1108. Defanging an IP Address

Easy String

Leetcode Link

Problem Description

The problem requires writing a function that takes a standard IPv4 address as input and outputs a modified version of this IP address, called the "defanged" IP address. An IPv4 address is a string comprising four numbers separated by periods (e.g., "192.168.0.1"). The defanged version of this IP address is one where every period, ".", is replaced with "[.]" (e.g., "192[.]168[.]0[.]1"). This is usually done to prevent the IP address from being accidentally used in a context where it might represent an actionable hyperlink, such as in documentation or logs.

Intuition

The intuition behind the solution is straightforward. Since the task is to replace every period in the IP address with "[.]", Python's string method `replace` can be utilized. This inbuilt method takes two parameters: the substring we want to replace (`"."`) and the substring we want to insert in place of it (`"[.]"`). By calling `address.replace('.', '[.]')`, we are telling the program to go through the entire string `address` and replace instances of `'.'` with `'[.]'`. This method scans the string from the beginning to the end while applying the replacement and builds the new defanged IP address string to be returned.

Solution Approach

The solution provided is simple and leverages Python's built-in string methods. Here is an explanation of the steps involved in the implementation of the solution provided:

1. The `defangIPaddr` function takes in a parameter `address`, which is a string representing a valid IPv4 address.
2. To 'defang' this IP address, the `replace` method of Python strings is called on `address`. The replace method is a common string operation that searches for occurrences of a specified substring within the string, and replaces each occurrence with another specified substring.
3. In this instance, we are telling the `replace` method to search for the substring `'.'` (which represents the periods in the IP address) and replace each occurrence with the substring `'[.]'`.
4. The `replace` method does not change the original string but returns a new string with all the replacements made, which is exactly what is required for this problem. This is because strings in Python are immutable, meaning they cannot be altered once created.
5. The result is then returned as the output of the function. This single line of code: `return address.replace('.', '[.]')` is the complete implementation of the solution.

No external data structures, complex patterns, or algorithms are involved in this solution. It's a direct application of a string method to perform the required transformation.

Example Walkthrough

Let's walk through a small example to illustrate the solution approach for defanging an IPv4 address.

Suppose we are given the following IP address as input:

```
"123.45.67.89"
```

Our goal is to take this IP and transform it into a "defanged" version. According to the problem description, defanging an IP means replacing every period `"."` with `"[.]"`. We want to end up with:

```
"123[.]45[.]67[.]89"
```

Here's how we apply the solution approach:

1. We call the function `defangIPaddr`, passing in our example IP address:

   ```
   defanged_ip = defangIPaddr("123.45.67.89")
   ```

2. Within the `defangIPaddr` function, we use Python's `replace` method on the passed address:

   ```
   return address.replace('.', '[.]')
   ```

3. The `replace` method searches for each instance of the period `.` in the string `"123.45.67.89"` and replaces it with `"[.]"`.
4. The `replace` method processes the string from start to finish and performs the replacement sequentially:
   - After encountering the first period, we get `"123[.]45.67.89"`.
   - After the second period, it becomes `"123[.]45[.]67.89"`.
   - Lastly, after the third period, we achieve the final result: `"123[.]45[.]67[.]89"`.
5. The new defanged IP address string `"123[.]45[.]67[.]89"` is then stored in `defanged_ip`.
6. The function returns this new string, completing the operation.

By calling `defangIPaddr` with our example input, we have converted the standard IP address into its defanged format without altering the original string, using a very straightforward and efficient method.

Python Solution

```python
class Solution:
    def defangIPaddr(self, address: str) -> str:
        # Replace all instances of the '.' character with '[.]'
        # This is used to 'defang' an IP address to prevent it from being
        # instantly recognizable as an IP address and might help to prevent
        # some simplistic automated systems from detecting it.
        defanged_address = address.replace('.', '[.]')

        # Return the modified IP address
        return defanged_address

```

Java Solution

```java
class Solution {
    // Function to defang an IPv4 address
    public String defangIPaddr(String address) {
        // Replace each period '.' in the address with '[.]'
        String defangedAddress = address.replace(".", "[.]");

        // Return the defanged IP address
        return defangedAddress;
    }
}

```

C++ Solution

```cpp
class Solution {
public:
    // Function to convert a standard IP address into a defanged version
    // where every period '.' is replaced with "[.]"
    string defangIPaddr(string address) {
        // Iterate through the string in reverse order using the index
        for (int i = address.size() - 1; i >= 0; --i) { // Fixed the start index to size - 1
            // Check if the current character is a period '.'
            if (address[i] == '.') {
                // Replace the period with "[.]" at the current index
                address.replace(i, 1, "[.]"); // Use '1' for the length of the period to be replaced
            }
        }

        // Return the modified address with defanged IP
        return address;
    }
};

```

Typescript Solution

```typescript
// This function takes an IP address as a string and replaces every period ('.') with '[.]'
// to 'defang' it, which is often done to prevent URLs or IPs from being automatically
// hyperlinked or processed by text parsers.
// @param {string} address - The original IP address string to be defanged.
// @return {string} The defanged IP address.
function defangIPaddr(address: string): string {
    // Split the original IP address into an array of strings, breaking it at each period ('.').
    const addressParts: string[] = address.split('.');

    // Join the array of string parts back together, inserting '[.]' between the elements, effectively
    // replacing periods with '[.]'.
    const defangedAddress: string = addressParts.join('[.]');

    // Return the resulting defanged IP address.
    return defangedAddress;
}

```

Time and Space Complexity

The time complexity of the `defangIPaddr` method is `O(n)`, where `n` is the length of the input string `address`. This is because `str.replace()` is a linear-time operation that goes through each character in the input string once to make the replacements.

The space complexity is also `O(n)`, as the `replace` function creates a new string with the `'[.]'` replacing each `'.'` in the original `address`. In the worst case, if there are `k` occurrences of `'.'` in the address, the new string will have a length of `n + 3k` (since each `'.'` is replaced by `'[.]'` which is three characters long), which is still linear with respect to the size of the input.