Step 1: Please use Loop Analysis method to analyze

public static int search(int arr[], int x)

Please explain your answer.

```
// Java code for linearly search x in arr[]. If x
// is present then return its location, otherwise
// return -1

class GFG
{
public static int search(int arr[], int x)
{
    int n = arr.length;
    for(int i = 0; i < n; i++)
    { //O(n)
        if(arr[i] == x)
            return i;
    }
    return -1;
}

public static void main(String args[])
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;

    int result = search(arr, x);
    if(result == -1)
        System.out.print("Element is not present in
array");
    else
        System.out.print("Element is present at index
" + result);
}
}
```

There are n elements being searched in an array, thus, Time Complexity = O(n)

Step 2: Optional homework 1268. Search Suggestions System - LC

```
class Solution:

    def suggestedProducts(self, products: List[str], searchWord: str) ->
List[List[str]]:

        products.sort()

        l = []

        n = len(searchWord)

        for i in range(1, n+1):
```

```
            l1 = []
            count = 0
            for p in products:
                if searchWord[:i] == p[:i] and count < 3:
                    l1.append(p)
                    count += 1
            l.append(l1)

        return l
```

Let the length of products be m and the length of searchWord be n. The time complexity of products.sort() is O(mlogm) and the time complexity of the loop is O(n*m). The time complexity of the function is O(mlogm + n*m).