MTech CSE – 1st Semester
Student Name: SIDDHARTH KUMAR
Student ID: A125021

Question 5

Solve the following recurrence relation arising from the LUP decomposition solve procedure:

$$T(n) = \sum_{i=1}^{n} \left[O(1) + \sum_{j=1}^{i-1} O(1) \right] + \sum_{i=1}^{n} \left[O(1) + \sum_{j=i+1}^{n} O(1) \right].$$

Answer:

Understanding the Recurrence

The given recurrence consists of two summations, each iterating from $i = 1$ to n. Each summation represents a nested-loop structure where a constant-time operation is executed repeatedly.

The objective is to count the total number of constant-time operations and determine the asymptotic time complexity.

Analysis of the First Summation

Consider:

$$\sum_{i=1}^{n} \left[O(1) + \sum_{j=1}^{i-1} O(1) \right].$$

Inner Loop

The inner summation

$$\sum_{j=1}^{i-1} O(1)$$

executes exactly $i - 1$ times, giving:

$$\sum_{j=1}^{i-1} O(1) = O(i).$$

Total Cost of the First Summation

Substituting into the outer sum:

$$\sum_{i=1}^{n} O(i).$$

Using the identity:

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

we obtain:

$$\sum_{i=1}^{n} O(i) = O(n^2).$$

Analysis of the Second Summation

Now consider:

$$\sum_{i=1}^{n} \left[O(1) + \sum_{j=i+1}^{n} O(1) \right].$$

Inner Loop

The inner summation

$$\sum_{j=i+1}^{n} O(1)$$

executes $n - i$ times, resulting in:

$$\sum_{j=i+1}^{n} O(1) = O(n - i).$$

Total Cost of the Second Summation

Thus, the second summation becomes:

$$\sum_{i=1}^{n} O(n-i).$$

As i ranges from 1 to n, the term $(n-i)$ ranges from $n-1$ to 0, and:

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

Therefore:

$$\sum_{i=1}^{n} O(n-i) = O(n^2).$$

Combining Both Parts

From the above analysis:

- The first summation contributes $O(n^2)$,
- The second summation contributes $O(n^2)$.

Hence, the total time complexity is:

$$T(n) = O(n^2).$$

Interpretation in the Context of LUP Solve

This recurrence arises from the *solve phase* of the LUP decomposition, which includes:

- forward substitution to solve $Ly = Pb$,
- backward substitution to solve $Ux = y$.

Each phase consists of a nested loop with a linear number of constant-time operations, leading to a quadratic overall cost.

Final Result

$$\boxed{T(n) = O(n^2)}$$

Conclusion

Although the LUP decomposition itself requires $O(n^3)$ time, the recurrence analyzed here corresponds only to the forward and backward substitution steps. This confirms that once the factorization is available, each system solve runs in quadratic time, making the overall method efficient for multiple right-hand sides.