904. Fruit Into Baskets

Similar Question/patterns:

1. what's the length of the longest contiguous substring that only contains 2 unique numbers
2. Find the longest continuous sub array that has exactly 2 distinct elements

Initial thoughts

1. Initially tried to solve this question using sliding protocol
2. Let's take an example of [1,2,3,2,2]
3. Here using my code..first will find insert fruits [1,2] and count is 2
4. Now as the next fruits is 3…now we adjust our pointers

```
elif fruits[j]-fruits[i] == 2:
    i = j
    j = j-1
    count = 0
```

5. See the code and get intuition of sliding window code

```
class Solution:
    def totalFruit(self, fruits: List[int]) -> int:
        i = 0
        j = 0
        count = 0
        max_c = 0
        while(j<len(fruits)):
            if fruits[j]-fruits[i] == 1 or fruits[j]-fruits[i] == 0:
                j = j+1
                count += 1
                if count > max_c:
                    max_c = count
            elif fruits[j]-fruits[i] == 2:
                i = j-1
                j = j-1
                count = 0
            else:
                i = j
                count = 0
        return max_c
```

6. But my solution is failing for testcase = [1,2,4,2,2]
7. Now referring sliding window soln on leetcode

Leetcode sliding window soln:

1. Here we initializing fruit_types with a counter
2. Lets take a test case [1,2,4,2,2]
3. We'll initialize 2 pointers left and right `left = right = 0`
4. If the fruit is new we'll update the counter and find the distinct fruits
5.
```
while right < len(fruits):
    # check if it is a new fruit, and update the counter
    if fruit_types[fruits[right]] == 0:
        distinct += 1
    fruit_types[fruits[right]] += 1

    # too many different fruits, so start shrinking window
    while distinct > 2:
        fruit_types[fruits[left]] -= 1
        if fruit_types[fruits[left]] == 0:
            distinct -= 1
        left += 1

    # set max_fruits to the max window size
    max_fruits = max(max_fruits, right-left+1)
    right += 1
```
6. Dry run
7. [1,2,4,2,2]
8. Right = 0 ⇒fruits[0]
9. If counter(right) == 0:
10. Distinct = 1
11. {1 : 1}
12. Max_fruits = max(0,0-0+1) ==1
13. Right += 1
14. Right = 1 ⇒fruits[1]
15. Distinct = 2
16. {1:1,2:1}

17. Max_fruits = max(1,1-0+1) == 2
18. Right += 1

Leetcode sol2 hashmap

1.
```python
class Solution:
    def totalFruit(self, fruits: List[int]) -> int:
        res,hashmap = 0,defaultdict(int)

        l = 0
        for r in range(len(fruits)):
            hashmap[fruits[r]] += 1

            if len(hashmap) > 2:
                hashmap[fruits[l]] -= 1
                if hashmap[fruits[l]] == 0:
                    del hashmap[fruits[l]]
                l += 1

            res = max(res, r-l+1)

        return res
```
2. Lets dry run this code
3. Test case = [1,1,1,2,4,2,2]
4. Res = 0, hashmap = {}
5. L = 0
6. For r in range(len(fruits)):
   {1:1}

   Res = max(0,0-0+1) = 1

7. R = 1 ⇒ fruits[1]
8. {2:1} hashmap = {1:1,2:1}
9. Res = max(1,1-0+1) = 2

10. R = 2 ⇒ fruits[2]

11. {4:1} hashmap = {1:1,2:1,4:1}
12. As len(hashmap)>2:
13. Hashmap[1] -= 1
14. If {1:0} then delete it
15. Hashmap = {2:1,4:1}
16. L = 1
17. Res = max(2,2-1+1) = 2

18. R = 3 ⇒fruits[3]
19. Hashmap = {2:2,4:1}
20. Res = max(2,3-1+1) = 3

21. R=4 ⇒fruits[4]
22. Hashmap = {2:3,4:1}
23. Res = max(3,4-1+1) = 4