2103. Rings and Rods

Easy | Hash Table | String

Problem Description

In this problem, we are dealing with `n` rings of three possible colors: red, green, or blue. These rings are placed on ten rods that are numbered from `0` to `9`. We need to determine the number of rods that have at least one ring of each color on them.

We're given a string `rings`, which is twice as long as the number of rings since it contains pairs of characters. Each pair consists of a color character (`'R'`, `'G'`, or `'B'`) followed by a digit character (`'0'` through `'9'`) that represents the rod number on which the ring is placed. For example, if the string is `"R3G2B1"`, it means we have a red ring on rod `3`, a green ring on rod `2`, and a blue ring on rod `1`.

The task is to parse this string, arrange the rings according to their rods and colors, and finally count how many rods have a full set of all three colors.

Intuition

To address this problem, we need to organize the rings in a manner that easily lets us count the rods with a full set of colors. A good approach for such organization issues is to use a hash table, in this case in the form of a dictionary.

Here's the step-by-step intuition:

- Create a hash table to keep track of the rings on each rod. In Python, we can use a dictionary (`mp`) where each key is a rod number and each value is a set of colors on that rod.
- Iterate through the `rings` string, taking two characters at a time (since the rings are described by pairs of characters).
- The first character of the pair represents the color of the ring, and the second character represents the rod it is placed on.
- For each pair, add the ring color to the set that corresponds to the rod number in our hash table.
- Since a set automatically handles duplicates (it only keeps unique items), we don't need to worry about counting multiple rings of the same color on the same rod.
- Once we have processed all the pairs, we count how many rods (keys in the dictionary) have all three colors in their associated set of colors.
- There should be exactly three colors in a set to indicate that a rod has a full set.

Solution Approach

The solution uses a `defaultdict` from Python's `collections` module, which is a subclass of the built-in `dict` that is provided with a default type when a key does not exist. In this case, the default type is `set`. This choice of data structure is likely because sets in Python handle uniqueness automatically and provide an efficient way to store the colors associated with each rod without duplicates.

Here is the breakdown of the solution algorithm:

- A `defaultdict` called `mp` where keys will be integers corresponding to rod numbers (0 to 9), and the values will be sets that store the color characters (`'R'`, `'G'`, `'B'`).
- The input `rings` string is parsed two characters at a time with a for loop that starts at index `1` and increments by `2` to ensure that we look at every color-position pair. This loop iterates over the indices of the rod characters.

```
for i in range(1, len(rings), 2):
```

- On each iteration, the first character (`rings[i - 1]`) represents the color of the ring, and the second character (`rings[i]`) converted to an integer, represents the rod number.
- The color character is added to the set of colors corresponding to the rod number in our `defaultdict`. Since we are using a set, if the same color is added again to the same rod, it won't change the set contents because of the uniqueness constraint of the set.

```
c = int(rings[i])
mp[c].add(rings[i - 1])
```

- After all pairs are processed, we count the number of entries in `mp` whose values are sets with a length of `3`. This length check ensures that a rod has all three colors (`'R'`, `'G'`, and `'B'`). We use a generator expression within the `sum` function to do this count concisely.

```
return sum(len(v) == 3 for v in mp.values())
```

Essentially, this solution capitalizes on the Python set and `defaultdict` behavior to model the problem as a simple counting problem.

Example Walkthrough

Let's walk through a small example using the solution approach described above. Consider the string `rings = "B0R0G0R9G9"`. This string means we have rings of the following colors on the corresponding rods: blue, red, and green on rod `0`, and blue, red, and green on rod `9`.

Now, let's follow the steps outlined in the solution approach:

1. Initialize a `defaultdict` of sets, `mp`.
2. We process the `rings` string two characters at a time. For the first two characters, `'B0'`, we add the color `'B'` (blue) to the set associated with rod `0` in `mp`.
3. For the next two characters, `'R0'`, we add the color `'R'` (red) to the set for rod `0` in `mp`.
4. Continuing this way, `'G0'` adds the color `'G'` (green) to the set for rod `0`.
5. At this point, the set for rod `0` contains all three colors: `{'B', 'R', 'G'}`.
6. For the characters `'B9'`, `'R9'`, and `'G9'`, we add each of the three colors to the set for rod `9` in `mp`, resulting in another set with all three colors: `{'B', 'R', 'G'}`.
7. Now that we've processed the entire string, we have a `defaultdict` `mp` that looks like this:

```
{
    0: {'B', 'R', 'G'},
    9: {'B', 'R', 'G'}
}
```

8. The final step is to count the number of entries in `mp` with sets of length `3`. Both rods `0` and `9` have sets of `3` unique colors, indicating a complete set of colors.
9. Thus, we count `2` rods that have at least one ring of each color, and the final result would be `2` for this example.

Following these steps with actual code will produce the expected result, demonstrating that the solution corrects the problem.

Solution Implementation

Python

```
from collections import defaultdict

class Solution:
    def countPoints(self, rings: str) -> int:
        # Initialize a dictionary to hold sets of colors for each rod position
        rod_to_colors = defaultdict(set)

        # Iterate over each color-position pair
        for i in range(1, len(rings), 2):
            # Retrieve the position of the rod by converting the string number to int
            rod_position = int(rings[i])

            # Add the color (denoted by a letter) to the set of colors for this rod
            rod_to_colors[rod_position].add(rings[i - 1])

        # Count the rods which have exactly 3 different colors (R, G, B)
        # and return the total count
        return sum(len(colors) == 3 for colors in rod_to_colors.values())
```

Java

```
class Solution {
    // Method to count the number of rods that have all three colors of rings
    public int countPoints(String rings) {
        // Initialize a map to store the rings on each rod with rod number as key
        Map<Integer, Set<Character>> rodToRingsMap = new HashMap<>();

        // Iterate over the string to populate the map with rods and their rings
        for (int i = 1; i < rings.length(); i += 2) {
            int rodNumber = rings.charAt(i) - '0'; // Get the rod number from the string
            // Add the ring color to the set associated with the current rod
            // If the rod does not exist in the map, it initializes with a new HashSet
            rodToRingsMap.computeIfAbsent(rodNumber, k -> new HashSet<>()).add(rings.charAt(i - 1));
        }

        int count = 0; // Counter for rods with all three colors
        // Loop through the values of the map
        for (Set<Character> ringsOnRod : rodToRingsMap.values()) {
            // If a rod has all three colors (R, G, B), increment the counter
            if (ringsOnRod.size() == 3) {
                count++;
            }
        }

        // Return the total count of rods with all three ring colors
        return count;
    }
}
```

C++

```
#include <unordered_map>
#include <unordered_set>
#include <string>

class Solution {
public:
    int countPoints(string rings) {
        // Create a map to store the sets of rings for each rod
        unordered_map<int, unordered_set<char>> rodToRingsMap;

        // Iterate through the string, considering pairs of ring color and rod
        for (int i = 1; i < rings.size(); i += 2) {
            // Convert the rod character to an integer
            // '0' character has an int value of 48 according to ASCII,
            // subtracting '0' translates char '0'-'9' to int 0-9
            int rod = rings[i] - '0';

            // Insert the ring color (rings[i - 1]) into the set belonging to the rod
            rodToRingsMap[rod].insert(rings[i - 1]);
        }

        // Initialize the answer to count the number of rods with all 3 colors of rings
        int completeRodsCount = 0;

        // Iterate through rods 0 to 9
        for (int rod = 0; rod < 10; ++rod) {
            // If the set contains all 3 colors, increase the count
            if (rodToRingsMap[rod].size() == 3) {
                ++completeRodsCount;
            }
        }

        // Return the number of rods that have all 3 colors of rings
        return completeRodsCount;
    }
};
```

TypeScript

```
function countPoints(rings: string): number {
    // Helper function to convert a color character to a unique number.
    const getColorCode = (color: string) => color.charCodeAt(0) - 'A'.charCodeAt(0);

    const lengthOfRings = rings.length;

    // Calculate a target number which reperesents having all three colors on a rod.
    const targetCombination = (1 << getColorCode('R')) + (1 << getColorCode('G')) + (1 << getColorCode('B'));

    // Initialize the count-array which holds the combination of colors on each of the 10 rods.
    const rodColorCounts = new Array(10).fill(0);

    // Iterate over pairs of characters (color, rod number) in the input string.
    for (let i = 0; i < lengthOfRings; i += 2) {
        // Bitwise OR the color's code into the count for the corresponding rod.
        const rod = parseInt(rings[i + 1]);
        rodColorCounts[rod] |= 1 << getColorCode(rings[i]);
    }

    // Use reduce to tally rods that have all three colors.
    return rodColorCounts.reduce((rodCountAccumulator, currentRodValue) => {
        return rodCountAccumulator + (currentRodValue === targetCombination ? 1 : 0);
    }, 0);
}
```

```
from collections import defaultdict

class Solution:
    def countPoints(self, rings: str) -> int:
        # Initialize a dictionary to hold sets of colors for each rod position
        rod_to_colors = defaultdict(set)

        # Iterate over each color-position pair
        for i in range(1, len(rings), 2):
            # Retrieve the position of the rod by converting the string number to int
            rod_position = int(rings[i])

            # Add the color (denoted by a letter) to the set of colors for this rod
            rod_to_colors[rod_position].add(rings[i - 1])

        # Count the rods which have exactly 3 different colors (R, G, B)
        # and return the total count
        return sum(len(colors) == 3 for colors in rod_to_colors.values())
```

Time and Space Complexity

The given Python code uses a loop that iterates over the string `rings` and a dictionary to store the colors of rings present on each rod.

Time Complexity

The time complexity of the code is determined by the for loop, which iterates over the characters of the string `rings`, but skipping every other character (as it increments `i` by 2 for each iteration). If `N` is the length of the `rings` string, then the number of iterations of the loop is approximately `N/2`. Each operation within the loop (accessing characters, adding them to the set, and incrementing the counter) is performed in constant time, O(1). As such, the total time complexity is O(N/2), which simplifies to `O(N)`.

Space Complexity

The space complexity is determined by the additional data structures used, which, in this case, is the dictionary `mp`. The dictionary size depends on the number of unique rods (digits) in the `rings` string. In the worst case, each rod will have a set storing up to three colors (since there are only three different colors possible). Hence, the space occupied by the dictionary is proportional to the number of rods with an additional constant factor for the sets. If there are `k` different rods, the space complexity is `O(k)`. Moreover, considering that `k` is bounded by a constant (since rod digits in the string can range only from 0 to 9), the space complexity simplifies to `O(1)`.

Note: Another perspective on space complexity is that it can be considered as O(N) if we count the total number of color entries across all sets. However, since there can only be three entries per set and ten possible rods, the previous conclusion that it's O(1) holds for the maximum possible space taken by the sets.