Q: Consider the following list, and use the Prefix Sum algorithm and Tree architecture to compute the Prefix Sum:

$$S = \{3, 1, 7, 10\}$$

1. Display the steps of the algorithm.
2. Calculate the time complexity of the algorithm.
3. Modify the algorithm for $n > N$ and compute its time complexity.

A: The answer to the first one is "Display the steps of the algorithm" is as follows:

Tree-based Approach for Prefix Sum

The tree-based approach for calculating Prefix Sum involves two main phases:

1. **Up-Sweep (also known as the Reduce phase):** In this phase, the elements of the list are combined in pairs in a tree structure until we reach a single value at the root. This phase essentially reduces the list step by step.
2. **Down-Sweep (also known as the Distribution phase):** In this phase, the Prefix Sum values are distributed back down the tree from the root to the leaves.

> Phase 1: Up-Sweep (Reduce)

In this phase, the elements are combined in a tree-like structure, with each node being the sum of its child nodes. The process works in a binary fashion, reducing the number of elements at each step until only the root remains.

1. Level 1 (first reduction): We start by summing adjacent pairs of elements from the list: $\{3+1=4, 7+10=17\}$

   So, the list becomes: $\{4, 17\}$
2. Level 2 (second reduction): In the second level, we sum the two elements from the previous step: $\{4+17=21\}$

   Now, the value at the root is 21. This concludes the Up-Sweep phase, as we have reached a single value at the root.

> Phase 2: Down-Sweep (Distribution)

In the Down-Sweep phase, we start from the root and propagate Prefix Sum values back down to the leaves, ensuring that each element in the list gets its correct prefix value.

1. Initialize the root with 0: We set the root value to 0 and propagate this down to the first level. The list at this point looks like: $\{0, 21\}$

   The left child gets the value from the parent (0), and the right child gets the sum of the parent's value (0) and the left child's previous value (4). So, we update the list to: $\{0, 4\}$
2. Level 1 (propagate values): Now we propagate these values further down to the original list elements.

   For the first pair: $\{0+3=3, 0=0\}$

   For the second pair: $\{4+7=11, 4+10=14\}$

   Therefore, after propagating the values, the final list of Prefix Sums is: $\{0, 3, 4, 11, 21\}$

The answer to the second one is "Calculate the time complexity of the algorithm" is as follows:

Time Complexity of the Tree-based Algorithm

The time complexity of the tree-based Prefix Sum algorithm is:

1. **Up-Sweep:** At each level, the number of elements is halved, and the depth of the tree is $\log n$, where n is the number of elements. This results in $O(\log n)$ time.
2. **Down-Sweep:** The Down-Sweep phase also has $\log n$ levels, making the total complexity for this phase $O(\log n)$.

Overall time complexity for the tree-based approach:

$$T(n) = O(\log n)$$

The answer to the third one is "Modify the algorithm for $n > N$ and compute its time complexity" is as follows:

Modification for $n > N$ (Data More Than Processors)

In the case where $n > N$ (i.e., more data elements than processors), the algorithm needs to distribute the work among available processors. This is usually done by blocking the data, where each processor is assigned a block of elements. Each processor computes the Prefix Sum for its block sequentially, and then the results are combined.

Steps:

1. Local Prefix Sum: Each processor computes the Prefix Sum of its block. For example, if processor 1 is assigned $\{3, 1\}$ and processor 2 is assigned $\{7, 10\}$, each processor computes:
   - Processor 1: $\{3, 4\}$
   - Processor 2: $\{7, 17\}$
2. Global Prefix Sum: A Prefix Sum is computed for the last element of each processor's block. In our example, Combine the last elements of each block: $\{4, 17\} \rightarrow \{0, 4\}$ (from the Up-Sweep phase).
3. Adjustment Phase: Each processor adjusts its local results by adding the final result of the previous block to its local Prefix Sum. For example, processor 2 will add 4 to each of its local results:
   - Processor 2: $\{7 + 4 = 11, 17 + 4 = 21\}$

Time Complexity:

1. Local Prefix Sum: Each processor processes $\frac{n}{N}$ elements sequentially, so the complexity is $O(\frac{n}{N})$ per processor.
2. Global Prefix Sum: The global Prefix Sum phase is computed in $O(\log N)$ time using the Up-Sweep and Down-Sweep processes across the processors.
3. Adjustment Phase: Similar to the local computation, the adjustment phase takes $O(\frac{n}{N})$ time.

Overall time complexity when $n > N$:

$$T(n) = O(\frac{n}{N} + \log n)$$

Q: Consider the following list, and use the Prefix Sum algorithm and Mesh architecture to compute the Prefix Sum:

$$S = \{1, 4, 3, 6, 3, 10, 2, 7, 14, 2, 4, 5, 10, 1, 1, 1\}$$

1. Calculate the time complexity and the final cost of the algorithm.
2. If $n > N$, modify the algorithm to compute the Prefix Sum and its time complexity.

A: The answer to the first one is ”Calculate the time complexity and the final cost of the algorithm” is as follows:

Prefix Sum Using a Mesh Architecture

Mesh Architecture Overview

In a Mesh architecture, processors are arranged in a 2D grid. Each processor can communicate with its adjacent neighbors (north, south, east, and west). The number of processors in the grid is typically $\sqrt{N} \times \sqrt{N}$, where N is the total number of processors. This kind of architecture allows for efficient parallel computation.

Steps for Computing Prefix Sum

We assume that the list S of 16 elements is assigned to 16 processors, one element per processor, arranged in a 4×4 mesh grid.

1. Initial Setup (Assign Data to Processors): Each processor holds one element of the input list:

$$\begin{bmatrix} P_{1,1}=1 & P_{1,2}=4 & P_{1,3}=3 & P_{1,4}=6 \\ P_{2,1}=3 & P_{2,2}=10 & P_{2,3}=2 & P_{2,4}=7 \\ P_{3,1}=14 & P_{3,2}=2 & P_{3,3}=4 & P_{3,4}=5 \\ P_{4,1}=10 & P_{4,2}=1 & P_{4,3}=1 & P_{4,4}=1 \end{bmatrix}$$

2. Step 1: Local Computation of Prefix Sums (Within Each Row): Each processor computes a local Prefix Sum within its row by communicating with the processor to its left. This is done in parallel for each row:

   - For row 1:

     $P_{1,2} = P_{1,1} + P_{1,2} = 1 + 4 = 5$

     $P_{1,3} = P_{1,2} + P_{1,3} = 5 + 3 = 8$

     $P_{1,4} = P_{1,3} + P_{1,4} = 8 + 6 = 14$

     Updated row 1: $\{1, 5, 8, 4\}$

   Similar steps are applied for the other rows and update the matrix:

$$\begin{bmatrix} 1 & 5 & 8 & 14 \\ 3 & 13 & 15 & 22 \\ 14 & 16 & 20 & 25 \\ 10 & 11 & 12 & 13 \end{bmatrix}$$

3. Step 2: Global Prefix Sum Across Rows (Between Rows): After computing local Prefix Sums within each row, processors in the first column (leftmost) send their values down the column to the processors below them. These values are used to adjust the Prefix Sums for the rows below.

   - Processor $P_{2,1}$ adds the value from $P_{1,1}$:

     $P_{2,1} = P_{2,1} + P_{1,4} = 3 + 14 = 17$

   - Similarly, $P_{3,1}$ and $P_{4,1}$ are updated:

     $P_{3,1} = P_{3,1} + P_{2,4} = 14 + 22 = 36$

     $P_{4,1} = P_{4,1} + P_{3,4} = 10 + 25 = 35$

   Using these new values, the rows below are updated similarly and update matrix:

$$\begin{bmatrix} 1 & 5 & 8 & 14 \\ 17 & 27 & 29 & 36 \\ 36 & 38 & 42 & 47 \\ 35 & 36 & 37 & 38 \end{bmatrix}$$

The final result (Prefix Sum) is:

$$\{1, 5, 8, 14, 17, 27, 29, 36, 36, 38, 42, 47, 35, 36, 37, 38\}$$

Time Complexity

In a Mesh architecture, the time complexity is determined by how information is propagated across the rows and columns of the grid.

1. Local Computation (Within Rows): Since each processor only communicates with its immediate neighbor in the row, the time complexity for this step is proportional to the number of elements in a row, which is $\sqrt{n}$ for an n-element list: $O(\sqrt{n})$
2. Global Computation (Between Rows): Communication between rows happens in parallel for each column, so the complexity is also $O(\sqrt{n})$ for this step.

Thus, the total time complexity is:

$$T(n) = O(\sqrt{n})$$

Cost of the Algorithm

The cost of an algorithm in parallel computing is defined as the product of the time complexity and the number of processors. Given that there are $N = n$ processors in a $\sqrt{n} \times \sqrt{n}$ mesh, the cost is:

$$Cost = T(n) \times N = O(\sqrt{n}) \times O(n) = O(n^{1.5})$$

Modification for $n > N$ (More Data Elements Than Processors)

When $n > N$, meaning there are more data elements than processors, the data must be divided among the available processors. Assume there are N processors, and each processor handles multiple data elements (in blocks).

Steps for $n > N$:

1. Divide Data into Blocks: Each processor is assigned $\frac{n}{N}$ data elements. For instance, if $n = 16$ and $N = 4$, each processor handles 4 elements.
2. Local Prefix Sum Computation: Each processor computes the Prefix Sum for its assigned block sequentially. This takes $O(\frac{n}{N})$ time per processor.
3. Global Prefix Sum Communication: After the local computation, processors communicate with each other to combine their results using the mesh architecture. This step takes $O(\sqrt{N})$ time, as it involves communication across the processors in the mesh.
4. Adjustment Phase: Each processor adjusts its local results based on the Prefix Sum values computed by other processors. This step also takes $O(\frac{n}{N})$ time.

Time Complexity for $n > N$:

The total time complexity for the case when $n > N$ is the sum of the local computation time and the global communication time:

$$T(n) = O(\frac{n}{N}) + O(\sqrt{N})$$

Final Cost for $n > N$:

The cost in this case is:

$$Cost = T(n) \times N = O((\frac{n}{N} + \sqrt{N}) \times N) = O(n) + O(N^{1.5})$$

Thus, the total cost depends on both the size of the data and the number of processors.