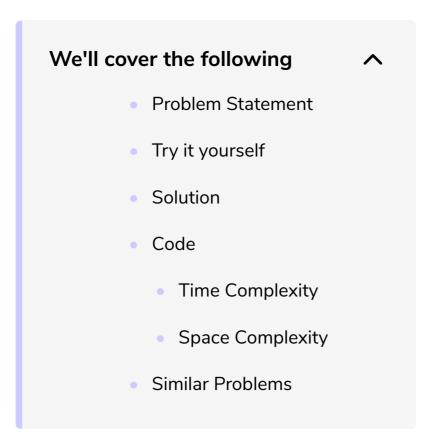


Fruits into Baskets (medium)



Problem Statement

Given an array of characters where each character represents a fruit tree, you are given **two baskets** and your goal is to put **maximum number of fruits in each basket**. The only restriction is that **each basket can have only one type of fruit**.

You can start with any tree, but once you have started you can't skip a tree. You will pick one fruit from each tree until you cannot, i.e., you will stop when you have to pick from a third fruit type.

Write a function to return the maximum number of fruits in both the baskets.

Example 1:

```
Input: Fruit=['A', 'B', 'C', 'A', 'C']
Output: 3
Explanation: We can put 2 'C' in one basket and one 'A' in the other from the subarray ['C', 'A', 'C']
```

Example 2:

```
Input: Fruit=['A', 'B', 'C', 'B', 'B', 'C']
Output: 5
Explanation: We can put 3 'B' in one basket and two 'C' in the other basket.
This can be done if we start with the second letter: ['B', 'C', 'B', 'B', 'C']
```

Try it yourself

Try solving this question here:

```
Java Python3 JS C++

import java.util.*;

class MaxFruitCountof2Types {
 public static int findLength(char[] arr) {
 // TODO: Write your code here
 return -1;
 }

8 }

TEST

SAVE RESET []
```

Solution

This problem follows the **Sliding Window** pattern and is quite similar to **Longest Substring with K Distinct Characters**. In this problem, we need to find the length of the longest subarray with no more than two distinct characters (or fruit types!). This transforms the current problem into **Longest Substring with K Distinct Characters** where K=2.

Code

Here is what our algorithm will look like, only the highlighted lines are different from Longest Substring with K Distinct Characters:

```
JS JS
           Python3
                        © C++
👙 Java
   import java.util.*;
                                                                                                 3 class MaxFruitCountOf2Types {
      public static int findLength(char[] arr) {
        int windowStart = 0, maxLength = 0;
        Map<Character, Integer> fruitFrequencyMap = new HashMap<>();
        for (int windowEnd = 0; windowEnd < arr.length; windowEnd++) {</pre>
         fruitFrequencyMap.put(arr[windowEnd], fruitFrequencyMap.getOrDefault(arr[windowEnd], 0) + 1);
          while (fruitFrequencyMap.size() > 2) {
           fruitFrequencyMap.put(arr[windowStart], fruitFrequencyMap.get(arr[windowStart]) - 1);
            if (fruitFrequencyMap.get(arr[windowStart]) == 0) {
             fruitFrequencyMap.remove(arr[windowStart]);
            windowStart++; // shrink the window
          maxLength = Math.max(maxLength, windowEnd - windowStart + 1);
        return maxLength;
      public static void main(String[] args) {
        System.out.println("Maximum number of fruits: " +
                             MaxFruitCountOf2Types.findLength(new char[] { 'A', 'B', 'C', 'A', 'C' }));
        System.out.println("Maximum number of fruits: " +
                             MaxFruitCountOf2Types.findLength(new char[] { 'A', 'B', 'C', 'B', 'B', 'C'
                                                                                 SAVE
                                                                                           RESET
                                                                                                    ×
                                                                                               3.333s
Output
 Maximum number of fruits: 3
 Maximum number of fruits: 5
```

Time Complexity

The time complexity of the above algorithm will be O(N) where 'N' is the number of characters in the input array. The outer <code>for</code> loop runs for all characters and the inner <code>while</code> loop processes each character only once, therefore the time complexity of the algorithm will be O(N+N) which is asymptotically equivalent to O(N).

Space Complexity

The algorithm runs in constant space O(1) as there can be a maximum of three types of fruits stored in the frequency map.

Similar Problems

Problem 1: Longest Substring with at most 2 distinct characters

Given a string, find the length of the longest substring in it with at most two distinct characters.

Solution: This problem is exactly similar to our parent problem.

