Solution Review: Big O of Nested Loop with Multiplication

This review provides a detailed analysis of the different ways to solve the Big O of Nested Loop with Multiplication Quiz!

We'll cover the following

Solution

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

```

Explanation#

The answer is $O(n)$. Have a look at the slides below for an in-depth explanation of the answer.

> In the slides below, **rtc** abbreviates the *running time complexity*.

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

Running time complexity

0

Lets dry run this code to calculate its running time complexity

1 of 61

Time Complexity#

The above slides give a detailed, step-by-step analysis of the c... H... ve provide a more summarized version.

The outer loop here runs $log(n)$ times. In the first iteration of the outer loop, the body of the inner loop runs once. In the second iteration, it runs twice, and so on. The number of executions of the body of the inner loop increases in powers of 2. So, if k is the number of iterations of the outer loop, the body of the inner loop runs a total of $1 + 2 + 4 + 8 + \cdots + 2^k$ times. This geometric series sums to $2^{k+1} - 1$. The inner loop condition requires that in the last time the inner loop runs, it runs at most n times. This requires $2^k < n$, i.e., $k < log_2 n$, or $k = \lfloor log_2 n \rfloor$. This means that the geometric series sum to $2^{\lfloor log_2 n \rfloor + 1} - 1$ or $2^{\lceil log_2 n \rceil} - 1$. In other words, the sum is **at least** $n - 1$. But we need an upper bound on this value, for Big O.

As you vary the value of n, you will notice the following behavior:

n	**Iterations of inner loop**
1	0
2	1
3	3
4	3
5	7
6	7

n	**Iterations of i[illegible]o**
7	7
8	7
9	15
10	15
11	15
12	15
13	15
14	15
15	15
16	15
17	31

The number of iterations of the inner loop changes on integer powers of 2, as you would expect with the $\lceil n \rceil$ exponent. At $n = 4$, the number of iteration is 3, at $n = 8$, it is 7 etc. So, on values of n that are integer powers of 2, the number of iterations is $n - 1$. That conforms to our lower bound on the number of iterations. The upper bound is evident on $n = 2^i + 1$, where

$i = 1, 2, 3, 4, ...$, i.e., $n = 3, 5, 9, 17, ...$ You will notice that a[illegible]e[illegible]u of n, the number of iterations of the inner loop is one less than the next higher integer power of 2, i.e., $2(n-1)-1 = 2n-3$.

Thus, the number of iterations of the inner loop body is at least $n-1$ and at most $2n-3$.

As described in the slides, the outer for loop statements account for $4\lceil log_2 n\rceil$, and the total contribution of the inner loop is at least $8n-12$. Thus, the total running time is at least $4\lceil log_2 n\rceil + 8n - 12$. Dropping the constants and the lower order terms gives us n.

Hence, the Big O Time Complexity is $O(n)$

Back

Challenge 3: Big O of Nested Loop wit...

Next

Challenge 4: Nested Loop with Multipl...

Completed

Report an Issue