Solution Review: Nested Loop with Multiplication (Advanced)

This review provides a detailed analysis of the different ways to solve the Nested Loop with Multiplication challenge.

We'll cover the following

Solution

```
n = 10  # can be anything
sum = 0
pie = 3.14
for i in range(n):
    j = 1
    while j < i:
        sum += 1
        j *= 2
    print(sum)

```

In the main function, the outer loop is $O(n)$ as it iterates over `n`. The inner while loop iterates over `i` which is always **less than** `n` and `j` is doubled each time, therefore we can say that it is $O(log_2(n))$. Thus, the total time complexity of the program given above becomes:

$$O(nlog_2(n))$$

The running time complexity mentioned above is a **loose bound**. To evaluate a *tighter* bound on the running time of the program given above, let's look at the inner loop once again.

The inner loop depends upon `j` which is less than `i` and is multiplied by `2` in each iteration. This means that the complexity of the inner loop is $O(log_2(\text{i}))$. But, the value of `i` at each iteration, of the outer loop, is different. The total complexity of the inner loop in terms of `n` can be calculated as such:

$$n \times log_2(i) \Rightarrow \sum_{i=1}^{n} log_2(i)$$

$$\Rightarrow log_2(1) + ... + log_2(n-1) + log_2(n)$$

as we know that; $log(a) + log(b) = log(ab)$

so above expression becomes:

$$\Rightarrow log_2(1 \times 2 \times ... \times (n-1) \times (n))$$

$$\Rightarrow log_2(n!)$$

> log($n!$) = $\sum_{k=1}^{n} log(k) < \sum_{k=1}^{n} log(n)$ = $nlog(n)$

Thus, the total time complexity of the inner-loop (considering the outer-loop) is $O(log_2(n!))$

The overall number of executions are summarized in the table below:

Statement	Number of Executions
n = 10	1
sum = 0	1
pie = 3.14	1
i	n
range(n)	1
j=1	n
print(pie);	$\frac{n}{3}$
while j < i:	$log_2(n!)$
sum+=1	$log_2(n!)$
j*=2	$log_2(n!)$
print(sum)	n

Time Complexity

As mentioned above, the running time complexity of the program is:

$$\text{Time Complexity} = 3 + 4n + 3log_2(n!)$$

To find the Big O time complexity,

1. Drop the leading constants $\Rightarrow n + log_2(n!)$
2. Drop the lower order terms $\Rightarrow log_2(n!)$

The Big O time complexity of the above is $\Rightarrow O(log_2(n!))$.

This is a **tight upper-bound** to the growth function of this program. And as mentioned above, the **loose upper bound** is $O(nlog_2(n))$.

Back

Challenge 6: Nested Loop with Multipl...

Next

Challenge 7: Nested Loop with Multipl...

Completed