








412. Fizz Buzz

Sept. 23, 2018 | 105.3K views

 Previous

 Next



Average Rating: 4.60 (53 votes)

Write a program that outputs the string representation of numbers from 1 to n .

But for multiples of three it should output "Fizz" instead of the number and for the multiples of five output "Buzz". For numbers which are multiples of both three and five output "FizzBuzz".

Example:

```
n = 15,

Return:
[
    "1",
    "2",
    "Fizz",
    "4",
    "Buzz",
    "Fizz",
    "7",
    "8",
    "Fizz",
    "Buzz",
    "11",
    "Fizz",
    "13",
    "14",
    "FizzBuzz"
]
```

Solution

You must have played FizzBuzz as kids. FizzBuzz charm never gets old. And so here we are looking at how you can take on one step at a time and impress your interviewer with a better and neat approach to solve this problem.

Approach 1: Naive Approach

Intuition

The moment you hear of FizzBuzz you think whether the number is divisible by 3, 5 or both.

Algorithm

1. Initialize an empty answer list.
2. Iterate on the numbers from 1...N.
3. For every number, if it is divisible by both 3 and 5, add FizzBuzz to the answer list.
4. Else, Check if the number is divisible by 3, add Fizz.
5. Else, Check if the number is divisible by 5, add Buzz.
6. Else, add the number.

Java Python Copy

```
2 def fizzBuzz(self, n):
3     """
4     :type n: int
5     :rtype: List[str]
6     """
7     # ans list
8     ans = []
9
10    for num in range(1,n+1):
11
12        divisible_by_3 = (num % 3 == 0)
13        divisible_by_5 = (num % 5 == 0)
14
15        if divisible_by_3 and divisible_by_5:
16            # Divides by both 3 and 5, add FizzBuzz
17            ans.append("FizzBuzz")
18        elif divisible_by_3:
19            # Divides by 3, add Fizz
20            ans.append("Fizz")
21        elif divisible_by_5:
22            # Divides by 5, add Buzz
23            ans.append("Buzz")
24        else:
25            # Not divisible by 3 or 5, add the number
26            ans.append(str(num))
27
28    return ans
```

Complexity Analysis

- Time Complexity: $O(N)$
- Space Complexity: $O(1)$

Approach 2: String Concatenation

Intuition

This approach won't reduce the asymptotic complexity, but proves to be a neater solution when FizzBuzz comes with a twist. What if FizzBuzz is now FizzBuzzJazz i.e.

```
3 ---> "Fizz" , 5 ---> "Buzz", 7 ---> "Jazz"
```

If you try to solve this with the previous approach the program would have too many conditions to check:

1. Divisible by 3
2. Divisible by 5
3. Divisible by 7
4. Divisible by 3 and 5
5. Divisible by 3 and 7
6. Divisible by 7 and 3
7. Divisible by 3 and 5 and 7
8. Not divisible by 3 or 5 or 7.

This way if the FizzBuzz mappings increase, the conditions would grow exponentially in your program.

Algorithm

Instead of checking for every combination of these conditions, check for divisibility by given numbers i.e. 3, 5 as given in the problem. If the number is divisible, concatenate the corresponding string mapping Fizz or Buzz to the current answer string.

For eg. If we are checking for the number 15, the steps would be:

```
Condition 1: 15 % 3 == 0 , num_ans_str = "Fizz"
Condition 2: 15 % 5 == 0 , num_ans_str += "Buzz"
=> num_ans_str = "FizzBuzz"
```

So for FizzBuzz we just check for two conditions instead of three conditions as in the first approach.

Similarly, for FizzBuzzJazz now we would just have three conditions to check for divisibility.

Java Python Copy

```
4 :type n: int
5 :rtype: List[str]
6 """
7 # ans list
8 ans = []
9
10 for num in range(1,n+1):
11
12     divisible_by_3 = (num % 3 == 0)
13     divisible_by_5 = (num % 5 == 0)
14
15     num_ans_str = ""
16
17     if divisible_by_3:
18         # Divides by 3
19         num_ans_str += "Fizz"
20     if divisible_by_5:
21         # Divides by 5
22         num_ans_str += "Buzz"
23     if not num_ans_str:
24         # Not divisible by 3 or 5
25         num_ans_str = str(num)
26
27     # Append the current answer str to the ans list
28     ans.append(num_ans_str)
29
30 return ans
```

Complexity Analysis

- Time Complexity: $O(N)$
- Space Complexity: $O(1)$

Approach 3: Hash it!

Intuition

This approach is an optimization over approach 2. When the number of mappings are limited, approach 2 looks good. But what if you face a tricky interviewer and he decides to add too many mappings?

Having a condition for every mapping is not feasible or may be we can say the code might get ugly and tough to maintain.

What if tomorrow we have to change a mapping or may be delete a mapping? Are we going to change the code every time we have a modification in the mappings?

We don't have to. We can put all these mappings in a Hash Table.

Algorithm

1. Put all the mappings in a hash table. The hash table fizzBuzzHash would look something like { 3: 'Fizz', 5: 'Buzz' }.
2. Iterate on the numbers from 1...N.
3. For every number, iterate over the fizzBuzzHash keys and check for divisibility.
4. If the number is divisible by the key, concatenate the corresponding hash value to the answer string for current number. We do this for every entry in the hash table.
5. Add the answer string to the answer list.

```
This way you can add/delete mappings to/from the hash table and not worry about changing the code.
```

So, for FizzBuzzJazz the hash table would look something like { 3: 'Fizz', 5: 'Buzz', 7: 'Jazz' }

Java Python Copy

```
4 :type n: int
5 :rtype: List[str]
6 """
7 # ans list
8 ans = []
9
10 # Dictionary to store all fizzbuzz mappings
11 fizz_buzz_dict = { 3 : "Fizz", 5 : "Buzz" }
12
13 for num in range(1,n+1):
14
15     num_ans_str = ""
16
17     for key in fizz_buzz_dict.keys():
18
19         # If the num is divisible by key,
20         # then add the corresponding string mapping to current num_ans_str
21         if num % key == 0:
22             num_ans_str += fizz_buzz_dict[key]
23
24     if not num_ans_str:
25         num_ans_str = str(num)
26
27     # Append the current answer str to the ans list
28     ans.append(num_ans_str)
29
30 return ans
```

Complexity Analysis

- Time Complexity: $O(N)$
- Space Complexity: $O(1)$

Rate this article: ★★★★★

Comments: 41 Sort By ▾



Type comment here.. (Markdown is supported)

 Preview  Post



smitti9 ★ 299  September 26, 2018 8:25 AM

I solved this problem and I'm now an expert at programming. Who wants to join me and make facebook 2?

294     Reply

SHOW 9 REPLIES



shashwatblack ★ 118  January 20, 2019 11:19 PM

Do remember that dicts do not preserve order. keys() could return the keys in different order than you expected. You should use OrderedDict.

74     Reply

SHOW 5 REPLIES



rhliu ★ 38  February 17, 2019 12:55 AM

approach 3 need to use LinkedHashMap. Otherwise it is wrong.

40     Reply

SHOW 2 REPLIES



NSMobileCS ★ 50  September 25, 2018 9:08 AM

A programming job I applied for asked "What's something that makes you unique? Be creative. (150 character limit)"

After overthinking it, I figured crammng an answer to 'FizzBuzz' in there would be an interesting challenge, hopefully showcasing some programming ability &/or sense of irony that might work for me.

34     Reply

SHOW 4 REPLIES



lanxiang123 ★ 45  October 28, 2019 5:39 AM

If the Naive solution has the same runtime and space complexity it's not so naive now is it

31     Reply

SHOW 2 REPLIES



terrible_whiteboard ★ 635  May 19, 2020 6:14 PM

I made a video if anyone is having trouble understanding the solutions (clickable link)

<https://youtu.be/lyZvPH5EBFU>

21     Reply

Read More



jamestrade ★ 8  December 2, 2018 10:46 AM

Why is Space Complexity for approach 1 $O(1)$ if we are adding to memory an array that depends of the size of n ? Shouldn't it be $O(n)$ as well?

7     Reply

SHOW 3 REPLIES



AILUNGANGEL ★ 16  October 21, 2018 2:31 PM

The third approach cannot guarantee the sequence, 5 may be checked before 3, therefore 15 is not represented as FizzBuzz but BuzzFizz, is that acceptable?

7     Reply

SHOW 5 REPLIES



Rai ★ 9  April 26, 2019 8:07 PM

In approach 1, How space complexity will be $O(1)$? We are creating 'ans' list of N size, then it should be $O(N)$ right?

6     Reply

SHOW 3 REPLIES



Swapnil510 ★ 20  January 9, 2019 9:15 AM

In Approach 3, more computation (divisions) are performed. We divide every number by 3 & 5. Instead, if-else ladder like if(n%15==0) elif(n%5==0) elif(n%3==0) will end up in much less divisions.

5     Reply

SHOW 1 REPLY

 1  2  3  4  5  6