Time And Space Complexity Solutions

Question 1. Analyze the time complexity of the following Java code and suggest a way to improve it:

int sum = 0;

for(int i = 1; i <= n; i++)

{ for(int j = 1; j <= i; j++)

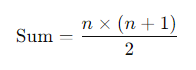
sum++;

The overall time complexity is O(n2)O(n^2)O(n2) because the number of operations grows quadratically with the input size n

**Improvement**

To improve the time complexity, we can avoid the nested loops and calculate the result directly. In this case, we are essentially counting the number of operations executed, which is equivalent to calculating the sum of the first n positive integers.

Instead of using nested loops, you can calculate the sum directly using the formula:

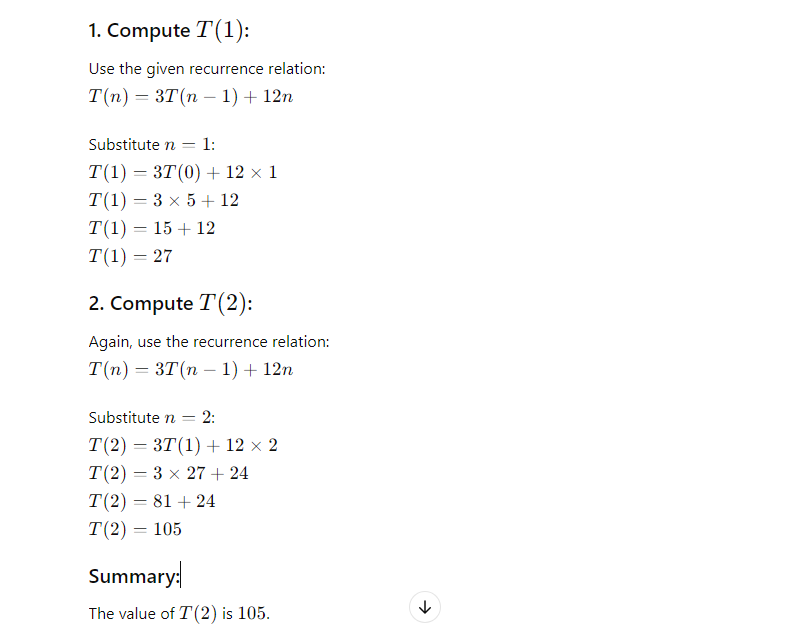


int n = 10; // Example value for n

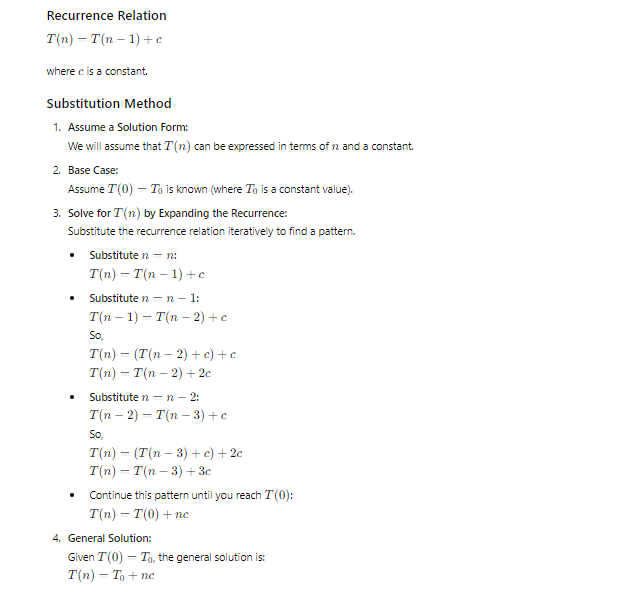
int sum = n \* (n + 1) / 2;

System.out.println("Sum: " + sum);

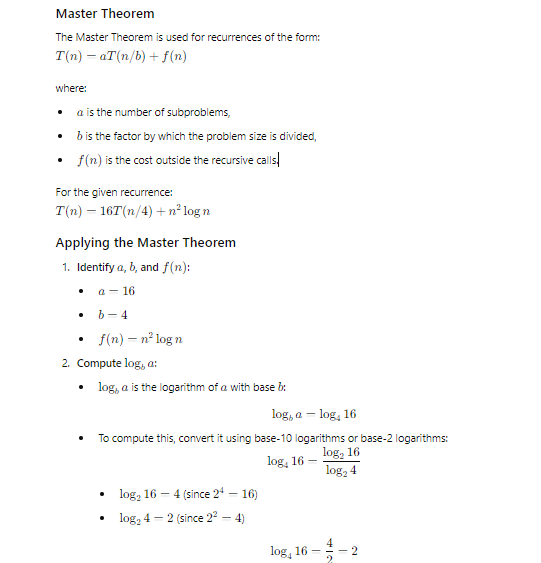
Question 2: Find the value of T(2) for the recurrence relation T(n) = 3T(n-1) + 12n, given that T(0) = 5.

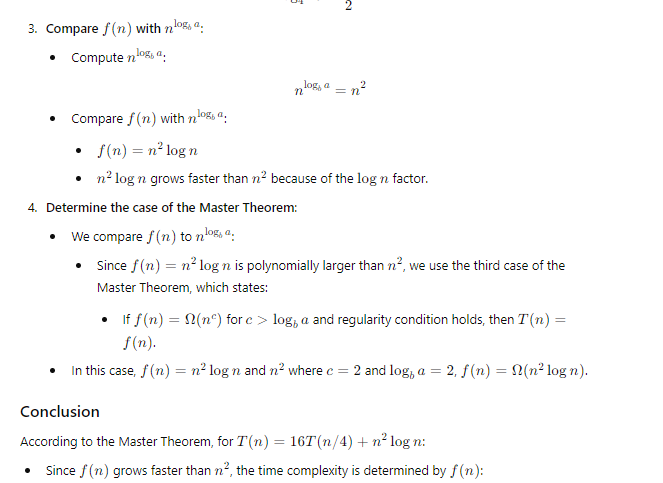


Question 3: Given a recurrence relation, solve it using a substitution method.  Relation : T(n) = T(n - 1) + c

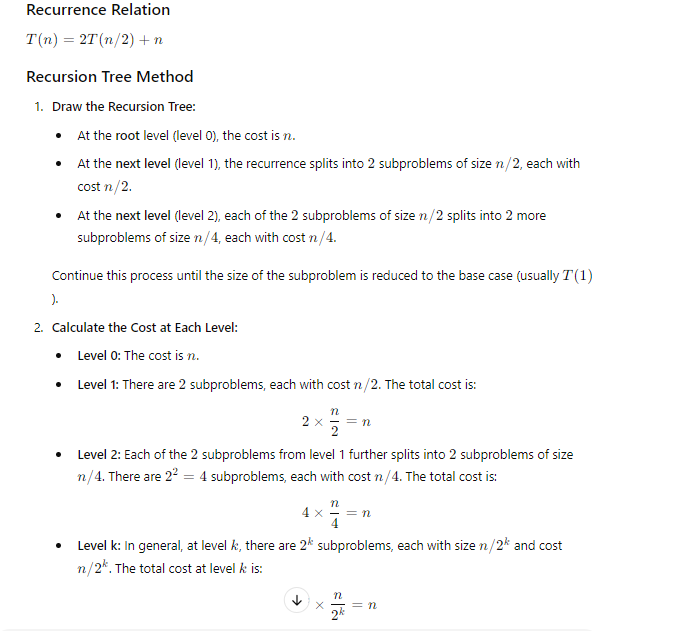


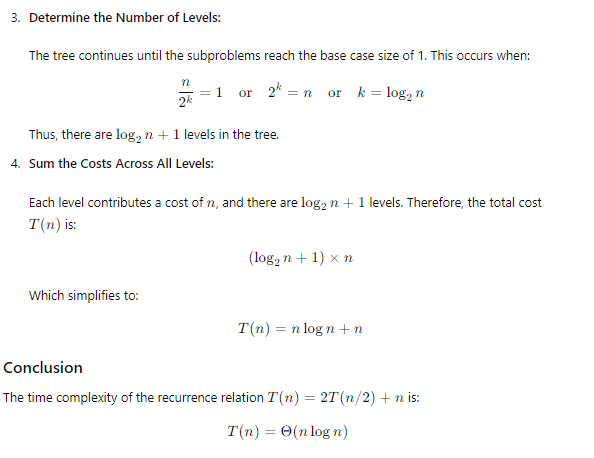
Question 4: Given a recurrence relation:  T(n) = 16T(n/4) + n2logn  Find the time complexity of this relation using the master theorem.





Question 5: Solve the following recurrence relation using recursion tree method T(n) = 2T(n/2) + n





Question 6. T(n) = 2T(n/2) + K, Solve using Recurrence tree method.

