① Algo 1: LOOP MYSTERY 1

FINDING WORST-CASE TIME COMPLEXITY

line 3 for $i = 1$ to n do ——— EXECUTES n times

line 4 for $j = i$ to n step k do ——— from i to n in steps k $\Rightarrow \frac{n-i}{k}$

line 5 for $l = 0$ to k do ——— $k+1$

Total Operations: $\sum_{i=1}^{n}\left(\frac{n-i}{k}\cdot(k+1)\right)$

Here, $+1$ is negligible for large k values

$$\Rightarrow \sum_{j=1}^{n}\left(\frac{n-i}{\cancel{k}}\cdot\cancel{k}\right)$$

$$\Rightarrow \sum_{i=1}^{n}(n-i) = \frac{n(n+1)}{2}$$

$$= n^2$$

$\therefore$ TIME COMPLEXITY is $O(n^2)$

(2) Algo 2: LoopMystery2

<u>Analysing</u> :

	ret	i
	0	1
1st iteration	0+1 = 1	2×1 = 2
	1+2 = 3	2×2 = 2^2
	3+4 = 7	$2^2 \times 2 = 2^3$
	⋮	⋮
	$\frac{M(M+1)}{2}$	2^k

∴ LOOP ENDS WHEN →

$$i > max$$

$$= 2^k > n^3 \qquad \because max = n^3$$

$$= k > \log n^3$$

$$= k > 3.\log n \rightarrow \text{using property}$$

$$\log(a^b) = b.\log a$$

∴ $O(\log n)$

TIME COMPLEXITY is $O(\log n)$

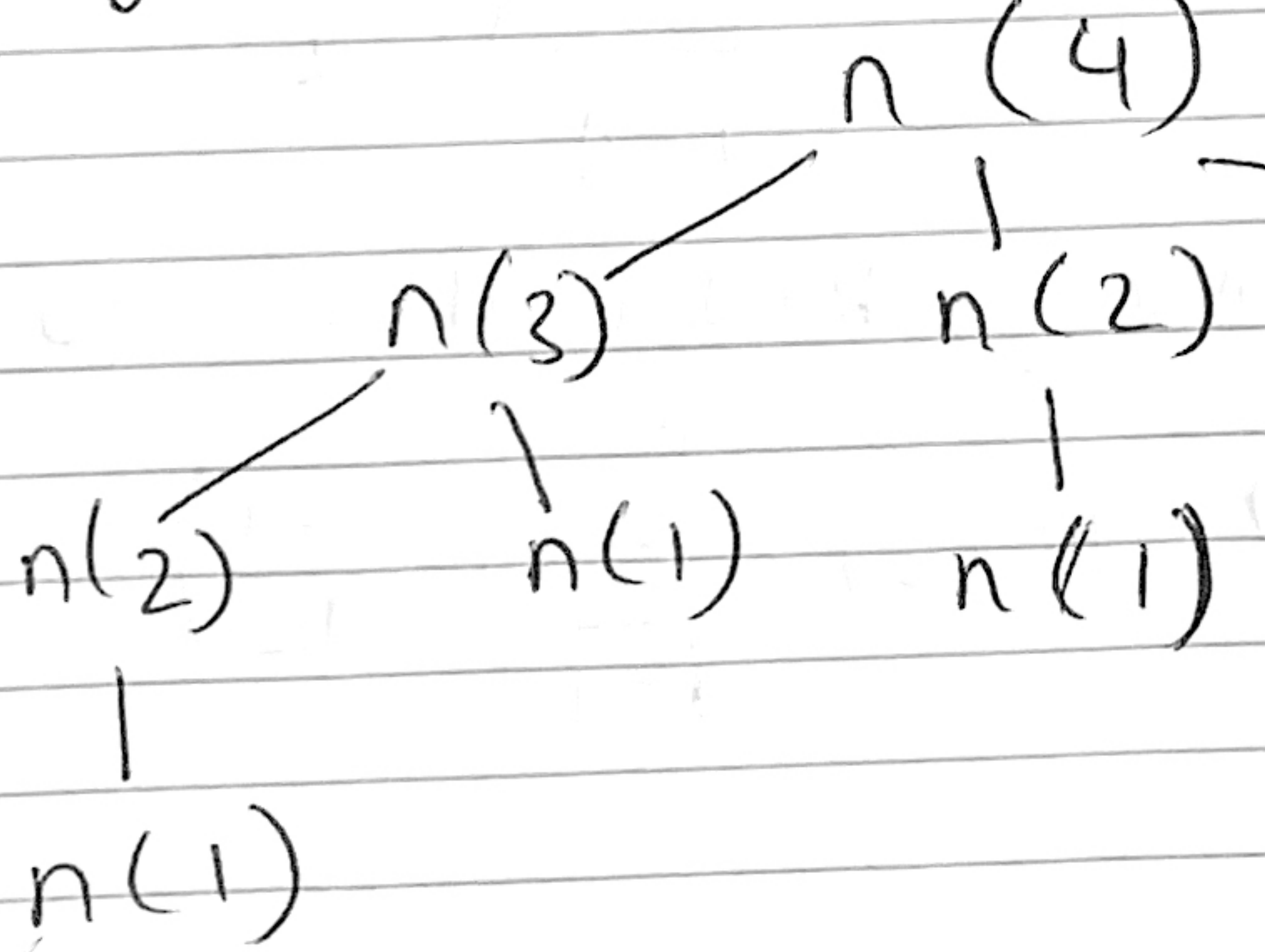
(3)

(a) The non-recursive part is the loop from 1 to $k-1$

→ The for loop will run $k-1$ times

∴ The non-recursive complexity of a single call is $\underline{\underline{O(k)}}$, as it depends on the value of k & ~~the~~ no. of iterations in the loop.

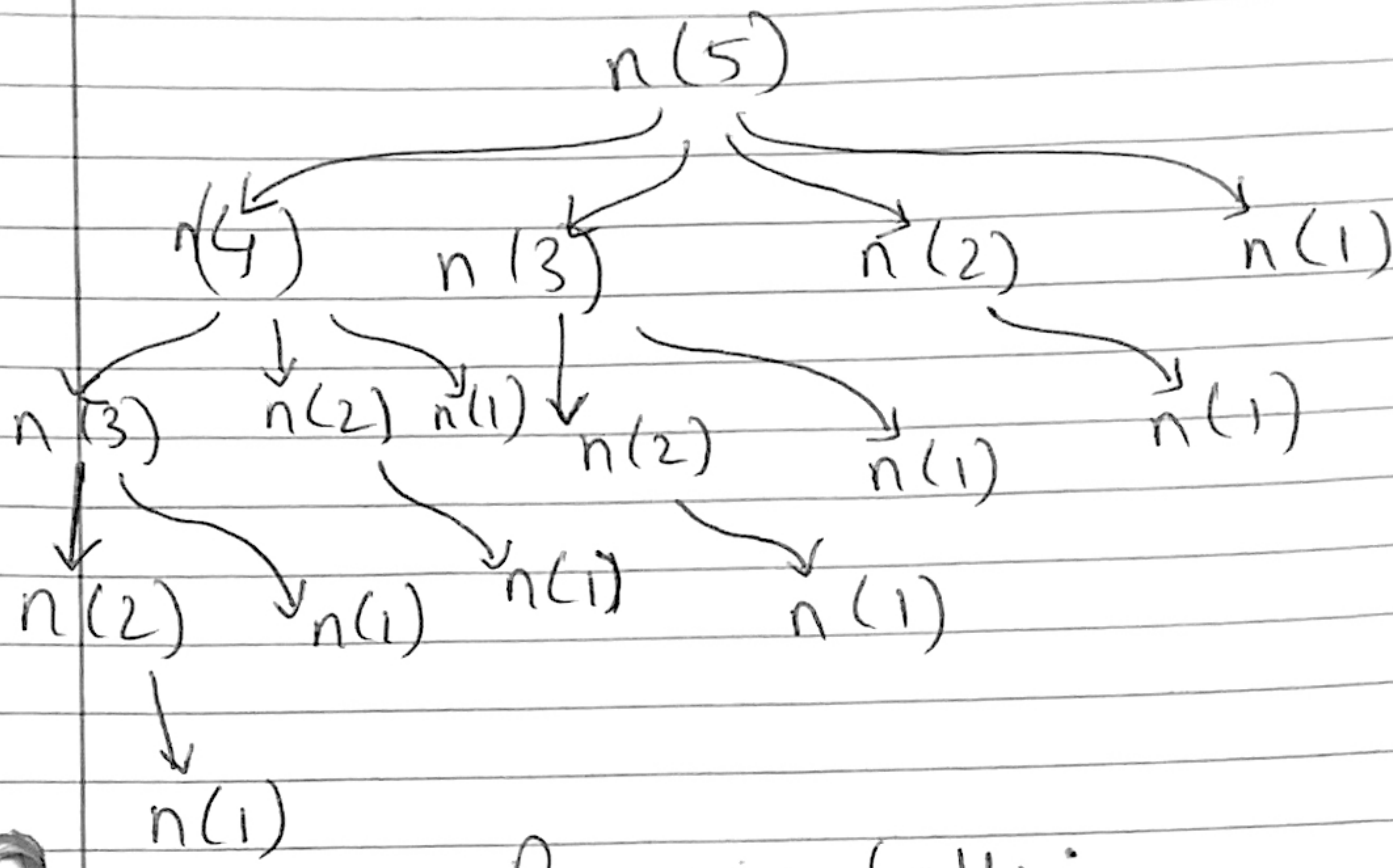
(b) for $n=4$



Count of Recursive calls:

n(4) : 1 call
n(3) : 1 call
n(2) : 2 calls
n(1) : 4 calls

(c) Recursion Tree for $n=5$



<u>Recursive Calls</u>:

n(5) : 1 call
n(4) : 1 call
n(3) : 2 calls
n(2) : 3 calls
n(1) : 8 calls

(d) $T(n) = T(n-1) + T(n-2) + \dots + T(1) + O(n)$

$$\therefore T(n) = \sum_{i=1}^{n-1} T(i) + O(n)$$