Fruits into Baskets (medium)

We'll cover the following

- Problem Statement
- Try it yourself
- Solution
- Code
    - Time Complexity
    - Space Complexity
- Similar Problems

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 you can’t skip a tree once you have started. 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 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 o
ther 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 o
ther 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++

```python
def fruits_into_baskets(fruits):
    start, max_fruits = 0, 0
    fmap = {}
    for winEnd in range(len(fruits)):
        fruit = fruits[winEnd]
        if fruit not in fmap:
            fmap[fruit] = 0
        fmap[fruit] += 1
        while len(fmap) > 2:
            start_fruit = fruits[start]
            fmap[start_fruit] -= 1
            if fmap[start_fruit] == 0:
                del fmap[start_fruit]
            start += 1
        max_fruits = max(max_fruits, winEnd-start+1)
    return max_fruits
```

Show Results Show Console

```
0.94s
```

2 of 2 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓	fruits_into_baskets([A, B, C, A, C])	3	3	Succeeded
✓	fruits_into_baskets([A, B, C, B, B, C])	5	5	Succeeded

Solution#

This problem follows the **Sliding Window** pattern and is quite similar to Longest Substring with K Distinct Characters (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5698217712812032/). 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 (https://www.educative.io/collection/page/5668639101419520/5671464854355968/5698217712812032/):

Java | Python3 | C++ | JS

```
  fruit_frequency = {}

  # try to extend the range [window_start, window_end]
  for window_end in range(len(fruits)):
    right_fruit = fruits[window_end]
    if right_fruit not in fruit_frequency:
      fruit_frequency[right_fruit] = 0
    fruit_frequency[right_fruit] += 1

    # shrink the sliding window, until we are left with '2' fruits in the
    while len(fruit_frequency) > 2:
      left_fruit = fruits[window_start]
      fruit_frequency[left_fruit] -= 1
      if fruit_frequency[left_fruit] == 0:
        del fruit_frequency[left_fruit]
      window_start += 1  # shrink the window
    max_length = max(max_length, window_end-window_start + 1)
  return max_length


def main():
  print("Maximum number of fruits: " + str(fruits_into_baskets(['A', 'B',
  print("Maximum number of fruits: " + str(fruits_into_baskets(['A', 'B',


main()
```

```
Output                                                     0.77s

 Maximum number of fruits: 3
 Maximum number of fruits: 5
```

Time Complexity#

The above algorithm's time complexity will be $O(N)$, where 'N' is the number of characters in the input array. The outer `for` loop runs for all characters, and the inner `while` 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.

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.

Back
Longest Substring with maximum K D...

Next
Longest Substring with Distinct Chara...

✓ Mark as Completed

Report an Issue