



# 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 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

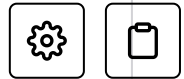
JS

C++

```
1 def fruits_into_baskets(fruits):
2     # TODO: Write your code here
3     d,start = {},0
4     ans = 0
5     for i in range(len(fruits)):
6         # extend windows
7         r_fruit = fruits[i]
8         if r_fruit not in d:
9             d[r_fruit] = 1
10        else:
11            d[r_fruit] += 1
12
13        # shrink
14        while len(d)>2:
15            l_fruit = fruits[start]
16            d[l_fruit] -= 1
17            if d[l_fruit]==0:
18                del d[l_fruit]
19            start +=1
20        ans = max(ans,i-start+1)
21    return ans
```



22

[Show Results](#)[Show Console](#) **2 of 2 Tests Passed**

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

0.15s

## 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/>)

968/5698217712812032/):



Java

Python3

C++

JS

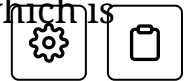
```
1 def fruits_into_baskets(fruits):
2     window_start = 0
3     max_length = 0
4     fruit_frequency = {}
5
6     # try to extend the range [window_start, window_end]
7     for window_end in range(len(fruits)):
8         right_fruit = fruits[window_end]
9         if right_fruit not in fruit_frequency:
10             fruit_frequency[right_fruit] = 0
11             fruit_frequency[right_fruit] += 1
12
13         # shrink the sliding window, until we are left with at most 2 unique fruits
14         while len(fruit_frequency) > 2:
15             left_fruit = fruits[window_start]
16             fruit_frequency[left_fruit] -= 1
17             if fruit_frequency[left_fruit] == 0:
18                 del fruit_frequency[left_fruit]
19             window_start += 1 # shrink the window
20         max_length = max(max_length, window_end - window_start + 1)
21     return max_length
22
23
24 def main():
25     print("Maximum number of fruits: " + str(fruits_into_baskets(['A', 'B', 'B', 'B', 'C', 'C', 'A', 'B'])))
26     print("Maximum number of fruits: " + str(fruits_into_baskets(['A', 'B', 'B', 'B', 'C', 'C', 'A', 'B'])))
27
28
```



## 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.

## 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](#)[Next →](#)[Longest Substring with K Distinct Cha...](#)[No-repeat Substring \(hard\)](#)

Completed



Report  
an Issue



Ask a Question

([https://discuss.educative.io/tag/fruits-into-baskets-medium\\_\\_pattern-sliding-window\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/fruits-into-baskets-medium__pattern-sliding-window__grokking-the-coding-interview-patterns-for-coding-questions))