affected by the size of the input.
* O(log n): Logarithmic time. The algorithm's performance grows logarithmically with the input size.
* O(n): Linear time. The algorithm's performance grows linearly with the input size.
* O(n log n): Linearithmic time. The algorithm's performance grows linearly times logarithmically with the input size.
* O(n^2): Quadratic time. The algorithm's performance grows quadratically with the input size.
* O(2^n): Exponential time. The algorithm's performance grows exponentially with the input size.

Best, Average, and Worst-case Scenarios

* Best-case scenario: The condition where the algorithm performs the minimum number of steps. For example, in a search operation, finding the item in the first position.
* Average-case scenario: The expected performance considering all possible inputs. It is a probabilistic measure.
* Worst-case scenario: The condition where the algorithm performs the maximum number of steps. For example, in a search operation, not finding the item until the last position or not at all.

1. **Setup:**
   * **Create a class Product with attributes for searching, such as productId, productName, and category.**
2. **Implementation:**
   * **Implement linear search and binary search algorithms.**
   * **Store products in an array for linear search and a sorted array for binary search.**

**Linear Search**

Linear search involves checking each element in the list one by one until the desired element is found or the end of the list is reached.

public class LinearSearch {

public static Product linearSearch(Product[] products, String targetName) {

for (Product product : products) {

if (product.getProductName().equals(targetName)) {

return product;

}

}

return null;

}

}

**Binary Search**

Binary search requires a sorted array and repeatedly divides the search interval in half. If the target value is equal to the middle element, the search is complete. If the target value is less than the middle element, the search continues in the left half, otherwise in the right half.

import java.util.Arrays;

public class BinarySearch {

public static Product binarySearch(Product[] products, String targetName) {

Arrays.sort(products, (a, b) -> a.getProductName().compareTo(b.getProductName()));

int left = 0;

int right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int comparison = products[mid].getProductName().compareTo(targetName);

if (comparison == 0) {

return products[mid];

} else if (comparison < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

}

1. **Analysis:**
   * **Compare the time complexity of linear and binary search algorithms.**

 Linear Search:

* Best Case: O(1)O(1)O(1) - The element is the first one.
* Average Case: O(n/2)≈O(n)O(n/2) \approx O(n)O(n/2)≈O(n) - The element is somewhere in the middle.
* Worst Case: O(n)O(n)O(n) - The element is the last one or not present.

 Binary Search:

* Best Case: O(1)O(1)O(1) - The element is the middle one.
* Average Case: O(logn)O(\log n)O(logn) - The search space is halved each time.
* Worst Case: O(logn)O(\log n)O(logn) - The element is not present and the search space is reduced to zero.
  + **Discuss which algorithm is more suitable for your platform and why.**

Suitability for E-commerce Platform

For an e-commerce platform where fast search performance is critical, binary search is more suitable due to its logarithmic time complexity, provided that the data is sorted. While linear search might be simpler to implement, its linear time complexity can become a performance bottleneck as the dataset grows.