**Name: Kulsoom Khurshid**

**Registration #: SP20-BCS-044**

**ASSIGNMENT # 2**

**Question 1)**

Fill in the following table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem** | **Metric size for its input** | **Its basic operation** | **whether the basic operation count can**  **be different for inputs of the same size** | | |
| **Best case** | **Average case** | **Worst case** |
| Computing the sum of n numbers | n | Addition in the loop until loop variable is <= n | No | No | No |
| Word processing | n | Word addition | No | No | No |
| Finding the largest element in a list of n numbers | n | Comparison if the value in max variable is smaller than the value at ith index | No | No | No |
| Euclid’s algorithm | Magnitude of larger or smaller of two input numbers/sum of magnitude of two input numbers | Modulo division | No | No | O(log n) |
| Solving linear equations | X and y | Crammers rule | No | No | No |
| Displaying a scene graphically | Pixels | matrix | No | No | No |

**Question 2)**

Consider the following algorithms

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Time complexity with input size n** | | | **Time complexity with double the input size** | | | **Complexity increased by** |
| **Best case** | **Average case** | **Worst case** | **Best case** | **Average case** | **Worst case** |
| **Bubble sort** | n | n2 | n2 | 2n | 4n2 | 4n2 | 2 |
| **Selection sort** | n2 | n2 | n2 | 4n2 | 4n2 | 4n2 | 4 |
| **Insertion sort** | n | n2 | n2 | 2n | 4n2 | 4n2 | 2 |
| **Merge sort** | nlogn | nlogn | nlogn | nlog(4n2) | nlog(4n2) | nlog(4n2) | 2 |
| **Quick sort** | nlogn | n2 | nlogn | nlog(4n2) | 4n2 | nlog(4n2) | 4 |
| **Heap sort** | nlogn | nlogn | nlogn | nlog(4n2) | nlog(4n2) | nlog(4n2) | 4 |
| **Linear search** | 1 | n | n/2 | 2 | 2n | n | 2 |
| **Binary search** | 1 | logn | logn | 2 | Log(2n) | Log(2n) | 2 |
| **Matrix Multiplication** | n3 | n3 | n3 | 4 n3 | 4n3 | 4n3 | 4 |
| **Algorithm with complexity 2n in all cases.** | 2n | 2n | 2n | 4n | 4n | 4n | 2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Time complexity with input size n** | | | **Time complexity with input size increased by one** | | | **Complexity increased by** |
| **Best case** | **Average case** | **Worst case** | **Best case** | **Average case** | **Worst case** |
| **Bubble sort** | n | n2 | n2 | n+1 | (n+1)2 | (n+1)2 | 2n+1 |
| **Selection sort** | n2 | n2 | n2 | (n+1)2 | (n+1)2 | (n+1)2 | 2n+1 |
| **Insertion sort** | n | n2 | n2 | N+1 | (n+1)2 | (n+1)2 | 2n+1 |
| **Merge sort** | nlogn | nlogn | nlogn | nlog(n+1)+ log(n+1) | nlog(n+1)+ log(n+1) | nlog(n+1)+ log(n+1) | Log(n+1) |
| **Quick sort** | nlogn | n2 | nlogn | nlog(n+1)+ log(n+1) | (n+1)2 | nlog(n+1)+ log(n+1) | Log(n+1) |
| **Heap sort** | nlogn | nlogn | nlogn | nlog(n+1)+ log(n+1) | nlog(n+1)+ log(n+1) | nlog(n+1)+ log(n+1) | Log(n+1) |
| **Linear search** | 1 | n | n/2 | 2 | N+1 | N+1/2 | 1 |
| **Binary search** | 1 | logn | logn | 2 | log(n+1) | log(n+1) | 1 |
| **Matrix Multiplication** | n3 | n3 | n3 | (N+1)3 | (N+1)3 | (N+1)3 | 3n2+3n+1 |
| **Algorithm with complexity 2n in all cases.** | 2n | 2n | 2n | 2(n+1) | 2(n+1) | 2(n+1) | 2 |

**Question 3)**

1. What is the smallest value of n such that an algorithm whose running time is 100n2 runs faster than an algorithm whose running time is 2n on the same machine?

**Answer:**

Polynomial time complexity runs much faster than exponential, if the value of n is very large.

|  |  |  |
| --- | --- | --- |
| **100n2** | **>** | **2n** |
| n = 1, 100\*12 = 100 | > | n = 1, 21 = 2 |
| n = 3, 100\*32 = 900 | > | n = 3, 23 = 8 |
| n = 5, 100\*52 = 2500 | > | n = 5, 25 = 32 |
| n = 7, 100\*72 = 4900 | > | n = 7, 27 = 128 |
| n = 9, 100\*92 = 8100 | > | n = 9, 29 = 512 |
| n = 11, 100\*112 = 12100 | > | n = 11, 211 = 2048 |
| n = 13, 100\*132 = 16900 | > | n = 13, 213 = 8192 |
| n = 14, 100\*142 = 19600 | > | n = 14, 214 = 16384 |
| n = 15, 100\*152 = 22500 | < | n = 15, 215 = 32768 |

At n = 15, 100n2 runs faster than 2n on the same machine.

1. Suppose we are comparing implementations of two algorithms Alg1 and Alg2 on the same machine. For inputs of size n, Alg1 runs in 8n 2 steps, while Alg2 runs in 64n lgn steps. For which values of n does Alg1 beat Alg2?

**Answer:**

Alg1 <= Alg2

8n2 <= 64 n log n

n2 <= n log n

n2 <= 8 n log n

n <= 8 log n

n <= 8 log n

This equality holds for 2 ≤ n ≤ 43

**Question 4)**

1. **Answer:**

Given:

T(N) = 1 ms

N = 1000

n = 1,000,000

Solution:

T(n) = cn log n ------------- (i)

Assume n = N;

T(N) = cN log N

Solving it for c, thus

c = -------------------- (ii)

Putting (ii) in (i)

T(n, N) = n log n

T(n, N) = 1,000,000 log (1,000,000)

T(n, N) = 2000 ms

2000 ms is used to sort 1,000,000 data items.

1. **Answer:**

Given:

T(N) = 1 ms

N = 100

n = 5,000

Solution:

T(n) = cn2 ------------- (i)

Assume n = N;

T(N) = cN2

Solving it for c, thus

c = -------------------- (ii)

Putting (ii) in (i)

T(n, N) = n2

T(n, N) = 50002

T(n, N) = 2500 ms

2500 ms is used to sort 5,000 data items.

1. **Answer:**

Given:

f(n) = n

f(n) = n3

N = 1000

T(N) = 10 s

n = 100,000

Solution:

For f(n) = n

T(n) = cn ------------- (i)

Assume n = N;

T(N) = cN

Solving it for c, thus

c = -------------------- (ii)

Putting (ii) in (i)

T(n, N) = n

T(n, N) = 100,000

T(n, N) = 1000 s

1000 s is used to sort 100,000 data items for f(n) = n.

For f(n) = n3

T(n) = cn3 ------------- (i)

Assume n = N;

T(N) = cN3

Solving it for c, thus

c = -------------------- (ii)

Putting (ii) in (i)

T(n, N) = n3

T(n, N) = 100,0003

T(n, N) = 10000000 s

10000000 s is used to sort 100,000 data items for f(n) = n3.

1. **Answer:**

Given:

TA(n) = 0.1n2 log10 n µs

TB(n) = 2.5n2 µs

Solution:

Algorithm B is better in the sense of Big-Oh.

Algorithm A is outperformed by algorithm B if

TB(n) < TA(n)

2.5n2 < 0.1n2 log10 n

< log10 n

25 < log10 n

Using logarithm rule,

n > 1025

If n < 109, I will recommend to use algorithm A.

1. **Answer:**

Given:

TA(n) = cA n log2 n µs ---------------------> (a)

TB(n) = cB n2 µs ----------------------------> (b)

n = 220

Algorithm A time = 10 µs for 1024 items

Algorithm B time = 1 µs for 1024 items

Solution:

cA =

cB =

cA =

cA = --------------------------> (i)

cB =

cB = ----------------------> (ii)

Putting (i) in (a) & (ii) in (b)

TA (220) = 220 log2 220

TA (220) = 20480 µs

TB (220) = (220)2

TB (220) = 1048576 µs

TB(220) > TA(220), Algorithm A is best for processing n = 220

1. **Answer:**

Given:

TA(n) = 5n log10 n µs

TB(n) = 25n

Solution:

Algorithm B is better in the sense of Big-Oh.

Algorithm A is outperformed by algorithm B if

TB(n) < TA(n)

25n< 5n log10 n

< log10 n

5 < log10 n

Using logarithm rule,

n > 105

1. **Answer:**

Given:

TA(n) = 0.1n log2 n µs

TB(n) = 5n

Solution:

Package B is better in the sense of Big-Oh.

Package A is outperformed by package B if

TB(n) < TA(n)

5n< 0.1n log2 n

< log2 n

50 < log10 n

Using logarithm rule,

n > 250 which is approximately 1015

Package A should be chosen to process 1012 databases.

1. **Answer:**

Given:

TA(n) = 0.001n ms

TB(n) = 500√n ms

Solution:

Package B is better in the sense of Big-Oh.

Package A is outperformed by package B if

TB(n) < TA(n)

500√n< 0.001n

<

500000 < √n

Taking sq on both sides

n > 25 × 1010

Package A should be chosen to process 109 records.

1. **Answer:**

Given:

TA(n) = cA n log10 n --------------> (a)

TB(n) = cB n -----------------------> (b)

n = 104

TA(n) = 100ms

TB(n) = 500 ms

Solution:

From (a) we can have,

cA =

cA =

cA = -------------> (i)

From (b) we can have,

cB =

cB =

cB = ----------------> (ii)

Putting (i) in (a) and (ii) in (b)

TA(n) = n log10 n

TB(n) = n

Package A is outperformed by package B if

TB(n) < TA(n)

n< n log10 n

< log10 n

20 < log10 n

Using logarithm rule,

n > 1020

Package A should be chosen to process 109 items.



|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Complexity** | **Time required to process 10, 000 items** |
| A | O(n2) | T(10, 000) = = 100,000 s |
| B | O(n1.5) | T(10, 000) = = 10,000 s |
| C | O(nlog n) | T(10, 000) = = 2,000 s |

1. **Answer:**

Given:

T­EP = 3 n 1.5

T­WP = 0.03 n 1.75

Solution:

Package A is better in the sense of Big-Oh.

Package EP is outperformed by Package WP if

TEP(n) < TWP(n)

3n1.5 < 0.03 n 1.75

<

100 < n0.25

Squaring on both sides

n0.5 > 1002

Taking square again

n > 108

To process data items up to 108, package WP should be used.

**Question 5)**

2. Suppose n = 3, how many times are the five assignments executed?

**Answer:**

27 times five assignments are executed, 3 times outer most loop will execute, 9 times 2nd loop and 27 times inner most loop is executed.

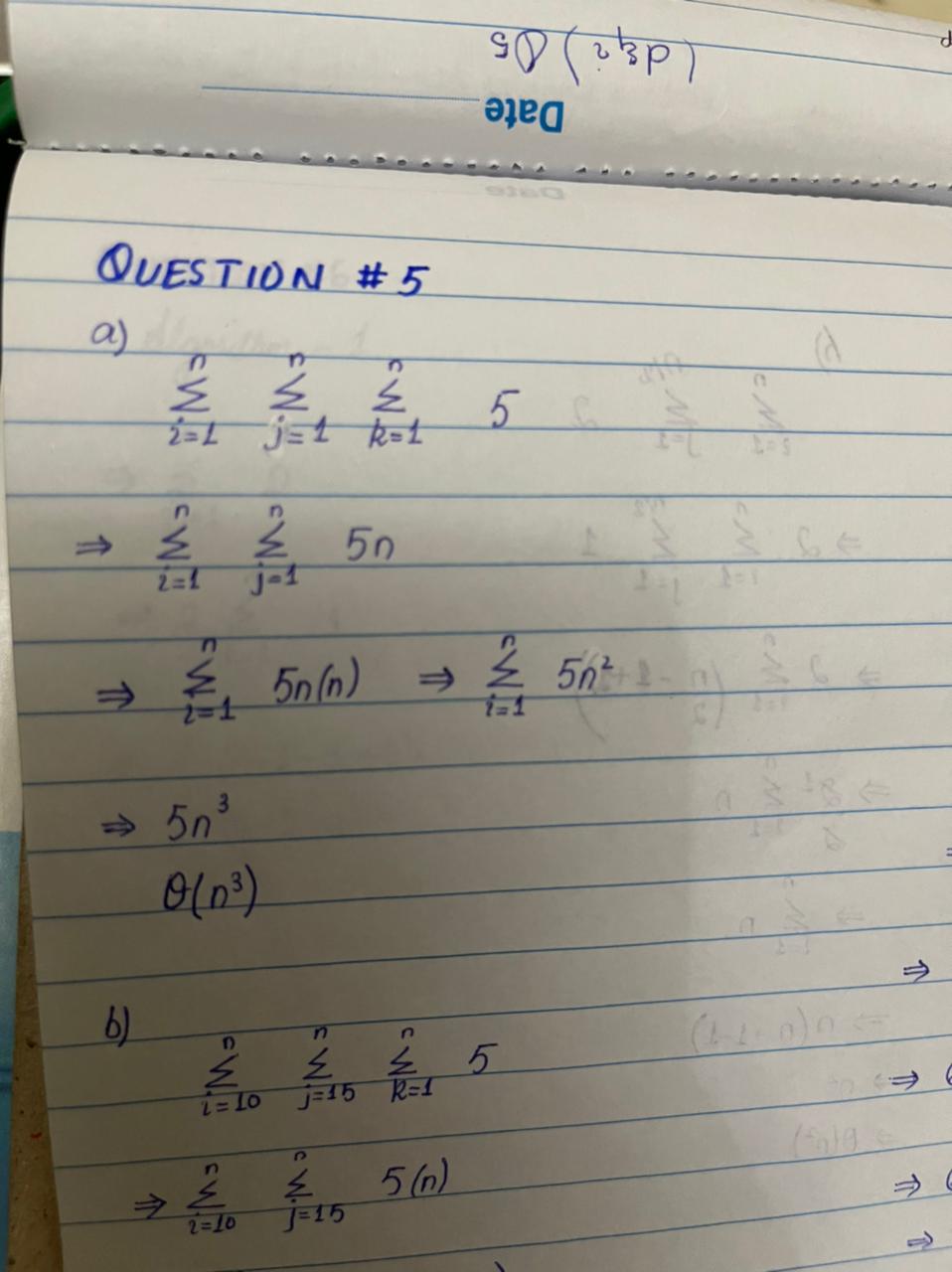
1. Does one loop depend on another?

**Answer:**

No, the loops are independent. Loops are depending on n.

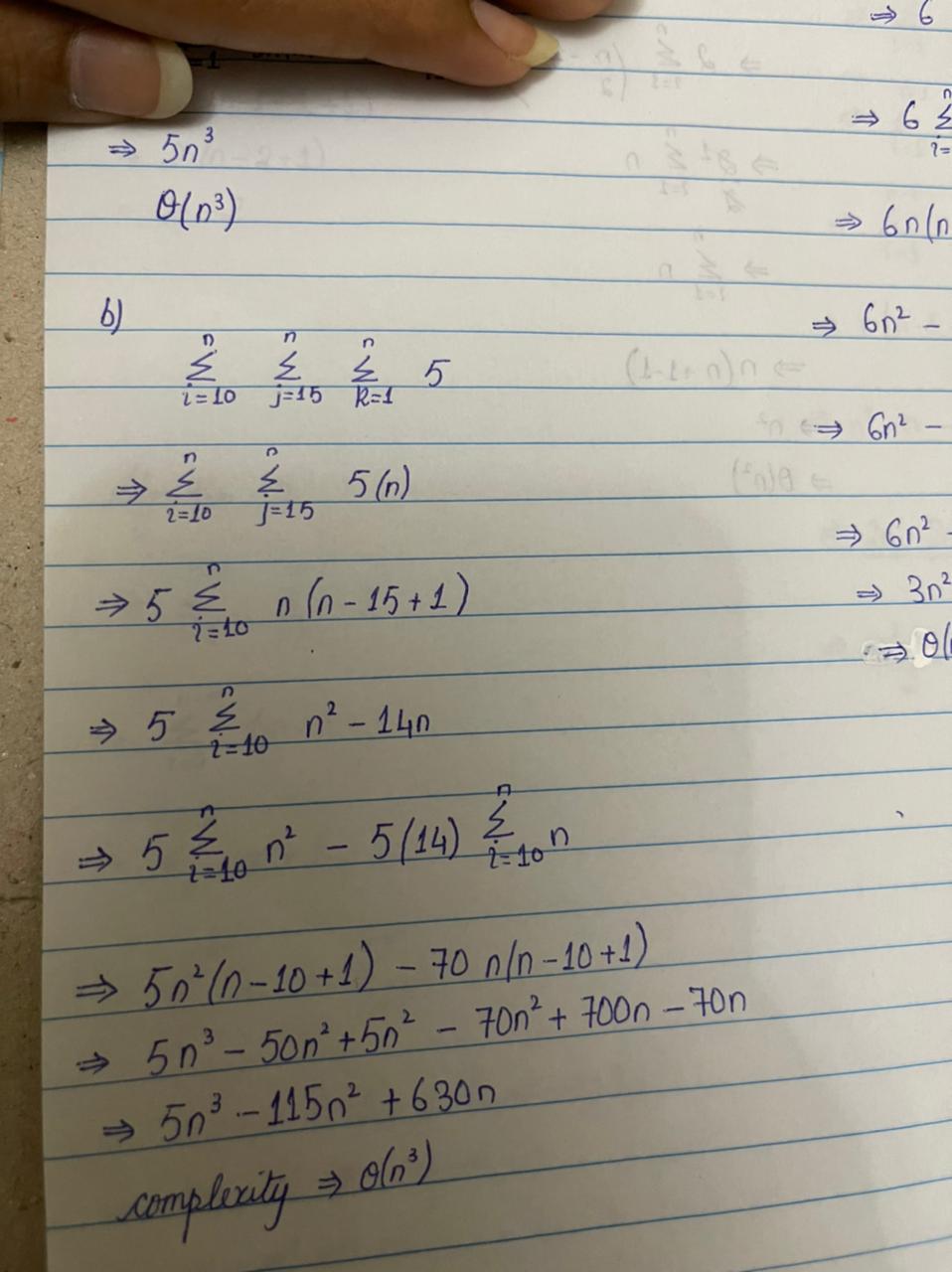
1. What is the complexity of the segment?

**Answer:**



2. What is the complexity of the segment?

**Answer:**

****



Notice the 20 assignments are in a doubly nested loop and the five instructions are in a triple loop. Suppose n = 10.

1. How many times will the 20 assignments be executed?

**Answer:**

100 times 20 assignments will be executed.

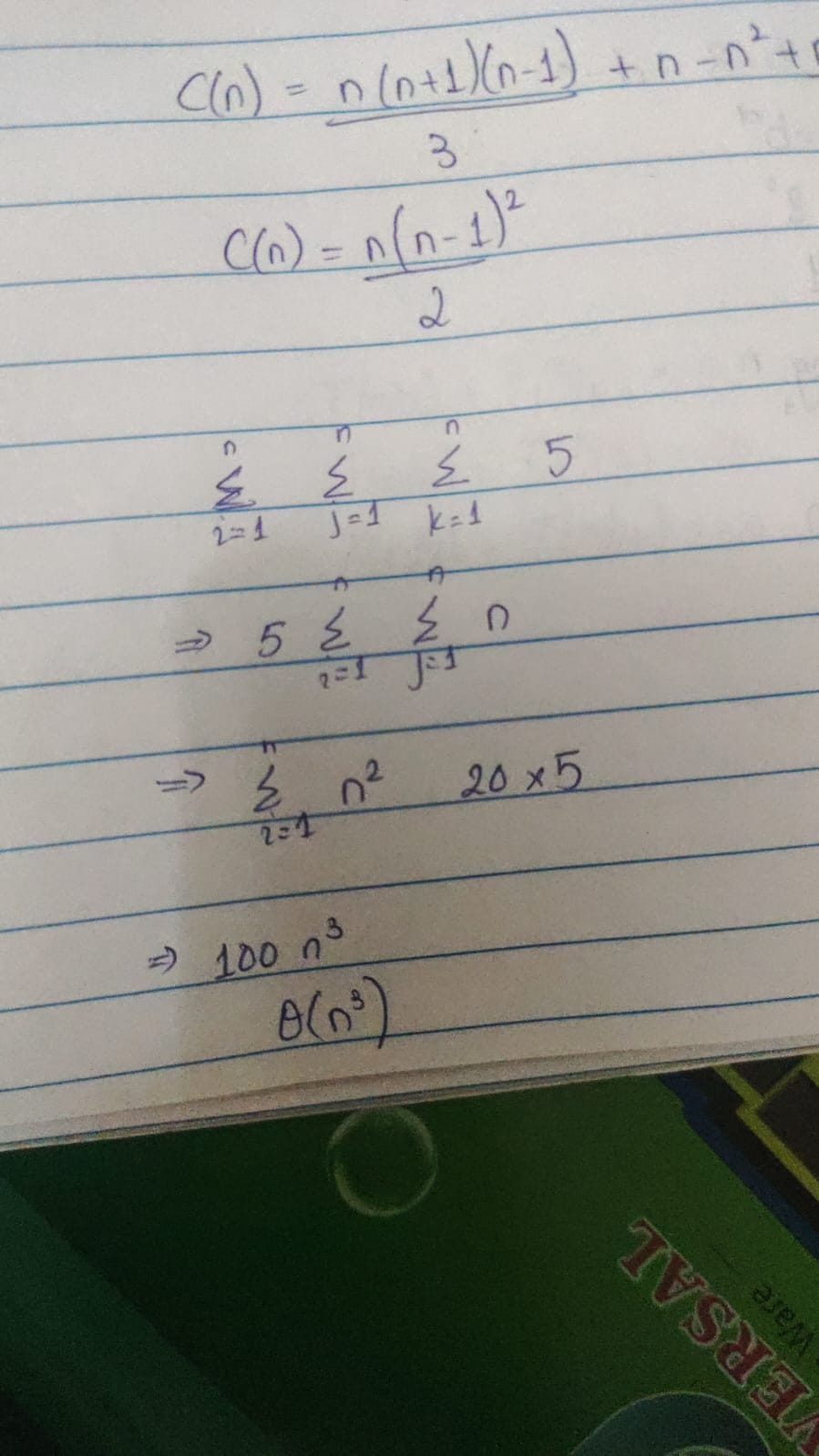
1. How many times will the five instructions be executed?

**Answer:**

500 times five assignments will be executed.

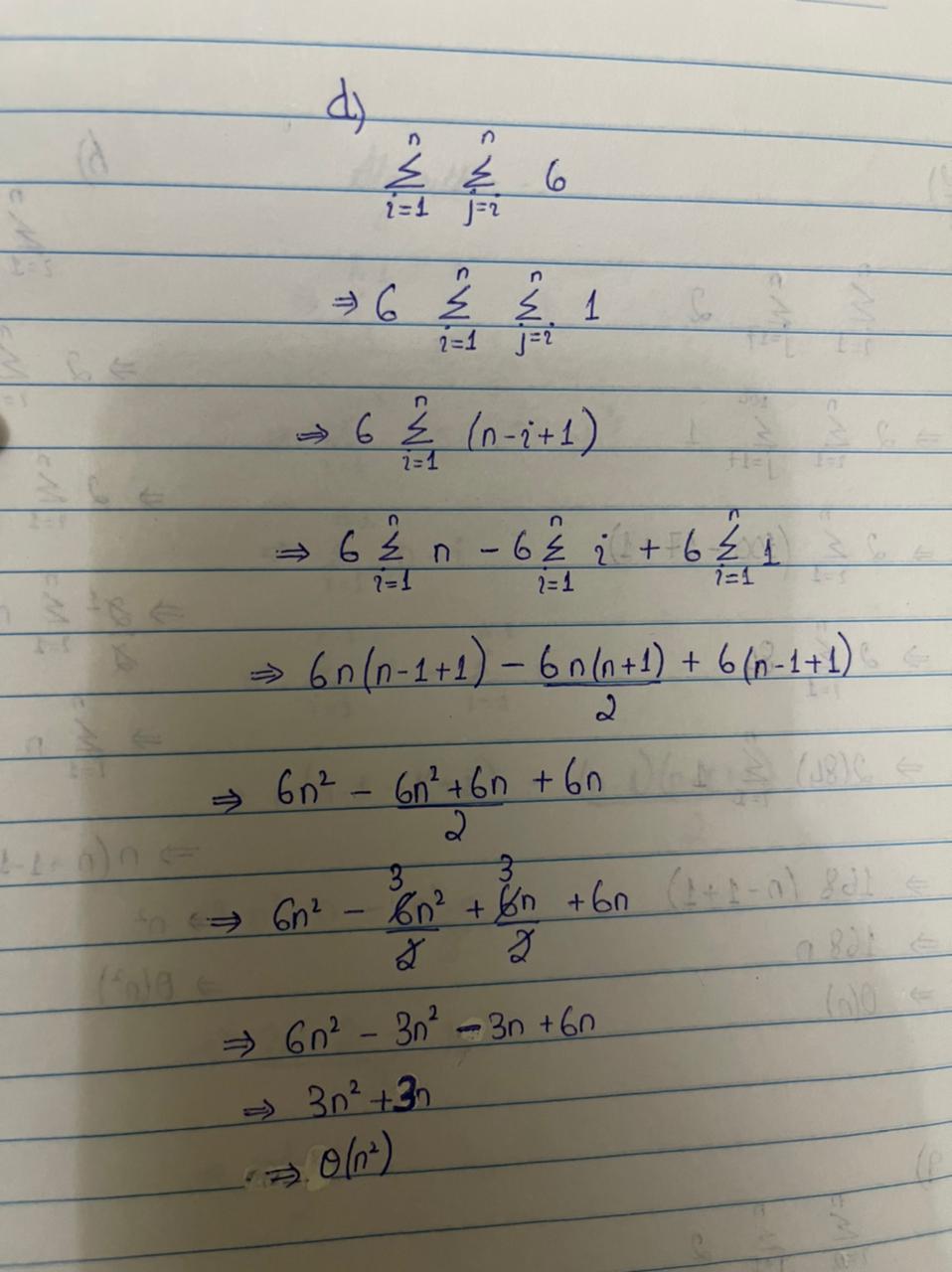
1. Now determine the complexity of the segment

**Answer:**

****

2. Determine the complexity.

**Answer:**

****

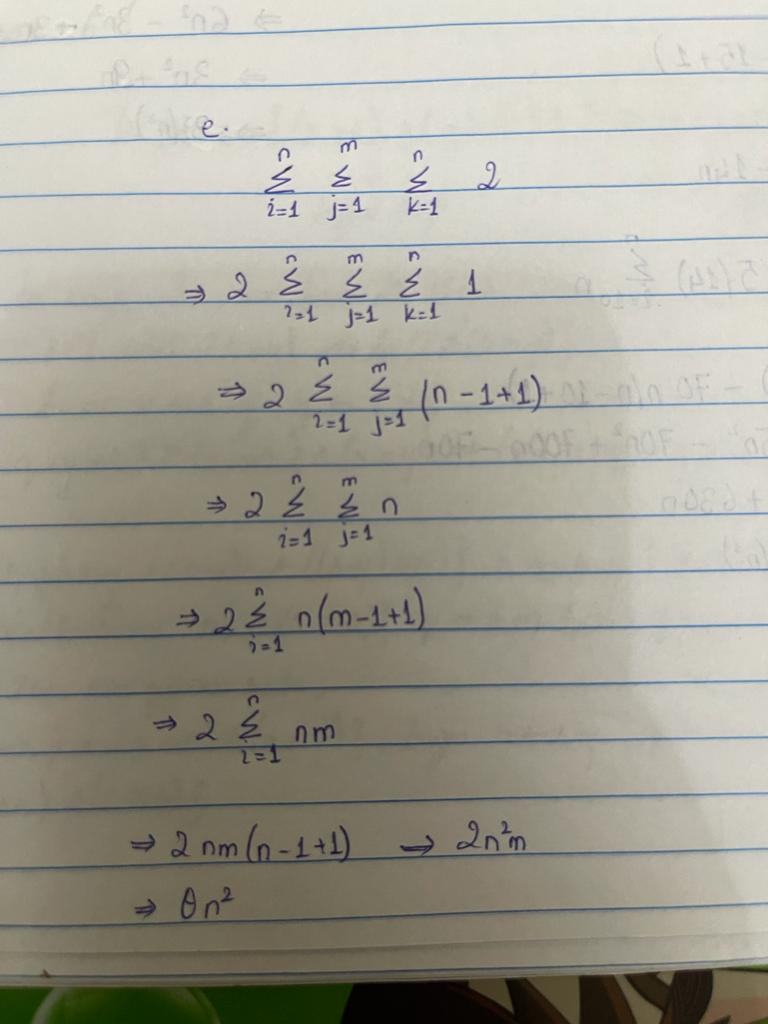
2. If n = 3 and m = 4 how many times are the two assignment statements executed?

**Answer:**

36 times two assignments will be executed.

1. Determine the complexity.

**Answer:**

****

2. How many times are the three assignments executed?

**Answer:**

n times three assignment will be executed.

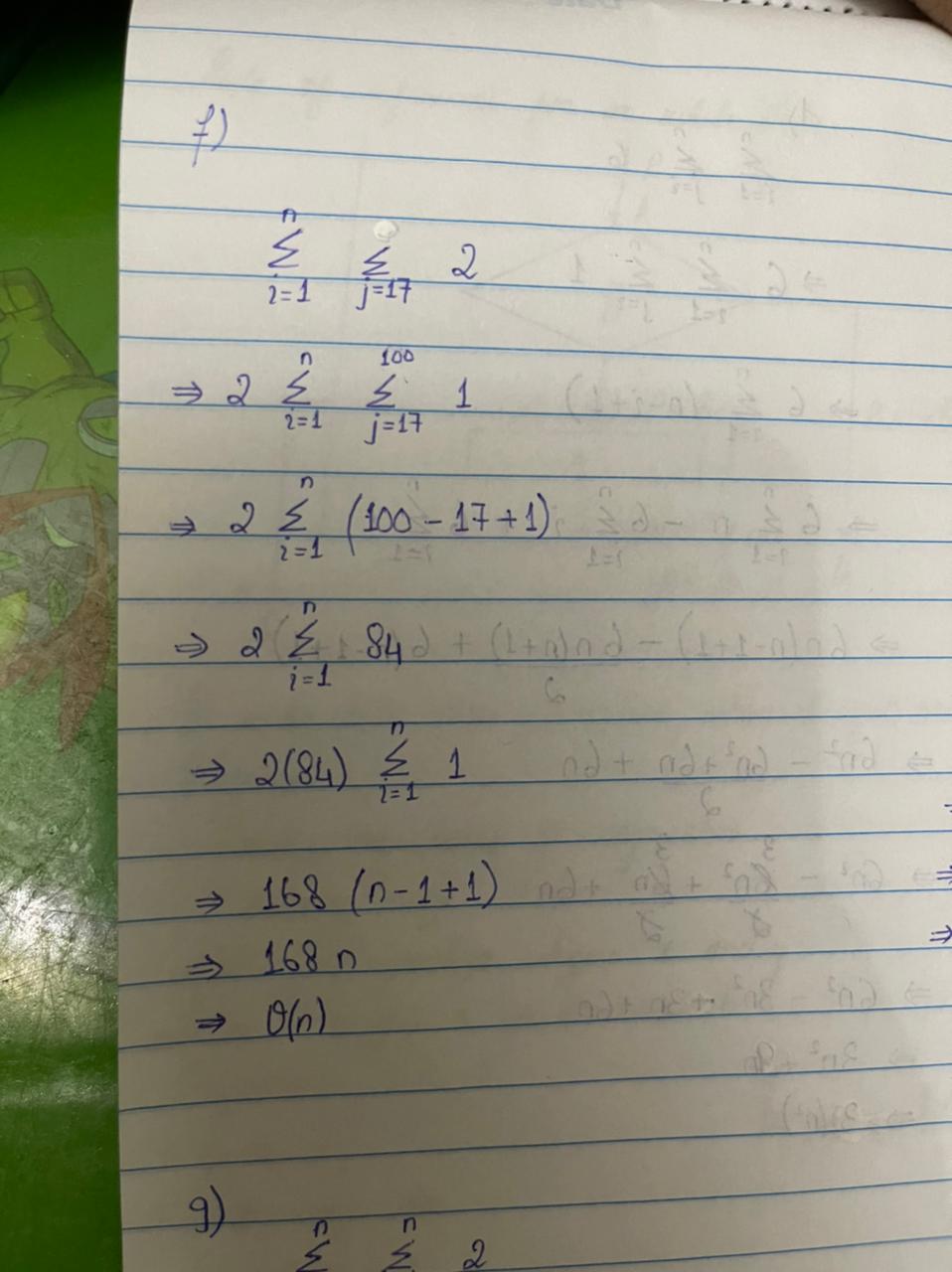
1. How many times are the two assignments executed?

**Answer:**

83n times two assignments will be executed

1. What is the order (complexity) of the loop?

**Answer:**

****

2. How many times are the three assignments executed?

**Answer:**

n times three assignments will be executed.

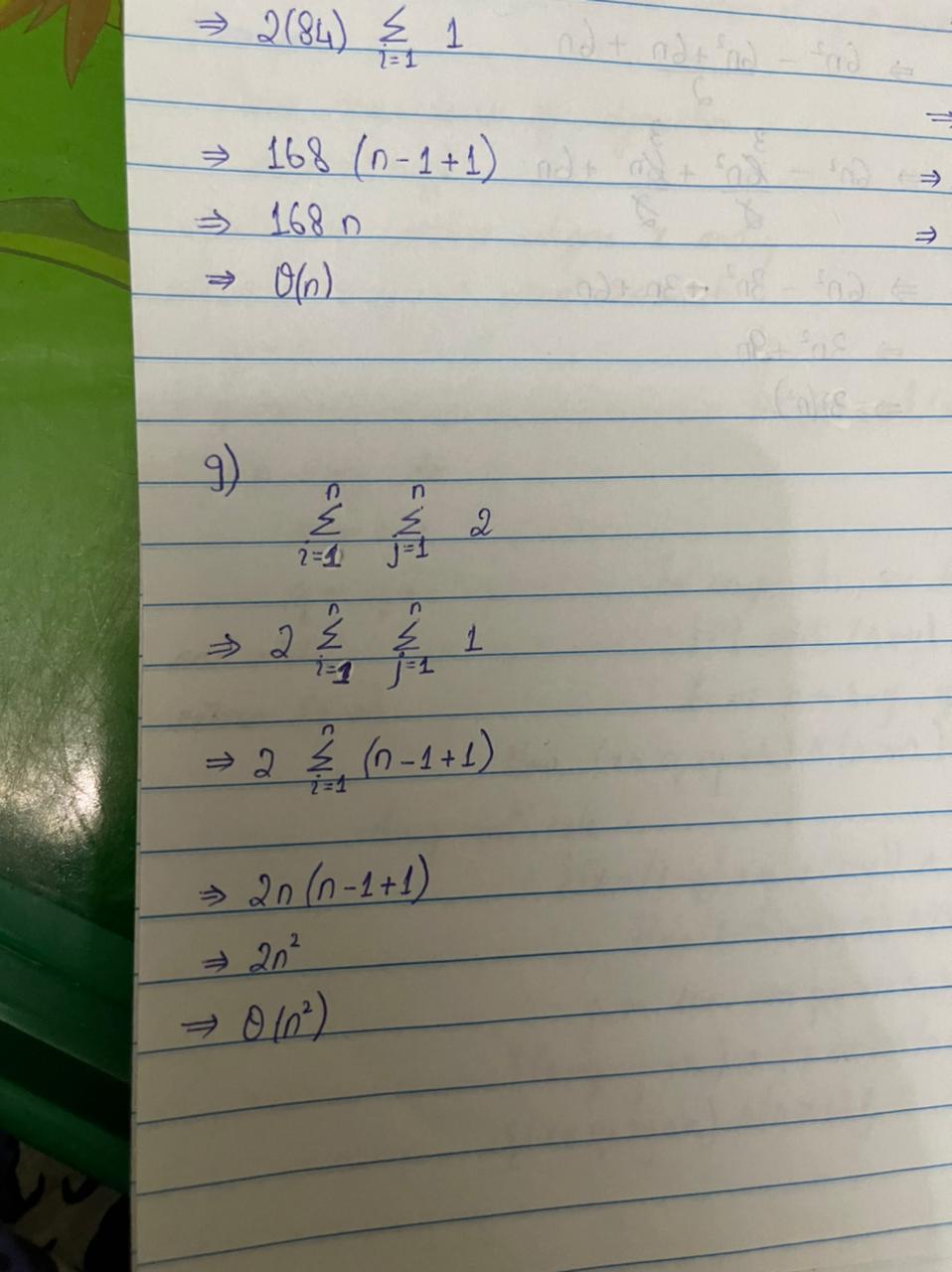
1. How many times are the two assignments executed?

**Answer:**

n\*n times the two assignments will be executed.

1. What is the order (complexity) of the loop?

**Answer:**

****



Let n = 8.

1. How many times are the three assignments executed?

**Answer:**

8 times three assignments will be executed.

1. How many time will the inner loop execute?

**Answer:**

32 times inner loop will be executed.

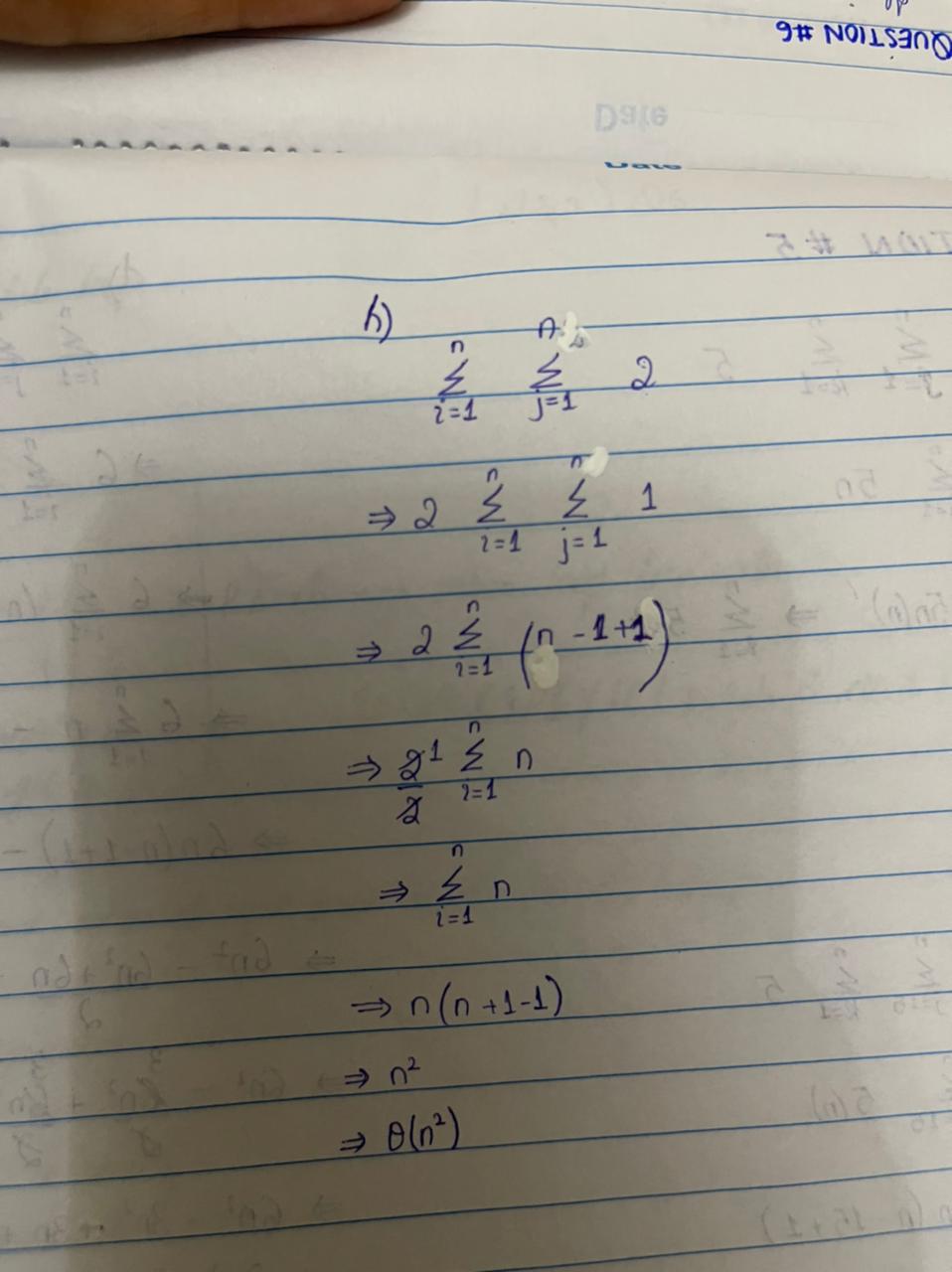
1. What will the values of j be for each execution of the inner loop?

**Answer:**

**1, 2, 4 and 8.**

1. Now determine the complexity for arbitrary n?

**Answer:**

****

2. Determine the complexity for arbitrary n?

**Answer:**

Θ (1).

**Question 6)**

1. **Algo 1**
2. What is its basic operation/Primitive Operation?

**Answer:**

* If A[i] < minval
* If A[i] > maxval

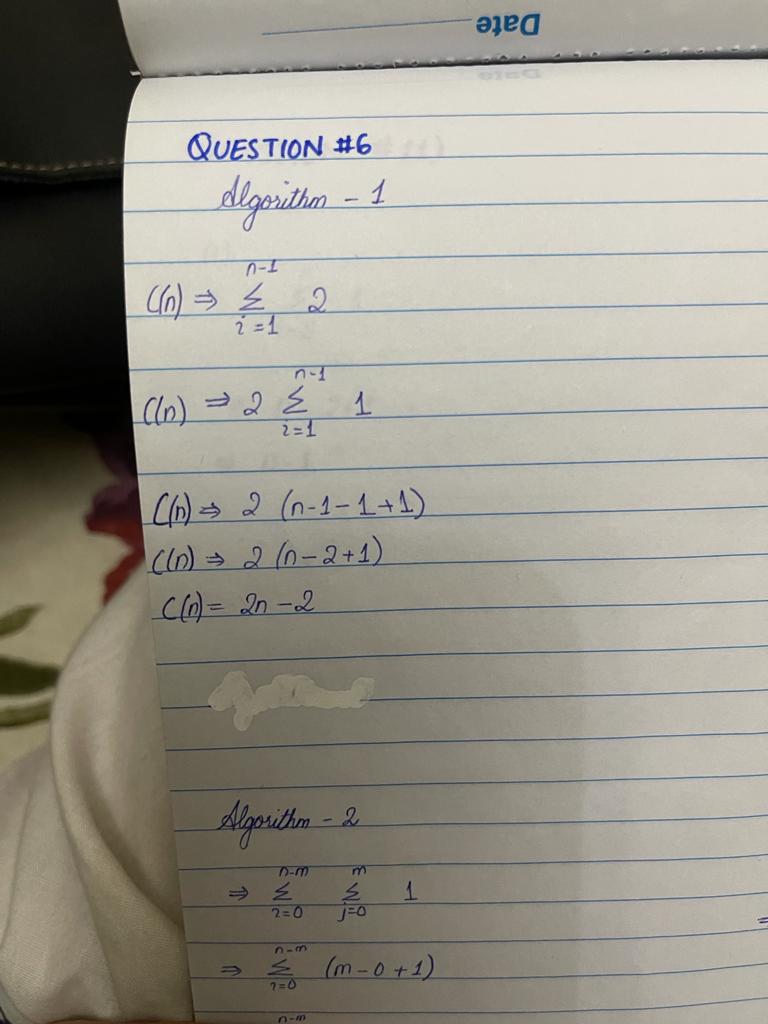
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

No, the count for basic operation remains same the loop will execute n-1 times and for every iteration basic operations are checked. Min and Max cannot be decided until the end of the array.

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**



1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ.

**Answer:**

Θ (n)

1. **Algo 2**
2. What is its basic operation/Primitive Operation?

**Answer:**

* while j < m and P[j] = T[i + j]
* j = j + 1

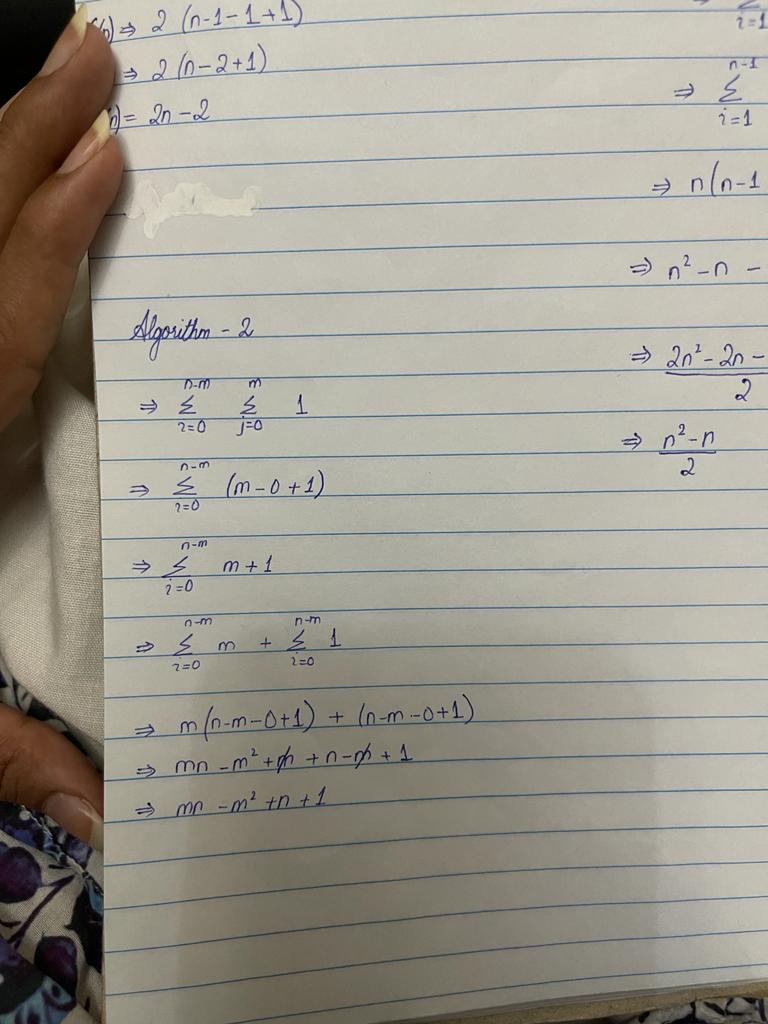
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

Yes, the basic operation count can be different for inputs of the same size. If the pattern is found it will return back. Pattern can be immediately found or maybe it will iterate till the end until pattern found.

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

Best case: Ω (m)

Worst case: O (mn)

1. **Algo 3**
2. What is its basic operation/Primitive Operation?

**Answer:**

* if d < dmin

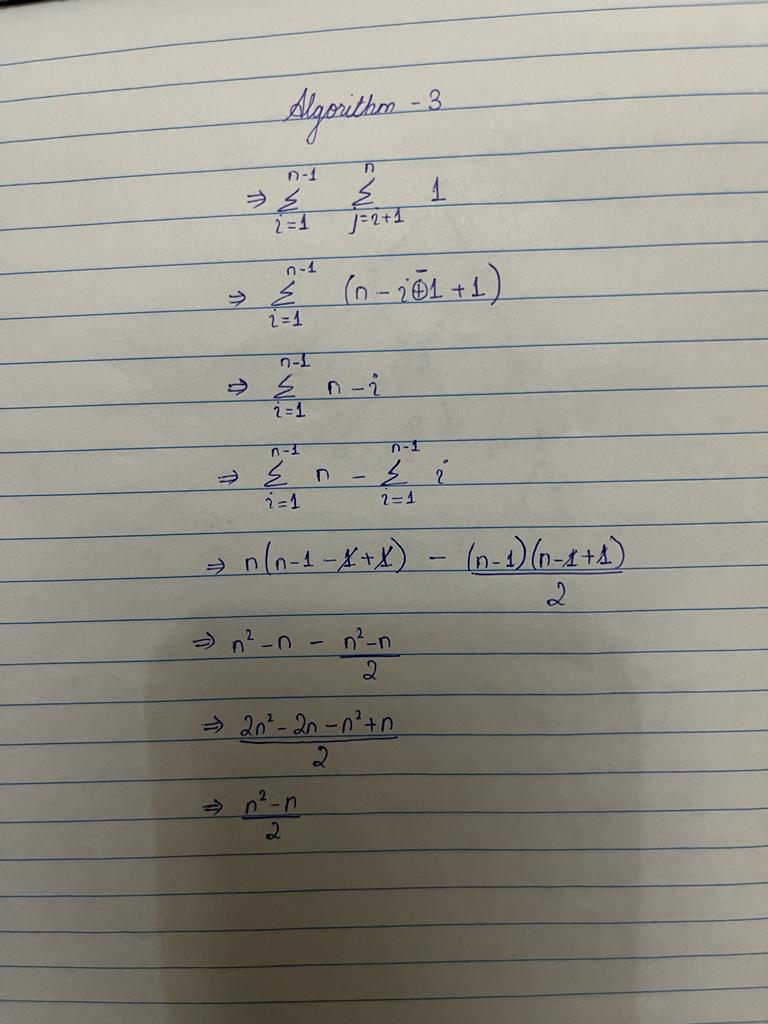
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

No, the basic operation count remains the same for the input of same size.

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

Θ (n2)

2. What is its basic operation/Primitive Operation?

**Answer:**

* I = I / 3

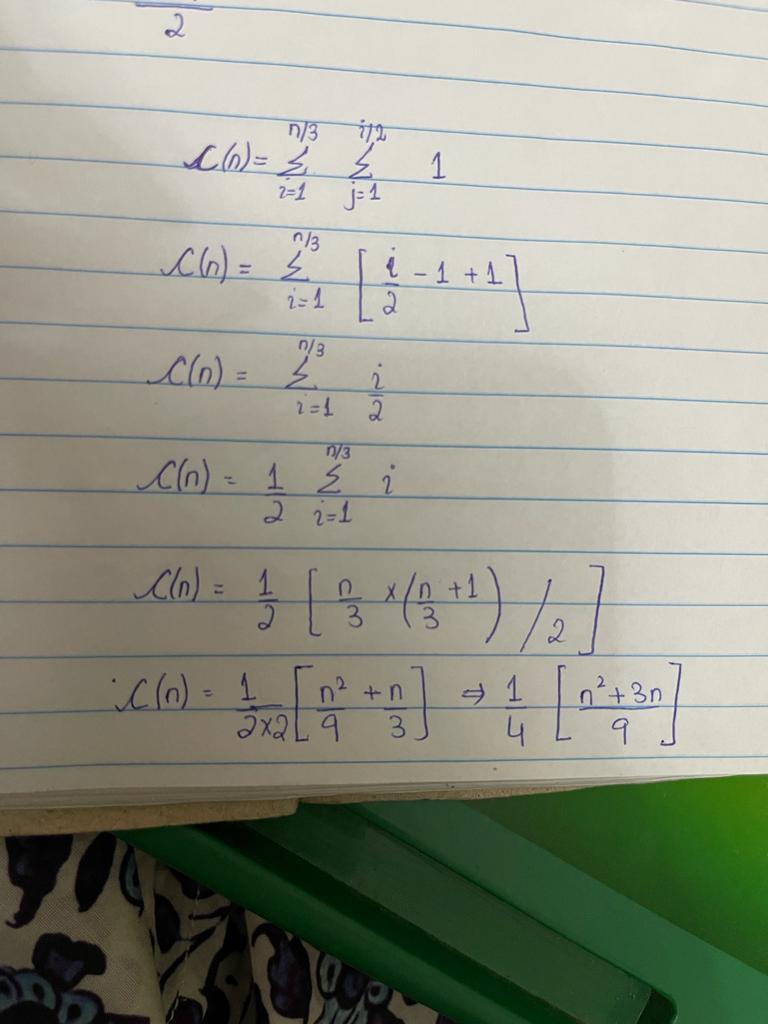
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

No, the basic operation count remains same for the input of same size.

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

Θ (n2)

2. What is its basic operation/Primitive Operation?

**Answer:**

* x = x + 1

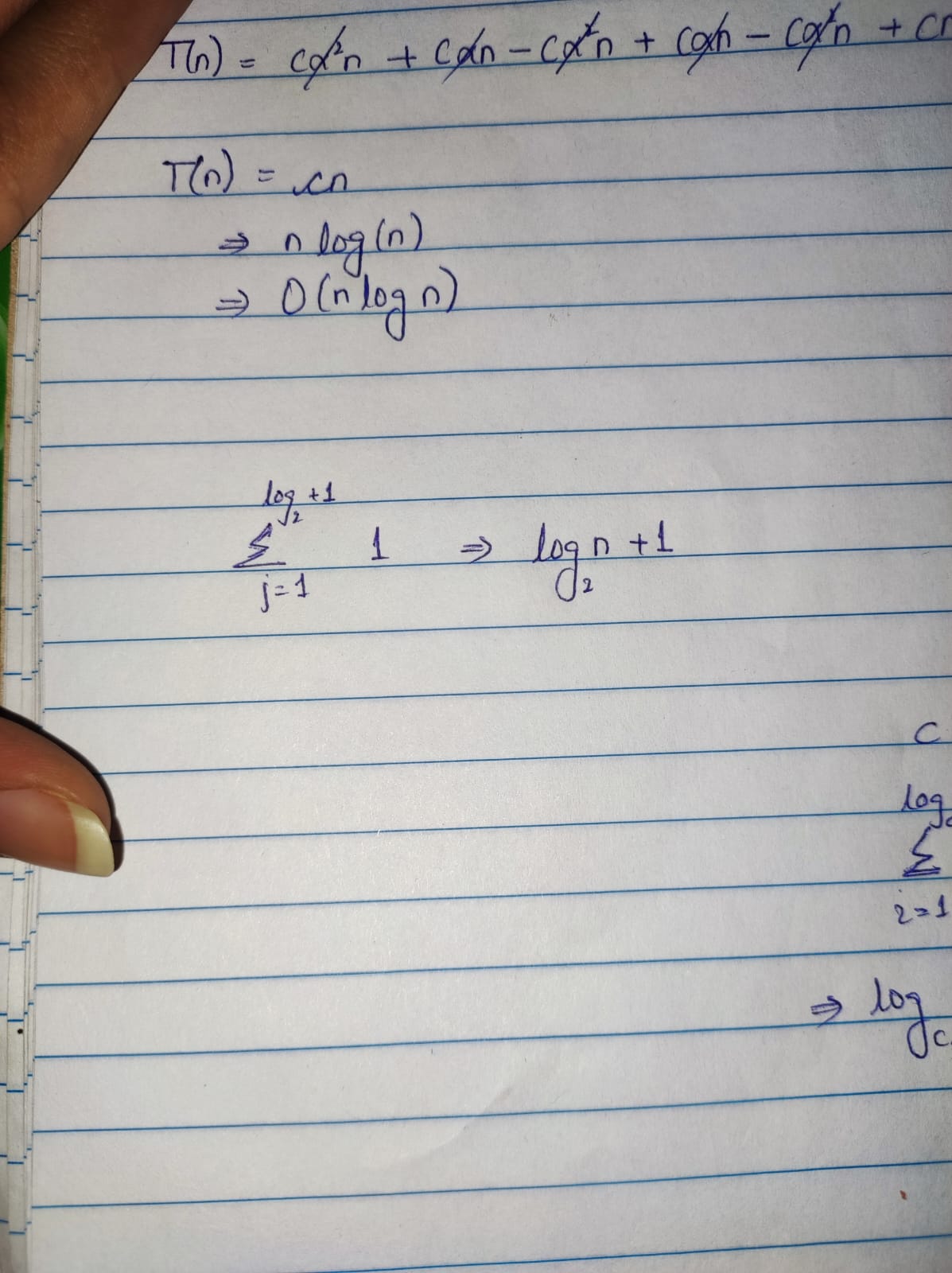
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

No.

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

Θ (log­­2 n)

2. What is its basic operation/Primitive Operation?

**Answer:**

Some O (1) operation.

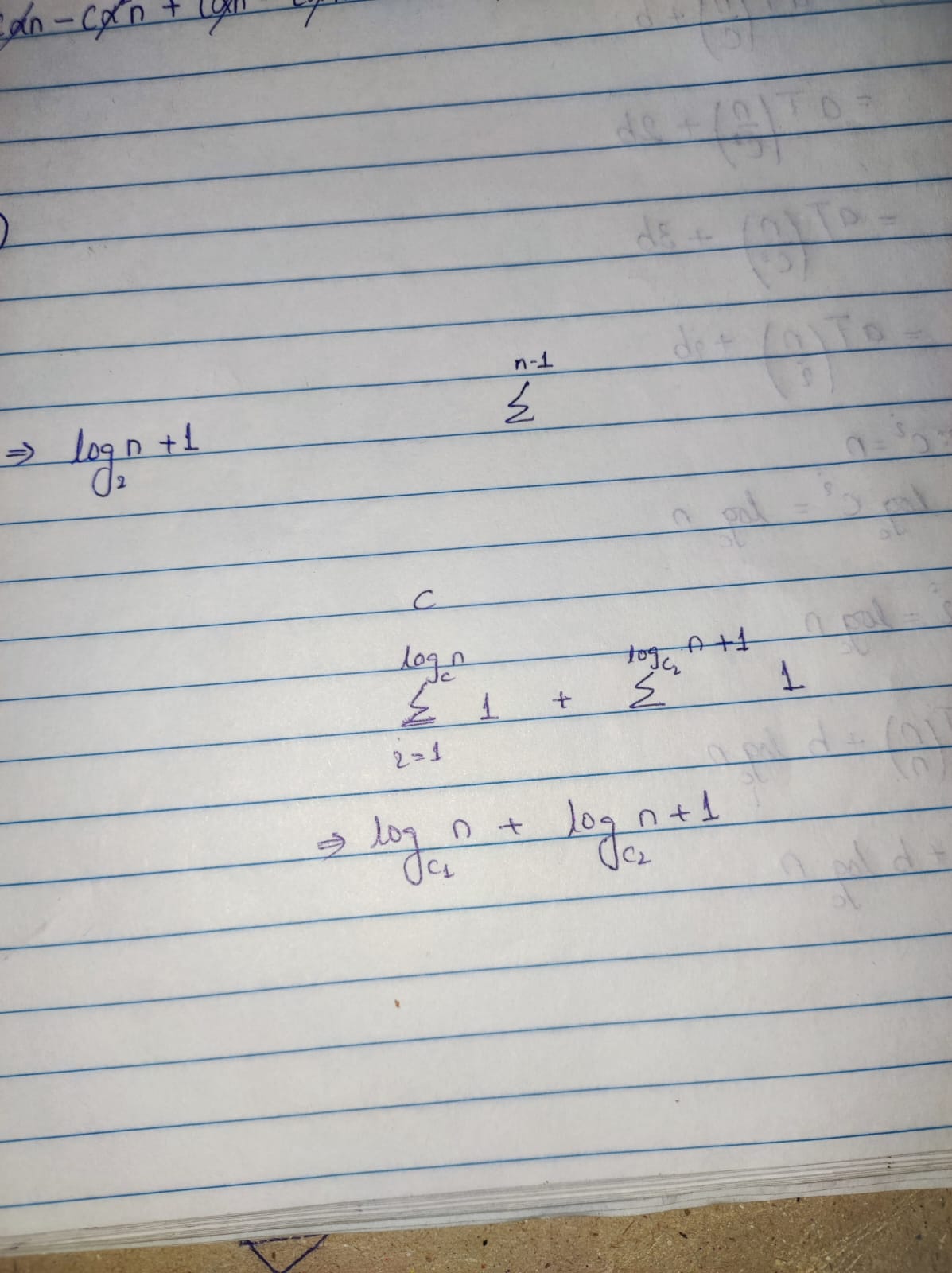
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

**No.**

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

Θ(log n)

2. What is its basic operation/Primitive Operation?

**Answer:**

Mij = min (mij, ak)

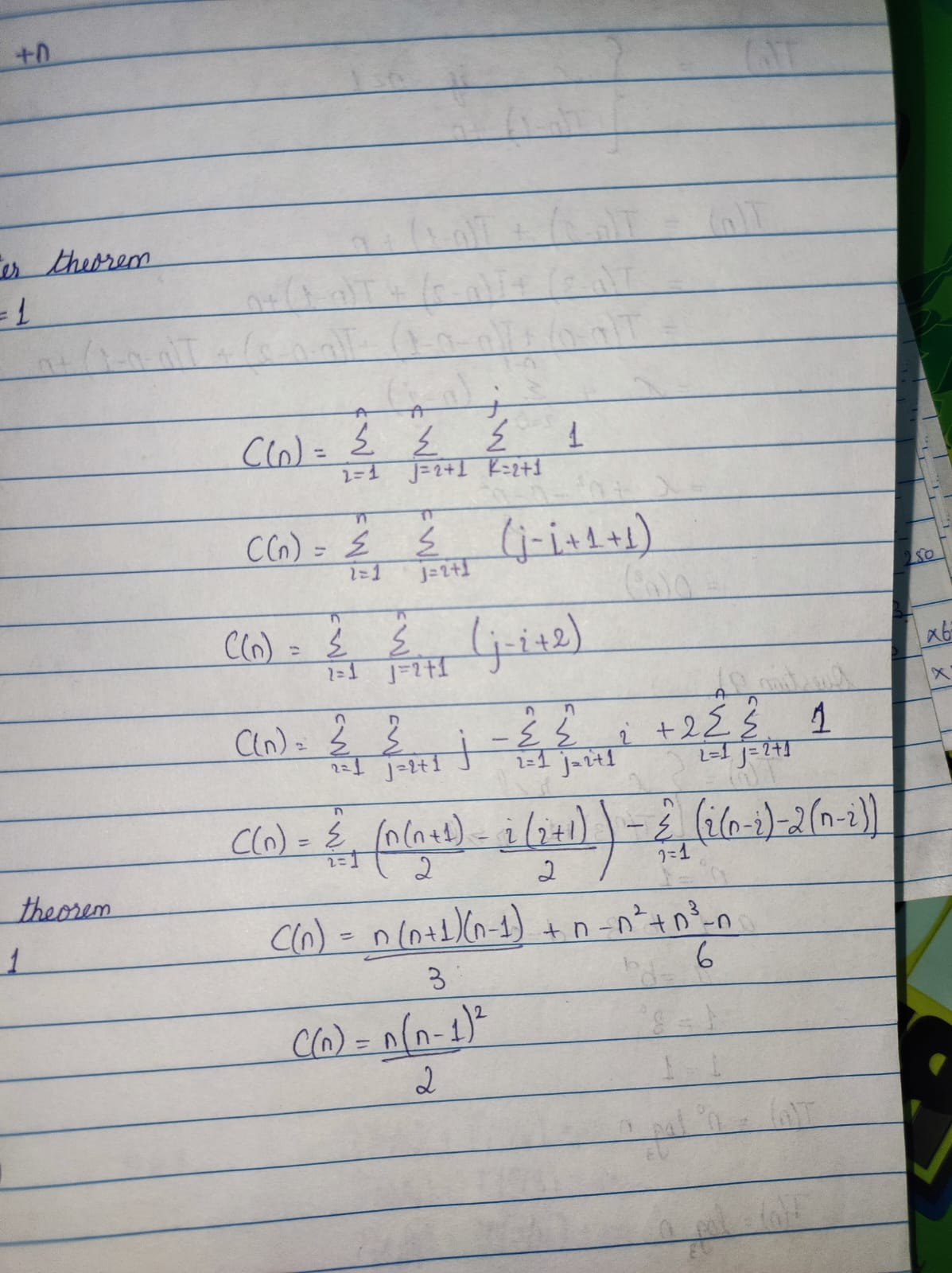
1. Whether the basic operation count can be different for inputs of the same size.

**Answer:**

**No.**

1. Setup the summation C(n) to count how many times is the basic operation executed. Simplify summation using standard formulas.

**Answer:**

****

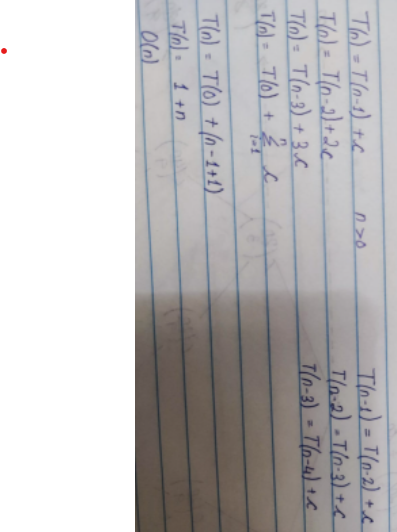
1. Determine Orders of growth of C(n). Use most appropriate notation among O, Ω and Θ

**Answer:**

O(n3)

**Question 7)**

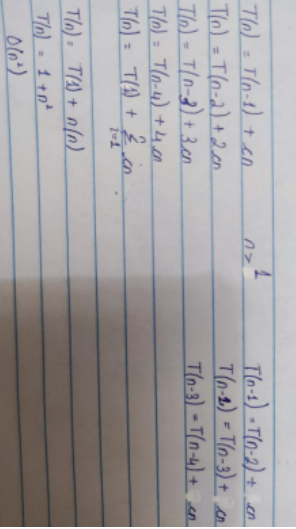
1. **Answer**



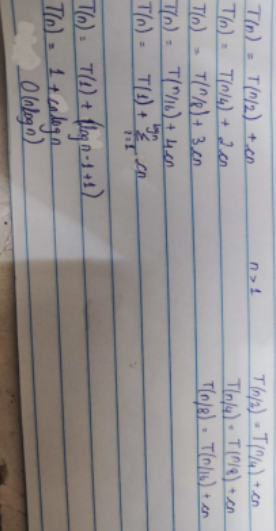
1. **Answer**
2. How many times will the task be executed?

Ans: 16 times the task will be executed.

1. So what is the order of the function?

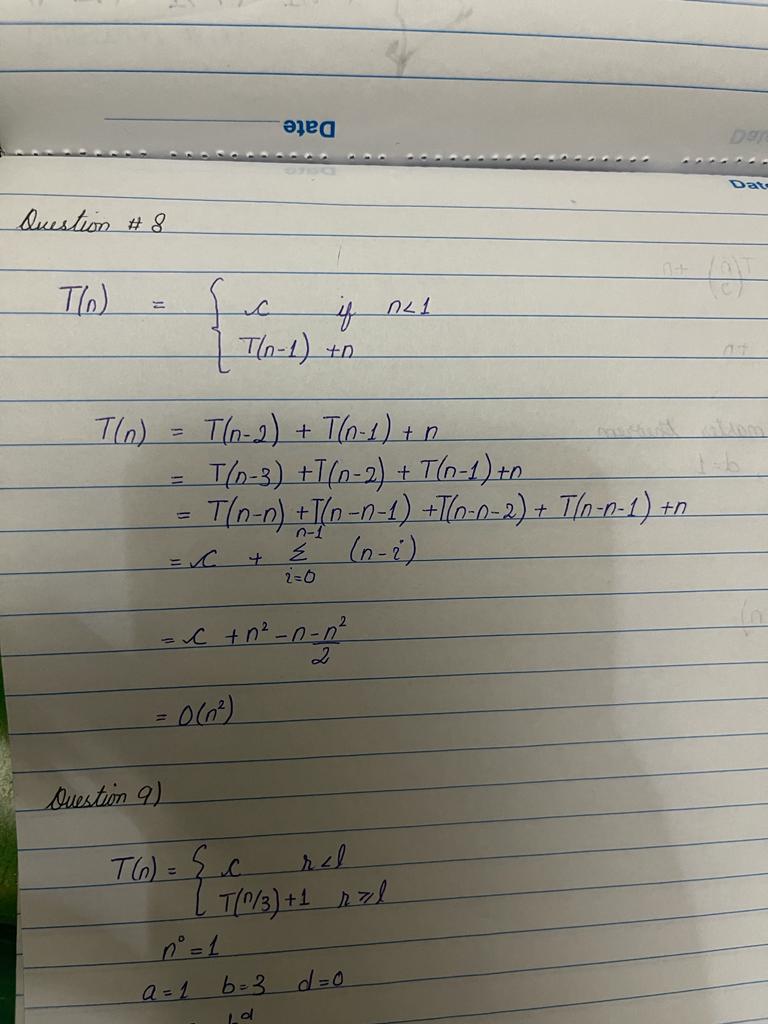
****

1. **Answer**

****

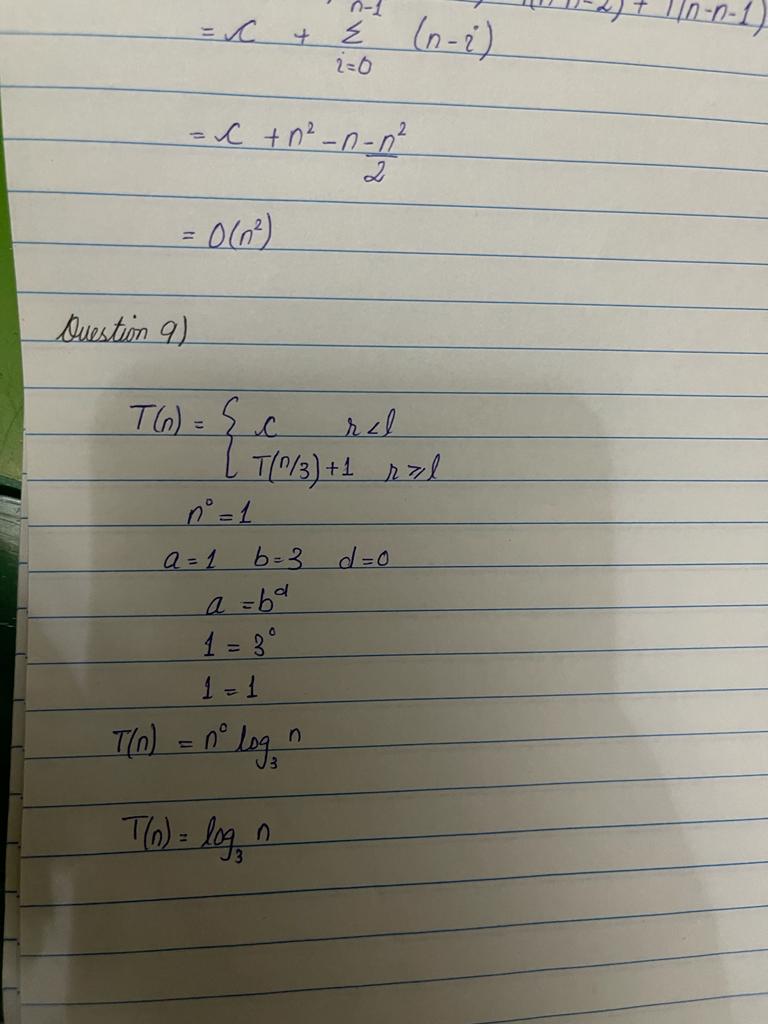
**Question 8)**

**Answer**



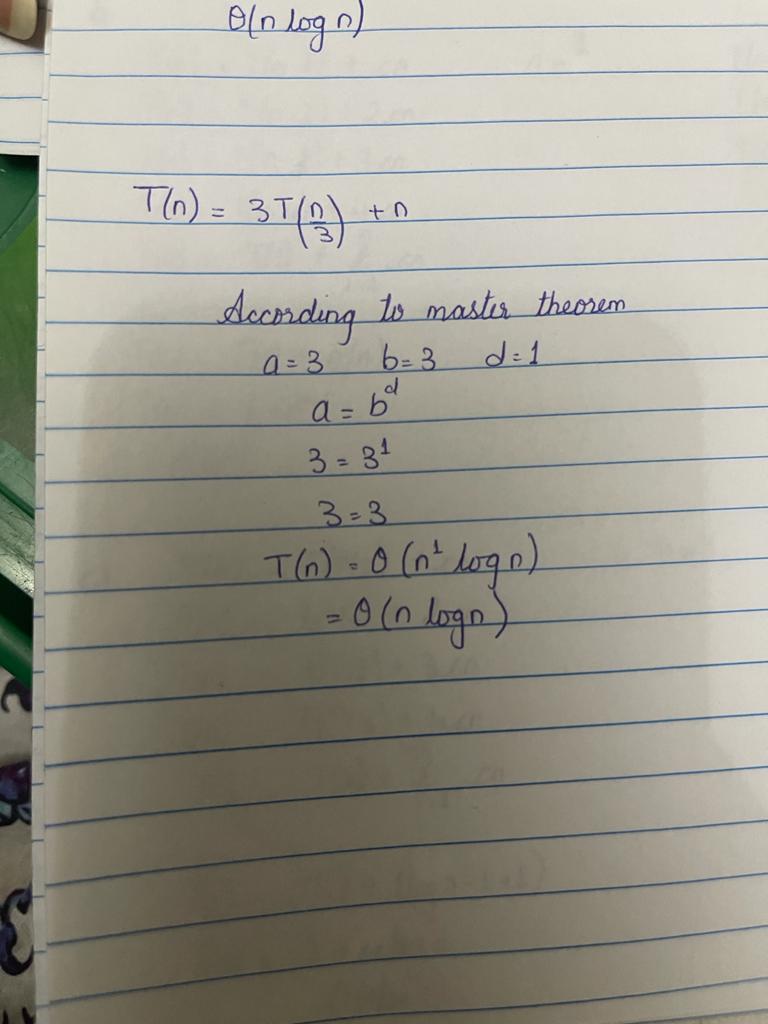
**Question 9)**

**Answer**



**Question 10)**

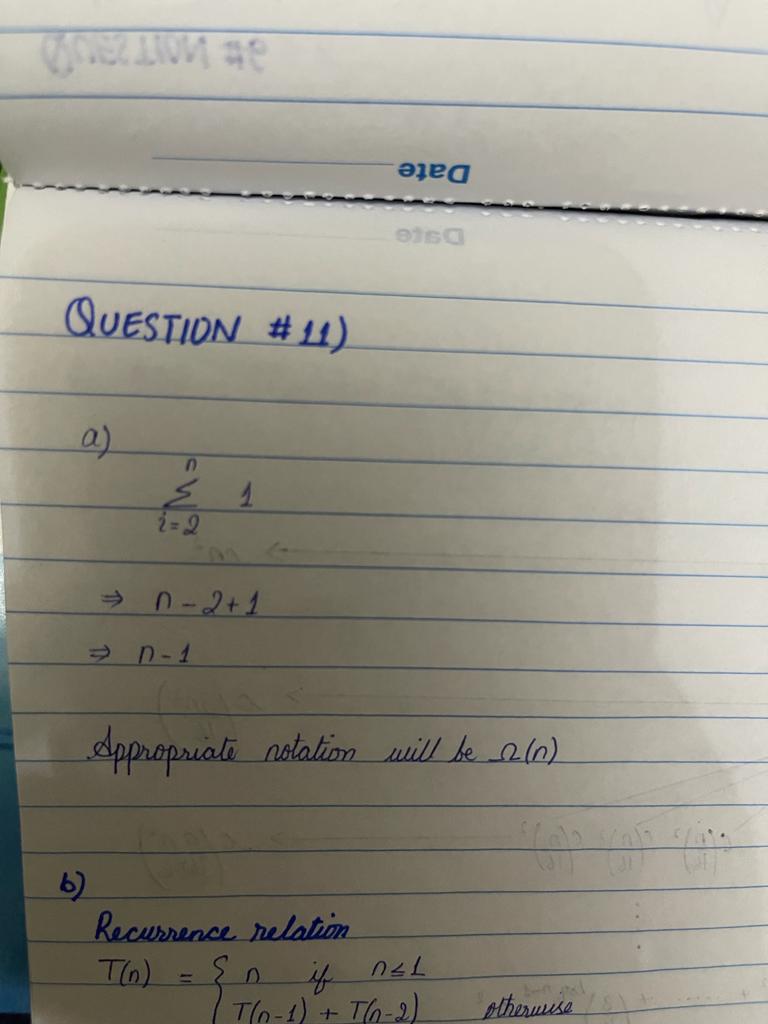
**Answer**



**Question 11)**

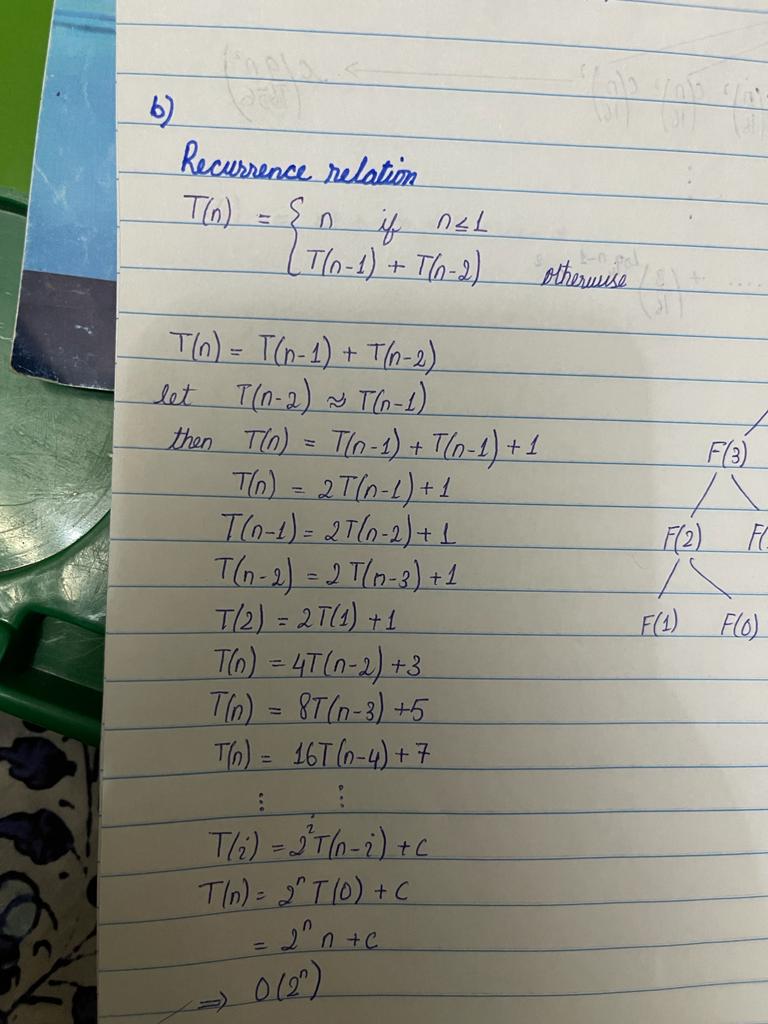
1. Setup the summation for counting the basic operation (i.e. how many times is the basic operation executed)? Use most appropriate notation among O, Ω and Θ to identify time complexity of iterative algorithm.

**Answer**

****

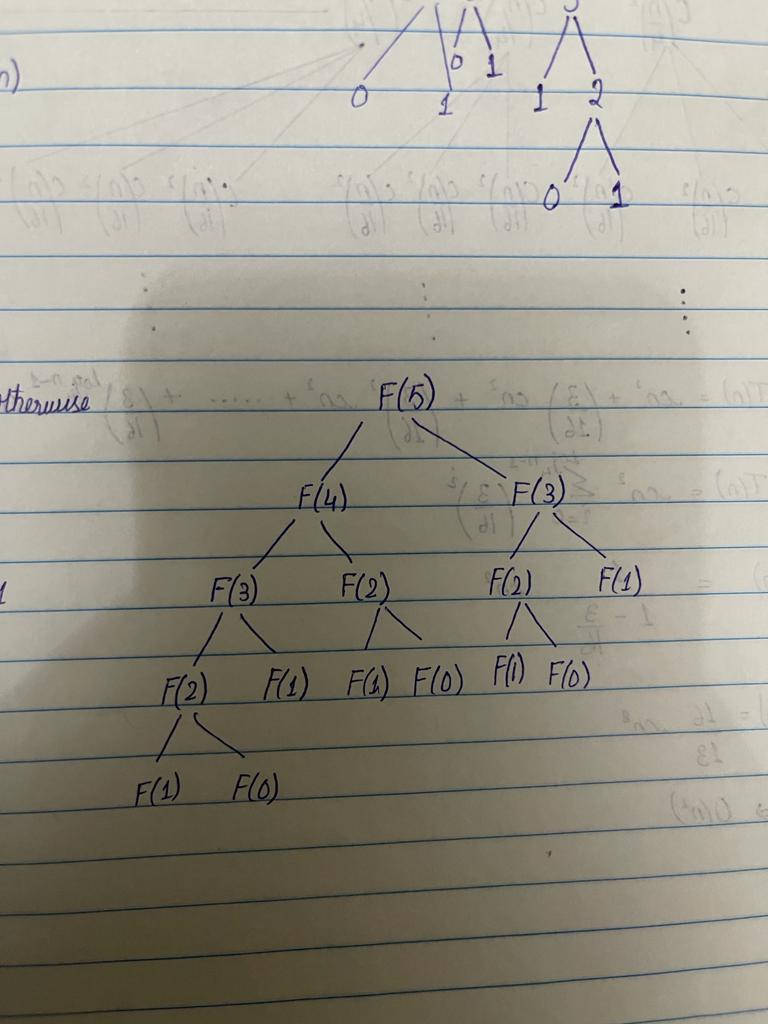
1. Set up a recurrence relation of recursive algorithm and solve it. Use most appropriate notation among O, Ω and Θ to indicate the time efficiency class of recursive algorithm.

**Answer**

****

1. Draw a recursion tree for F(5) where F(5) is 5th Fibonacci term.

**Answer**

****

1. Which of the algorithm iterative or recursive is most efficient. Justify your answer.

**Answer**

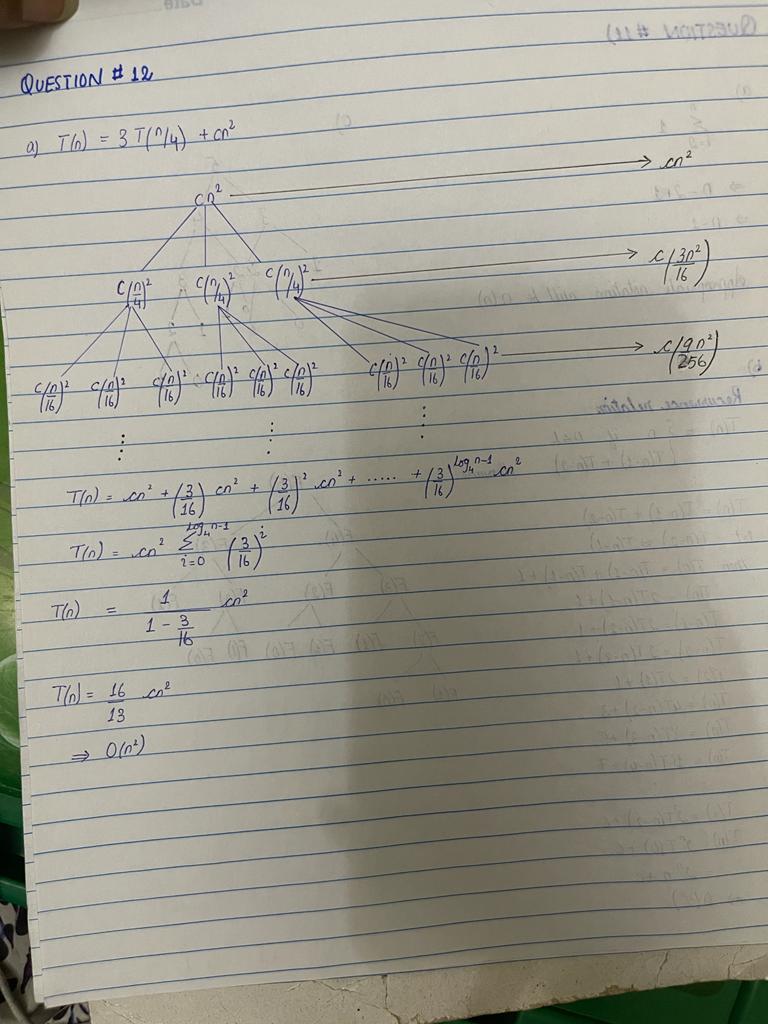
Iterative is more effective than recursion because recursion has time complexity O (2n) while iterative has Ω (n). Recursion grows exponentially while iterative grows linearly.

**Question 12)**

Solve the following recurrence using Recursion Tree Method.

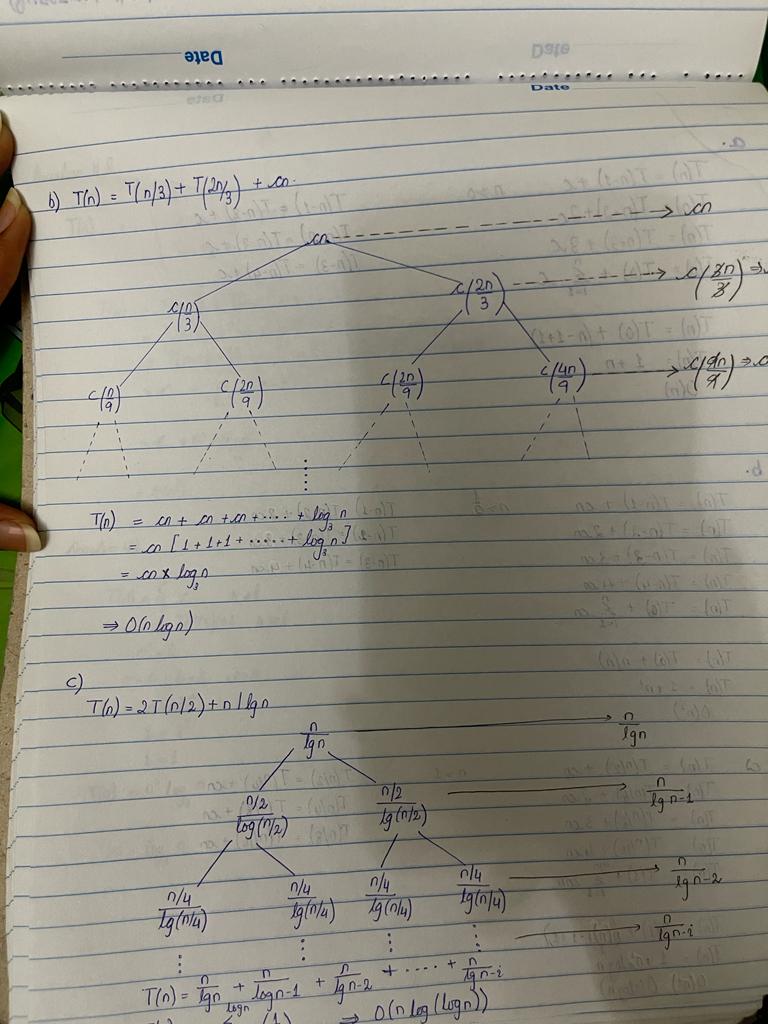
1. T(n) = 3T(n/4) + cn2

**Answer**



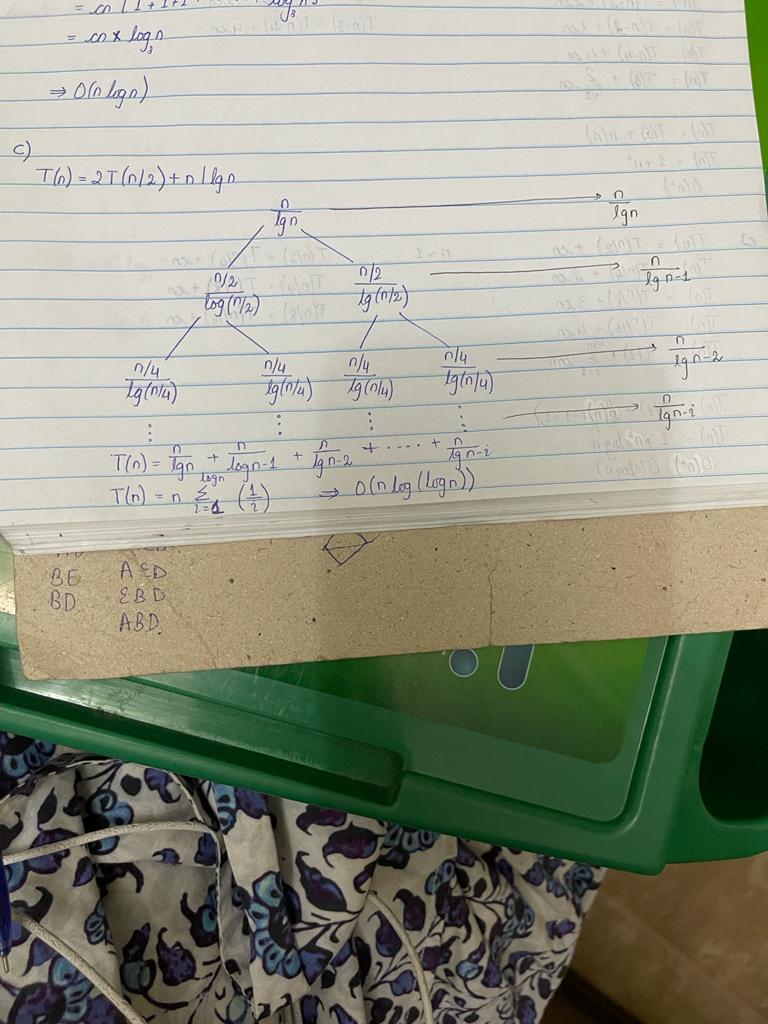
1. T(n) = T (n/3) + T (2n/3) + cn

**Answer**



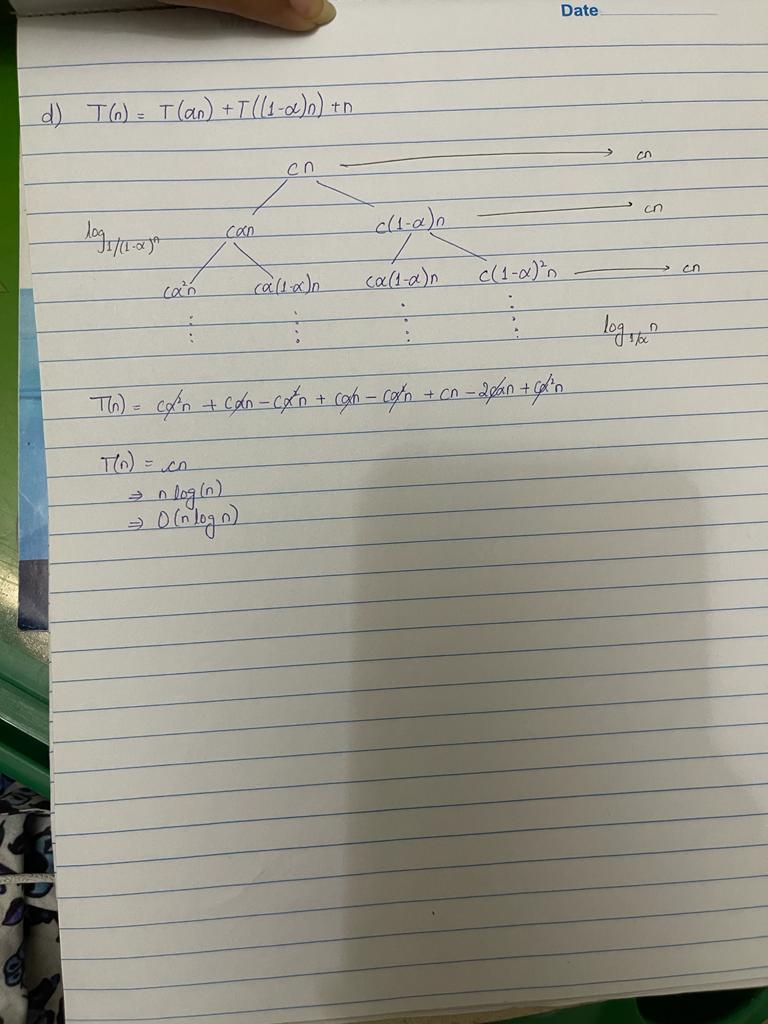
1. T (n) = 2T (n/2) + n/ lg n

**Answer**



1. T (n) = T (αn) + T ((1 − α)n) + n

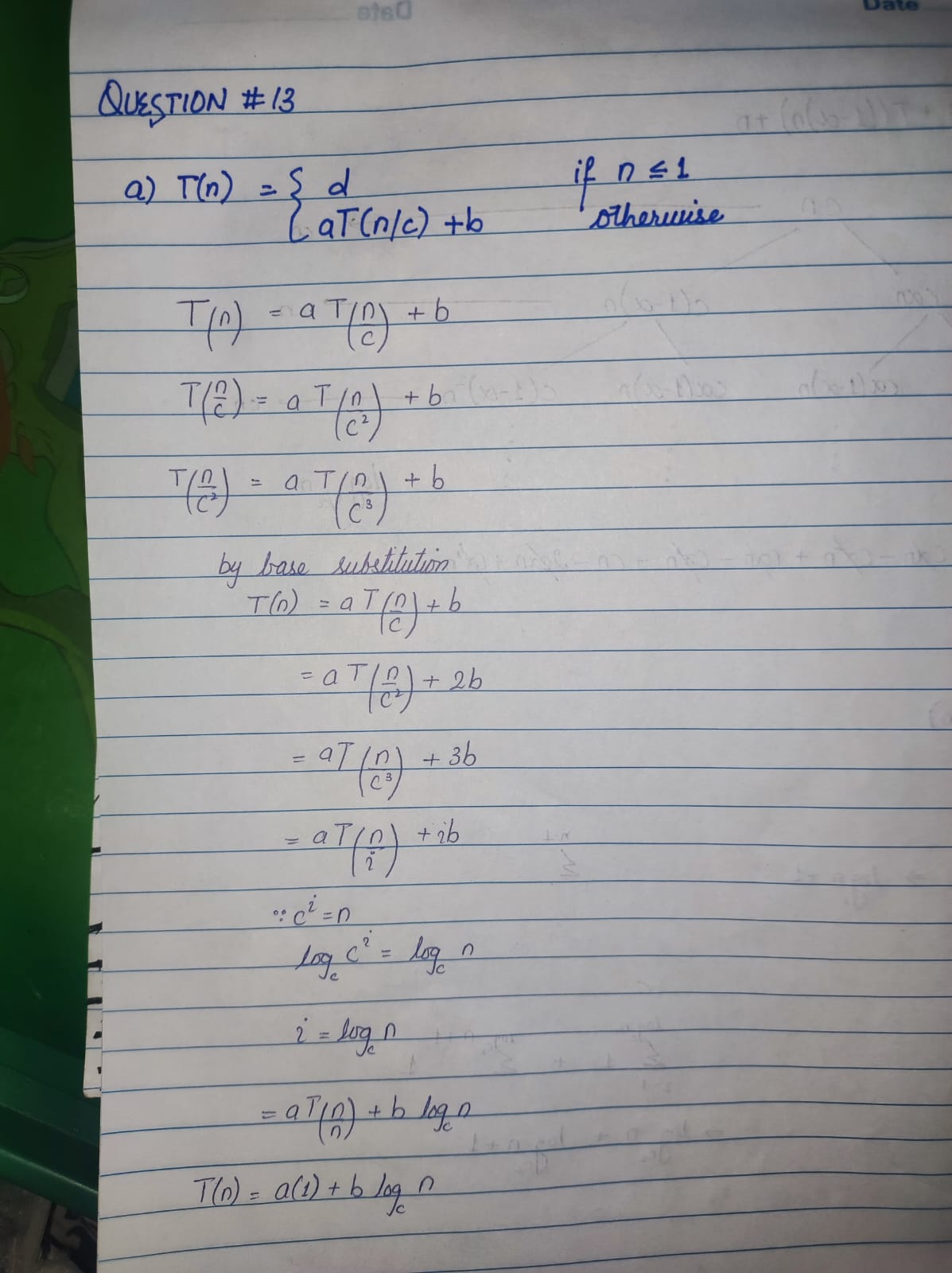
**Answer**



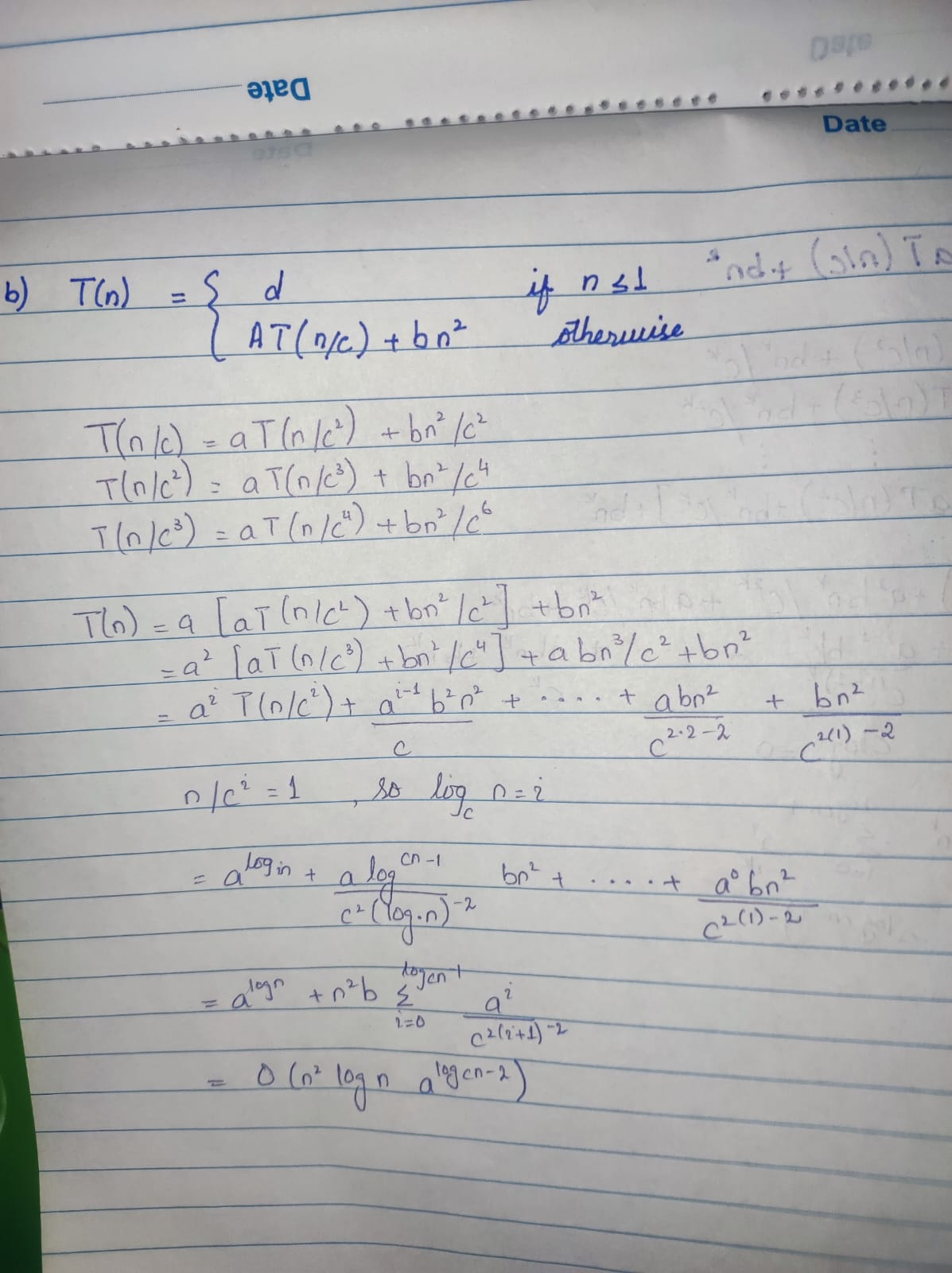
**Question 13)**

**Part A:**

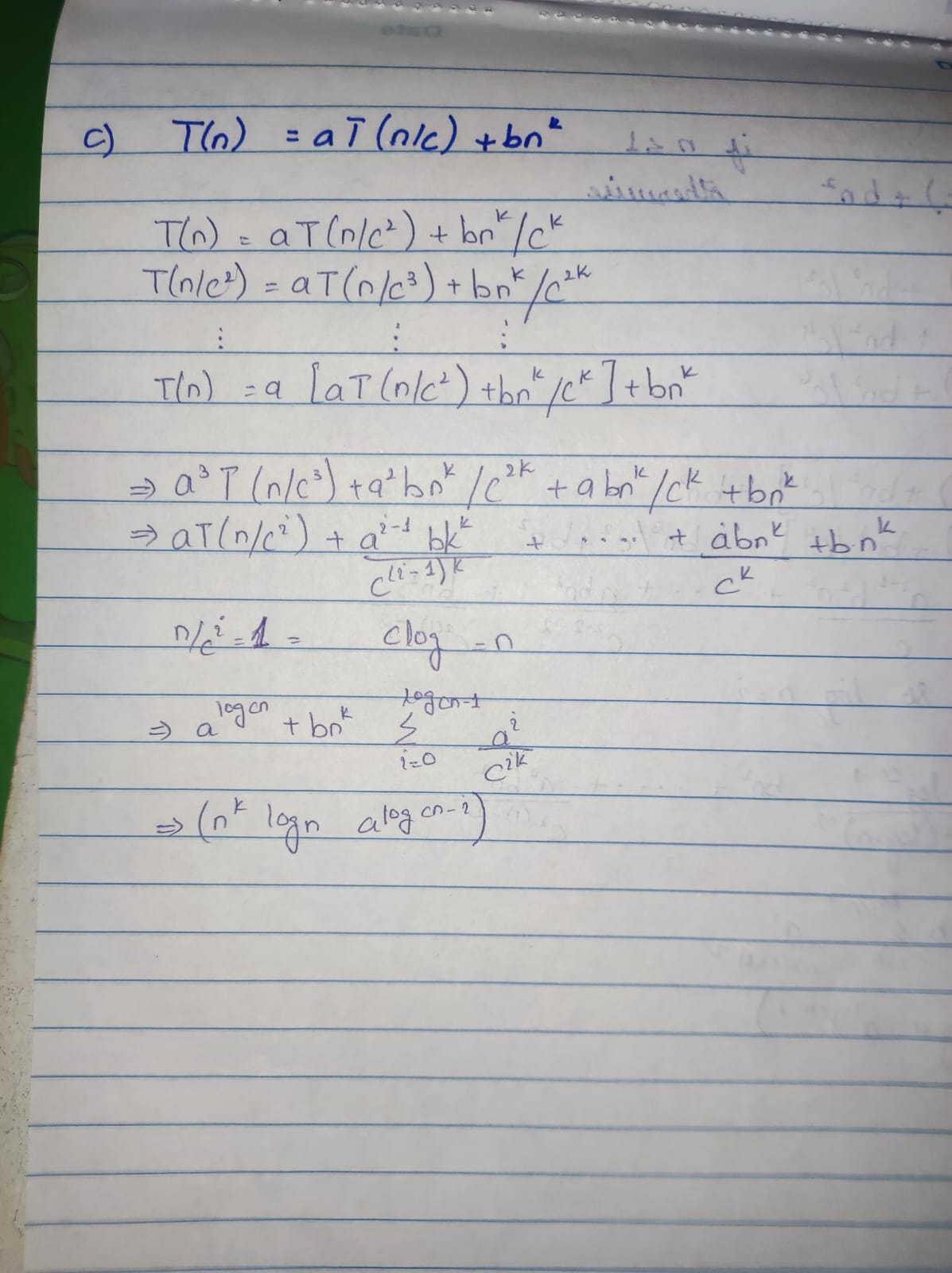
1. **Answer:**



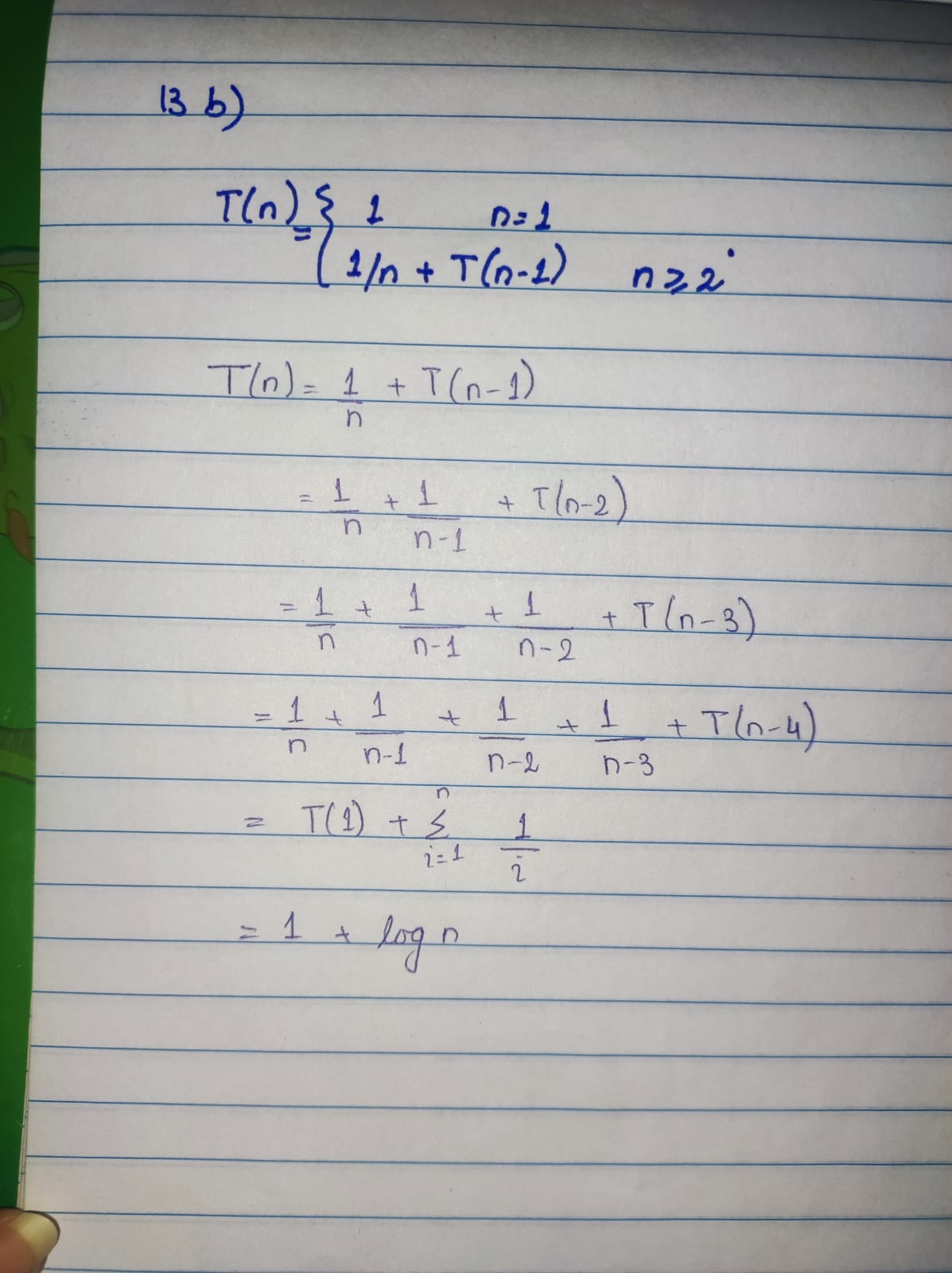
1. **Answer:**

****

1. **Answer:**

****

**Part B:**

****