Daily Coding Problem #298

Problem

This problem was asked by Google.

A girl is walking along an apple orchard with a bag in each hand. She likes to pick apples from each tree as she goes along, but is meticulous about not putting different kinds of apples in the same bag.

Given an input describing the types of apples she will pass on her path, in order, determine the length of the longest portion of her path that consists of just two types of apple trees.

For example, given the input `[2, 1, 2, 3, 3, 1, 3, 5]`, the longest portion will involve types `1` and `3`, with a length of four.

Solution

A naive solution would involve examining all sublists of our input, and taking the one of minimum length that only contains two distinct elements.

This solution would be $O(N^3)$, where N is the length of our input, since there will be $O(N^2)$ combinations of start and end ranges, and for each one we must examine the elements in between.

Fortunately, there is a linear-time solution we can implement instead.

At any given point when traversing the input, we must track of a few things:

- The two types of apples that are currently being collected, `a` and `b`.
- The number of times we have consecutively picked the most recent type of apple, `last_picked_count`.
- The current count of trees recently seen with apples `a` and `b`.

To see how these variables fit together, we can consider two cases: either the next tree we encounter is new, or it contains apples of one of the two types we are already collecting.

Suppose the former is true. Then going forward, the bags should hold apples of this new type, and apples that match the last one picked. Our current count will then be reset to `last_picked_count` plus one.

Now suppose the latter is true. In this case, we can increment the current count, and, if the apple is the same as the last one picked, increment `last_picked_count`.

In either case, if the current count is more than our running maximum, we should update that maximum.

One final detail to watch out for is that when we start iterating over this list, we must take care to set `last_picked_count` appropriately if the first two trees are the same.

```
def pick_fruit(trees):
    a, b = trees[0], trees[1]

    last_picked = b
    last_picked_count = (a == b)
    max_length_path = curr = 1

    for tree in trees[1:]:
        if tree not in (a, b):
            a, b = last_picked, tree
            last_picked = tree
            curr = last_picked_count + 1
        else:
            curr += 1
            if last_picked == tree:
                last_picked_count += 1
            else:
```

```
                last_picked = tree
                last_picked_count = 1

        max_length_path = max(curr, max_length_path)

    return max_length_path
```

Since we need only iterate over the input once, and update a constant number of variables in the process, this solution will take `O(N)` time and `O(1)` additional space.
