**ECOMMERCE PLATFORM SEARCH FUNCTION**

**ASYMPTOTIC NOTATION:**

Big O Notation:

* It describes the upperbound of an algorithms running time based on input size n.
* It helps to analyse and compare the efficiency of algorithms

1. O(1)- constant time
2. O(n)- Linear time
3. O(log n)- Logarithmic time
4. O(n2)- Exponential time

**PROGRAM:**

import java.util.\*;

public class Product {

List<Integer> id = new ArrayList<>();

HashMap<Integer, List<String>> ecommerce = new HashMap<>();

public void addItems(int pid, String name, String category) {

List<String> products = new ArrayList<>();

products.add(name);

products.add(category);

ecommerce.put(pid, products);

if (!id.contains(pid)) {

id.add(pid);

Collections.sort(id);

}

}

public void linearSearch(int pid) {

System.out.println("LINEAR SEARCH");

int flag = 0;

for (int i = 0; i < id.size(); i++) {

if (id.get(i) == pid) {

flag = 1;

System.out.println("Product found at " + (i + 1) + " Position");

System.out.println("Product " + pid + ": " + ecommerce.get(pid));

break;

}

}

if (flag == 0) {

System.out.println("Product not in sale");

}

}

public void binarySearch(int pid) {

System.out.println("BINARY SEARCH");

int l = 0, h = id.size() - 1;

while (l <= h) {

int mid = (l + h) / 2;

if (id.get(mid) == pid) {

System.out.println("Product found at " + (mid + 1) + " Position");

System.out.println("Product " + pid + ": " + ecommerce.get(pid));

return;

} else if (id.get(mid) < pid) {

l = mid + 1;

} else {

h = mid - 1;

}

}

System.out.println("Product not in sale");

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Product p = new Product();

System.out.println("Enter '1' to add items, '2' for Linear Search, '3' for Binary Search, '4' to exit");

while (true) {

System.out.print("Enter choice: ");

int choice = sc.nextInt();

if (choice == 1) {

System.out.print("Enter Product ID, Name, Category: ");

int productId = sc.nextInt();

String productName = sc.next();

String category = sc.next();

p.addItems(productId, productName, category);

} else if (choice == 2) {

System.out.print("Enter product ID to search: ");

int key = sc.nextInt();

p.linearSearch(key);

} else if (choice == 3) {

System.out.print("Enter product ID to search: ");

int key = sc.nextInt();

p.binarySearch(key);

} else {

System.out.println("Exiting...");

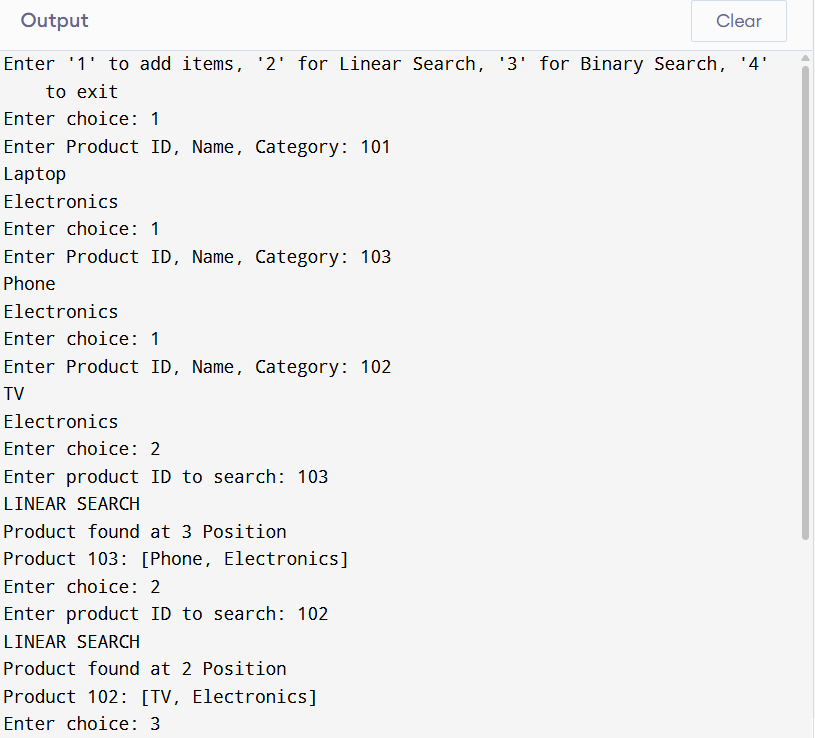
break;

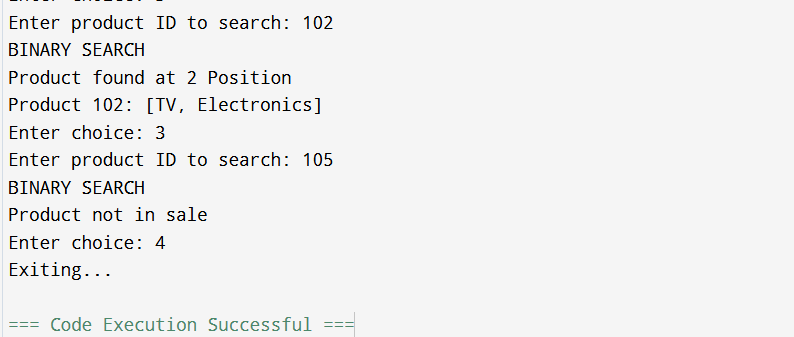
}

}

}

}





**BEST, WORST AND AVERAGE CASE IN SEARCH OPERATIONS:**

**FOR LINEAR SEARCH:**

1. BEST CASE- O(1)
2. AVERAGE CASE- O(n/2)->O(n)
3. WORST CASE- O(n)

**BINARY SEARCH:**

1. BEST CASE- O(1)
2. AVERAGE CASE- O(log n)
3. WORST CASE- O(log n)

**WHICH ALGORITHM IS SUITABLE:**

Binary search is suitable for searching because it only searches in one half which reduces the time complexity.