

Forum: [Hack Forums](#)

Thread: [Calculating time complexity O\(n\)](#)

TomDH - February 28, 2025, 9:51 am

Hey there, I'm preparing for a test with a friend
And we came across this question from an earlier test
Gemini and gpt give different answers
Would love some advice
I need to calculate the time complexity range of this code:
I think it's $n^3 \cdot c \cdot (\log \text{ of } n)$
He thinks it's $n^4 \cdot \log \text{ of } n$
Code

```
int func(int n, int c){  
    int s=0, m=0;  
    for(int i=4; i<n*n; i++){  
        for(int j=5; j<c*i; j*=c){  
            if(j^(c*n-1)){  
                for(int k=0;k<c*n;k++){  
                    m++;  
                    break;  
                }  
                for(int k=0;k<c*n;k++){  
                    s++;  
                }  
            }  
        }  
        return s>m? s:m;  
    }  
}
```

WhiteTV - February 28, 2025, 2:46 pm

I asked DeepSeek and got the following response:

"The time complexity of the given function is $O(c \cdot n^3 \cdot \log(n))$.

Let's analyze the nested loops:

1. The outer loop runs from $i=4$ to $i<n*n$, giving approximately n^2 iterations.
2. The inner loop starts with $j=5$ and multiplies j by c in each iteration ($j*=c$), creating a logarithmic pattern. For each value of i , this loop runs approximately $\log_c(c*i)$ times, which equals $1 + \log_c(i)$.
3. The third loop runs $c*n$ times for each iteration of the inner loop where certain conditions are met.

When we combine these factors:

- Outer loop: $O(n^2)$
- Inner loop: $O(\log_c(i))$ where i ranges from 4 to n^2
- Third loop: $O(c*n)$ for each valid inner loop iteration

The total complexity is the sum over i from 4 to n^2 of $(\log_c(i) \cdot c \cdot n)$, which simplifies to $O(c \cdot n^3 \cdot \log(n))$.

This analysis accounts for all nested loops and their respective iteration patterns, with the dominant factor being the combination of the quadratic outer loop, logarithmic inner loop, and the linear third loop."

I hope that could be of help with the explanation!

TomDH - March 1, 2025, 4:58 pm

(Feb 28, 2025 03:46 PM)WhiteTV Wrote: [\[CITE\]](#) I asked DeepSeek and got the following response:

"The time complexity of the given function is $O(c \cdot n^3 \log(n))$.

Let's analyze the nested loops:

1. The outer loop runs from $i=4$ to $i < n \cdot n$, giving approximately n^2 iterations.
2. The inner loop starts with $j=5$ and multiplies j by c in each iteration ($j*=c$), creating a logarithmic pattern. For each value of i , this loop runs approximately $\log_c(c \cdot i)$ times, which equals $1 + \log_c(i)$.
3. The third loop runs $c \cdot n$ times for each iteration of the inner loop where certain conditions are met.

When we combine these factors:

- Outer loop: $O(n^2)$
- Inner loop: $O(\log_c(i))$ where i ranges from 4 to n^2
- Third loop: $O(c \cdot n)$ for each valid inner loop iteration

The total complexity is the sum over i from 4 to n^2 of $(\log_c(i) \cdot c \cdot n)$, which simplifies to $O(c \cdot n^3 \log(n))$.

This analysis accounts for all nested loops and their respective iteration patterns, with the dominant factor being the combination of the quadratic outer loop, logarithmic inner loop, and the linear third loop."

I hope that could be of help with the explanation!

Thank you! that is the result I got myself, but after researching a lot more it turns out the correct answer is actually $n^3 \cdot c$, as the XOR condition is this is always triggered and then the middle loop runs only 1 time because of the break.
so its $n^2 \cdot n \cdot c$