Here’s a Complete documentation for your **FizzBuzz** problem solution:

1. Problem Statement

Given an integer `n`, return a string array `answer` (1-indexed) where:

- `answer[i] == "FizzBuzz"` if `i` is divisible by both `3` and `5`.
- `answer[i] == "Fizz"` if `i` is divisible by `3`.
- `answer[i] == "Buzz"` if `i` is divisible by `5`.
- `answer[i] == i` (as a string) if none of the above conditions are true.

Example 1:

Input: `n = 3`
Output: `["1", "2", "Fizz"]`

Example 2:

Input: `n = 5`
Output: `["1", "2", "Fizz", "4", "Buzz"]`

Example 3:

Input: `n = 15`
Output: `["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz", "11", "Fizz", "13", "14", "FizzBuzz"]`

Constraints:

- $1 <= n <= 10^4$

2. Intuition

The problem asks us to generate a sequence where:

- Numbers divisible by both 3 and 5 should be replaced with `"FizzBuzz"`.
- Numbers divisible by only 3 should be replaced with `"Fizz"`.
- Numbers divisible by only 5 should be replaced with `"Buzz"`.
- Other numbers should remain as-is, converted to a string.

The key intuition is to loop through all the numbers from 1 to `n`, check divisibility by 3 and 5, and append the appropriate result to the output list.

3. Key Observations

- Numbers divisible by both 3 and 5 should output "FizzBuzz".
- Numbers divisible by just one of the numbers, 3 or 5, need to output "Fizz" or "Buzz" respectively.
- All other numbers are represented by their string equivalents.
- We need to account for divisibility for every number between 1 and n to ensure correct results.

4. Approach

Step-by-Step:

1. **Loop through numbers from 1 to n:**
    - Start by initializing an empty list to store the results.
    - For each number in the range, check:
        - If the number is divisible by both 3 and 5, append "FizzBuzz".
        - If it's divisible by only 3, append "Fizz".
        - If it's divisible by only 5, append "Buzz".
        - Otherwise, append the number itself as a string.
2. **Return the result list** after the loop finishes.

Example Walkthrough:

For n = 5:

- For i = 1: 1 is neither divisible by 3 nor 5, so append "1".
- For i = 2: 2 is neither divisible by 3 nor 5, so append "2".
- For i = 3: 3 is divisible by 3, so append "Fizz".
- For i = 4: 4 is neither divisible by 3 nor 5, so append "4".
- For i = 5: 5 is divisible by 5, so append "Buzz".

Output: ["1", "2", "Fizz", "4", "Buzz"].

5. Edge Cases

- **Edge Case 1**: n = 1
    - The list will only contain the string "1", since it doesn't meet any divisibility condition.
- **Edge Case 2**: n = 3
    - The result will be ["1", "2", "Fizz"], testing the divisibility condition for 3.
- **Edge Case 3**: n = 5
    - The result will be ["1", "2", "Fizz", "4", "Buzz"], testing the divisibility condition for 5.
- **Edge Case 4**: n = 15
    - This will test all the divisibility conditions, including both 3 and 5, resulting in ["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz", "11", "Fizz", "13", "14", "FizzBuzz"].

6. Complexity Analysis

Time Complexity:

- **O(n)**: We iterate through the numbers from 1 to n exactly once. Each iteration performs constant-time checks and appends to the list, so the overall time complexity is linear.

Space Complexity:

- **O(n)**: We store the result in a list that has n elements, so the space complexity is also linear.

7. Alternative Approaches

While the solution provided is optimal in terms of time and space complexity (both O(n)), other alternatives could involve:

- Using a generator to yield results instead of appending to a list (this could be useful if you need to process the results lazily).
- Storing the results in a different data structure like a deque (although for this problem, a list is the simplest and most efficient choice).

However, no significant improvements can be made in terms of time complexity since we have to check each number between 1 and n.

8. Code Implementation

```
from typing import List

class Solution:
    def fizzBuzz(self, n: int) -> List[str]:
        result = []
        for i in range(1, n + 1):
            if i % 3 == 0 and i % 5 == 0:
                result.append("FizzBuzz")
            elif i % 3 == 0:
                result.append("Fizz")
            elif i % 5 == 0:
                result.append("Buzz")
            else:
                result.append(str(i))
        return result
```

9. Test Cases

Test Case 1:

Input: n = 3
Expected Output: ["1", "2", "Fizz"]

Test Case 2:

Input: n = 5
Expected Output: ["1", "2", "Fizz", "4", "Buzz"]

Test Case 3:

Input: n = 15
Expected Output: ["1", "2", "Fizz", "4", "Buzz", "Fizz", "7", "8", "Fizz", "Buzz", "11", "Fizz", "13", "14", "FizzBuzz"]

Test Case 4:

Input: n = 1
Expected Output: ["1"]

10. Final Thoughts

The FizzBuzz problem is a classic exercise in condition checking and list manipulation. The solution provided is efficient and works well within the given constraints ($1 <= n <= 10^4$). The problem is simple enough to demonstrate basic logic and control flow, yet it can be expanded into more complex variations, such as adding more divisibility conditions or working with larger ranges.

The approach outlined here efficiently handles the task, and the time complexity of $O(n)$ ensures it scales well even for the upper limit of n.