**E-commerce Platform Search Function**

**Big O Notation**

Big O notation is a mathematical concept used in computer science to describe the performance or complexity of an algorithm. Specifically, it provides an upper bound on the time complexity or space complexity of an algorithm as a function of the size of the input. It helps in analyzing how the runtime or space requirements of an algorithm grow with the input size, allowing us to compare the efficiency of different algorithms.

* O(1): Constant time complexity. For example, accessing an element in an array by index.
* O(log n): Logarithmic time complexity. For example, binary search in a sorted array.
* O(n): Linear time complexity. For example, linear search in an unsorted array
* O(n log n): Linearithmic time complexity. For example, mergesort and heapsort.
* O(n²): Quadratic time complexity. For example, bubble sort or insertion sort.

**Time Complexity**

**Linear Search**

public int linearSearch(int productId){

for (int i = 0; i < products.length; i++) {

if(products[i].getProductId() == productId){

return i;

}

}

return -1;

}

* **Best-case:** O(1). The element being searched is the first element in the list.
* **Average-case:** O(n). On average, the search will need to check half of the elements in the list.
* **Worst-case:** O(n). The element is either not in the list or is the last element.

**Binary Search**

public int binarySearch(int productId){

int low = 0;

int high = products.length-1;

while(low<=high){

int mid = low+(high-low)/2;

if(products[mid].getProductId()==productId){

return mid;

}else if(products[mid].getProductId()>productId){

high = mid-1;

}else{

low = mid+1;

}

}

return -1;

}

* **Best-case:** O(1). The middle element of the array is the target element.
* **Average-case:** O(log n). The search repeatedly divides the array in half.
* **Worst-case:** O(log n). The search has to go through the maximum number of divisions to find the element or determine it is not in the array.

**Conclusion**

Thus from the above time complexity, we can conclude that binary search is more efficient than linear search.