Chapter 10: PRAM Algorithms

What are we studying in this chapter?

- Introduction
- Computational Model
- Parallel Algorithms for:
  - Prefix Computation
  - List Ranking
- Graph Problems

- 6 hours

10.1 Introduction

So far in the previous chapters, we have seen how to write algorithms and programs for machines that uses only one processor. Now, let us see how to write the algorithms for parallel machines (i.e., computers with more than one processor). There are many applications that require extensive computations to be completed within hours or sometimes within minutes which is not possible using a computer with one processor. For example,

- Weather forecasting has to be done in a timely fashion
- In case of hurricanes or snowstorms, evacuation has to be done in a short period of time
- If an expert system is used to aid a physician in surgical procedures, decisions have to be made within seconds and so on.

The programs written for such applications have to perform an enormous amount of computations within the specified time. In such situations, we use computers with multiple processors or parallel machines.

The action taken by a computer with multiple processors can be explained using the following analogy: Suppose there are 200 numbers to be added and there are two persons. The first person can add 100 numbers and the second person can add remaining 100 numbers simultaneously. When both of them have finished adding 100 numbers each, one of them can add the two individual sums to get the final answer. Thus, two people can add 200 numbers in almost half the time required by one person.

On similar lines of the above problem, computations can be done parallely by a parallel computer thus saving the computational time. Now, let us see "What is the procedure to be followed by any parallel computer while performing an activity?" The various activities are shown below:

- The problem to be solved is partitioned into many sub-problems.
- Each sub-problem is solved by one processor
- When all the processors have completed solving the problems assigned to them, the partial solutions are combined to arrive at the final answer.
- If there are p-processors, then potentially we can cut down the solution time by a factor of p.

So, the algorithms that we write for a machine with single-processor are different from algorithms that we write for a computer with multiple-processors. (Note that the computers that have two processors are called dual-cores, the computers that have four processors are called quad-cores and so on.)

Now, let us see "How the algorithms are classified based on the machines for which they are written?" The algorithms are classified into two categories

- Sequential algorithms
- Parallel algorithms.

Sequential algorithms: The algorithms that are designed for a single-processor machine are called sequential algorithms. For example, all the algorithms that we have written so far in previous chapters are all sequential algorithms.

Parallel algorithms: The algorithms that are designed for a multiprocessor machine are called parallel algorithms.

Now, before designing parallel algorithms let us "Define the terms: speedup, asymptotic speedup, linear speedup, total work done by algorithm, efficiency of the algorithm, work optimal"

Definition: Let $S'(n)$ is the time complexity of a sequential algorithm for a given problem where n is the problem size. Let $T'(n, p)$ is the time complexity of a parallel algorithm on a p-processor machine. The speedup of the parallel algorithm is the ratio of $S'(n)$ and $T'(n, p)$ i.e., speedup of parallel algorithm $= \frac{S'(n)}{T'(n, p)}$

Definition: Let S(n) is the asymptotic time complexity of a given problem where n is the problem size. Let T(n, p) is the asymptotic time complexity of a parallel algorithm. The asymptotic speedup of the parallel algorithm is the ration of S(n) and T(n, p) i.e., asymptotic speedup $= \frac{S(n)}{T(n, p)}$. If $\frac{S(n)}{T(n, p)} = \theta(n)$, then the algorithm is said to have *linear speedup.*

Definition: Let T(n, p) is the time complexity of the parallel algorithm on a p-processor computer. Then the total work done by the algorithm is defined as the product number of processors used by the computer and the time complexity of the parallel algorithm. That is, work done by algorithm = p*T(n,p).

Definition: Let S(n) is the asymptotic run time of best known sequential algorithm for solving a given problem. Let p.T(n,p) is the work done by parallel algorithm. The efficiency of the algorithm is defined as the ratio of S(n) to work done by the algorithm

i.e., efficiency of the algorithm = $\frac{S(n)}{p*T(n,p)}$

Definition: If S(n) is the asymptotic run time of best known sequential algorithm and p*T(n,p) is the total work done by the algorithm, then the algorithm is said to be work optimal if p*T(n,p) = O(S(n)).

Note: A parallel algorithm is work-optimal if and only if that algorithm has linear speedup. The efficiency of the work-optimal parallel algorithm is given by θ(1).

Now, the question is "What is super-linear speedup?"

Definition: If it is possible to get a speedup of more than p for any problem on a p-processor machine then the speedup is called *superlinear speedup*. When the speedup is defined with respect to the actual run times on the sequential and parallel machines, it is possible to get *superlinear speedup*.

Now, let us see "How to solve the problem in parallel?" There are many ways of solving a problem in parallel:

- One of the solving techniques is to explore many techniques (i.e., algorithms) and identify the one that can be most parallelizable.
- To achieve good speedup, it is necessary to parallelize every component of the underlying technique.
- If a fraction f of the technique cannot be parallelized (i.e., it has to run serially), then the maximum speed that can be obtained is limited by f.
- Using Amdah's law maximum speedup can be calculated as shown below:

$$\text{Maximum speedup} = \frac{1}{f + \frac{1-f}{p}}$$

Example 10.1 : Obtain the maximum speedup when p = 10 and for various values of f = 0.5, 0.1, 0.01

Solution: The maximum speed up for the various values of f when p = 10 can be calculated as shown below using the formula:

$$\text{Maximum speedup} = \frac{1}{f + \frac{1-f}{p}}$$

f	Maximum speedup = $\frac{1}{f + \frac{1-f}{p}}$
0.5	Maximum speedup = $\frac{1}{0.5 + \frac{1-0.5}{10}} = 1.8181 \approx 2.$ So, speedup is slightly less than 2!
0.1	Maximum speedup = $\frac{1}{0.1 + \frac{1-0.1}{10}} = 5.26 \approx 5.$ So, speedup is slightly more than 5!
0.01	Maximum speedup = $\frac{1}{0.01 + \frac{1-0.01}{10}} = 9.17 \approx 9.$ So, speedup is slightly more than 9!

10.2 Computational Model

In this section, let us see "What are the different types of computational models" The different types of computational models are:

- Sequential computational model
- Parallel computational model
  - Fixed connection machines
  - Shared memory machines

Sequential computational model: The sequential model that we have employed so far is the Random Access Machine (RAM). In this model, we assume that various operations such as: addition, subtraction, multiplication and various other operations can be performed in one unit of time. This model is widely accepted as a valid sequential model

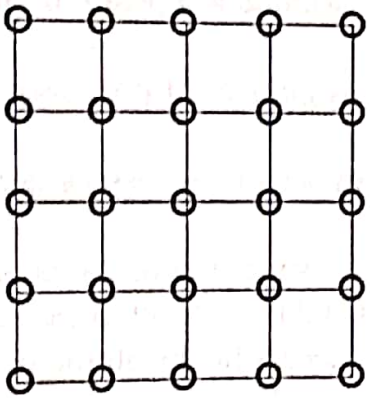
Parallel computational model: The main feature in parallel computing is inter-processor communication. That is, given any problem, the processors have to communicate among themselves and agree on the sub-problems each will work on. The processors need to communicate each other to see whether everyone has finished its task and so on. Each machine or processor in a parallel computer can be assumed to be a Random Access machine (RAM). The parallel models are classified into:

- Fixed connection models
- Shared memory models

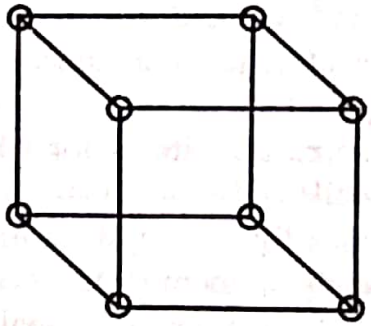
Now, let us see "What is a fixed connection model?" A fixed connection model is nothing but a graph G = (V, E) where

- V is set of vertices and each vertex represents a processor
- E is set of edges and each edge represent communication links between processors

The various examples of fixed connection machines are shown below:



(a) Mesh



(b) Hyper cube

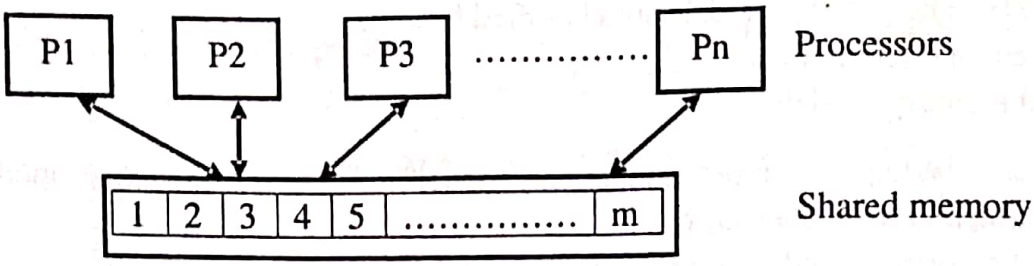
Observe the following points in any of the fixed connection model:

- In inter-processor communication, communication is done through the communication links. Any two processors connected by an edge in G can communicate in one step.
- In general, the two processors can communicate through any of the paths connecting them.
- The communication time depends on the lengths of these paths.

Now, let us see "What is shared memory model?"

Definition: The shared memory model is the one where a number of processors are working synchronously. This model is also called Parallel Random Access Machine (PRAM). The processors communicate with each other using a common block of global

memory that can be accessed by all processors. This global memory using which all the processors share and communicate is called shared memory or common memory. Communication is performed by writing to this shared memory and reading from this shared memory. The pictorial representation is shown below:



Suppose i and j are two processors. The processor i writes its message into memory cell j and next, the processor j reads from this cell. Thus, the two processors i and j communicate each other. Observe the following points:

- Each processor P1, P2,....Pn in a PRAM is a RAM with some local memory.
- A single step of a PRAM algorithm can be arithmetic operation such as addition, subtraction etc., comparison, assignment etc.
- The number of cells are assumed to be same as number of processors. But, this may not be true in all the cases
- There are some algorithms for which the number of processors are more than the number of cells and vice-versa.
- There is space for storing the various inputs to various processors and there is also space in the global memory to store the output and to store intermediate results
- Since the shared memory or global memory is accessible by all the processors, access conflicts may arise.

Now, let us see "What are the different ways of resolving read and write conflicts?" The various conflicts that may arise are shown below:

- EREW (**E**xclusive **R**ead and **E**xclusive **W**rite)
- CREW (**C**oncurrent **R**ead and **E**xclusive **W**rite)
- CRCW (**C**oncurrent **R**ead and **C**oncurrent **W**rite)

Now, let us see "What is EREW PRAM?"

Definition: The Exclusive Read and Exclusive Write (EREW) Parallel Random Access Machine (PRAM) is the shared memory model in which concurrent read or write is not allowed on any cell of the shared memory.

For example, at a given time, one processor might access cell six and at the same time other processor might access cell 15 and so on. But, the two processors cannot access memory cell 24 at the same time.

Now, let us see "What is CREW PRAM?"

Definition: The Concurrent Read and Exclusive Write (CREW) PRAM is a variation that permits concurrent reads and not concurrent writes.

For example, at a given time, two processors may access cell six for read operation. But, the two processors cannot write data to memory cell 24 at the same time.

Now, let us see "What is CRCW PRAM?"

Definition: The Concurrent Read and Concurrent Write (CRCW) PRAM is a variation that permits concurrent reads and concurrent writes.

For example, at a given time, two processors may access cell six for read operation and the two processors can write data to memory cell 24 at the same time.

Note: In CREW or CRCW PRAM observe that if more than one processor tries to read from the same cell, they read the same information.

Example 10.2 : Consider a 4-processor machine and write the code for performing concurrent read operation

---

Solution: There is a machine with 4-processors. The concurrent read operation must be performed from all 4-processors. This can be done using the following statement:

Processor *i* (in parallel for $1 \leq i \leq n$) does:
read M[1]

This concurrent read operation can be performed in one unit of time on CRCW as well as on CREW PRAMs.

Example 10.3 : Consider a 4-processor machine and write the code for performing concurrent write operation

---

Solution: There is a machine with 4-processors. The concurrent write operation must be performed from all 4-processors. This can be done using the following statement:

Processor *i* (in parallel for $1 \leq i \leq n$) does:
write M[1]

This concurrent write operation can be performed in one unit of time on CRCW PRAM.

Example 10.4 : Consider the following Boolean operation:

a[0] = a[1] || a[2] || .....|| a[n]

Write the algorithm to perform Boolean OR operation in O(1) time

Solution: The algorithm to perform Boolean OR operation in O(1) time is shown below:

Processor *i* (in parallel for $1 \leq i \leq n$) does;

if (a[i] = 1) then a[0] = a[i];

In a RAM, the time complexity of above Boolean operation is O(n). But, in a PRAM a[0] can be computed in θ(1) time using an n-processor CRCW PRAM. This is done as shown below:

- Assume that a[0] is zero to start with.
- In the first time step, processor *i* for $1 \leq i \leq n$, reads memory location a[i] and checks whether a[i] is 1. If it is true, 1 is written into memory location a[0].
- Since several of a[i]'s may be 1, several processor may write 1 to a[0] concurrently.

10.3 Fundamental techniques and algorithms

In this section, let us see "What are the two basic problems that arise in the parallel solution of numerous problems?"

The different types of basic problems that arise in the parallel solution are:

- Prefix computation problem
- List ranking problem

10.3.1 Prefix computation

Now, let us see "How to solve prefix computation problems using sequential method?" Let $\sum$ is a domain in which the binary operator $\oplus$ is defined. The binary operator $\oplus$ may represent addition, subtraction, multiplication etc. An operator $\oplus$ is said to be associative if for any three elements *x*, *y*, *z* from $\sum$:

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

Ex 1: (2 + 3) + 6 = 2 + (3 + 2)

Ex 2: (2 * 3) * 6 = 2 * (3 * 6)

Observe that if an operator is associative, the order in which the operation $\oplus$ is performed does not matter. Let us assume that $\oplus$ is unit time computable and that $\sum$ is closed under this operation. This indicates that if $x, y \in \sum$, then $(x \oplus y) \in \sum$

Now, let us see "What is prefix computation problem?"

Definition: Let $\sum$ has n elements $x_1, x_2, x_3 \ldots\ldots x_n$. The prefix computation problem is to compute n elements:

$$\underbrace{x_1}_{1},\ \underbrace{x_1 \oplus x_2}_{2},\ \underbrace{x_1 \oplus x_2 \oplus x_3}_{3},\ \underbrace{x_1 \oplus x_2 \oplus x_3 \oplus x_4}_{4},\ \ldots\ldots\ldots\ldots,\ \underbrace{(x_1 \oplus x_2 \oplus x_3 \oplus x_4 \ldots\ldots \oplus x_4)}_{n}$$

The output elements are often referred to as the prefixes.

Example 10.5: Let $\sum$ be the set of integers.

a) $\oplus$ is the usual addition operation. The input to the prefix computation problem is: 3, -5, 8, 2, 5, 4. Obtain the prefixes.

b) $\oplus$ is the usual multiplication operation. The input to the prefix computation problem is: 2, 3, 1, -2, -4. Obtain the prefixes.

Solution:

a) $\oplus$ is the usual addition operation Let $(x_1, x_2, x_3, x_4, x_5, x_6) = (3, -5, 8, 2, 5, 4)$

$x_1 = 3$

$x_1 \oplus x_2 = 3 - 5 = -2$

$x_1 \oplus x_2 \oplus x_3 = 3 - 5 + 8 = 6$

$x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 3 - 5 + 8 + 2 = 8$

$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 3 - 5 + 8 + 2 + 5 = 13$

$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 = 3 - 5 + 8 + 2 + 5 + 4 = 17$

So, the output = (3, -2, 6, 8, 13, 17)

b) $\oplus$ is the usual multiplication operation Let $(x_1, x_2, x_3, x_4, x_5) = (2, 3, 1, -2, -4)$

$x_1 = 2$

$x_1 \oplus x_2 = 2 * 3 = 6$

$x_1 \oplus x_2 \oplus x_3 = 2 * 3 * 1 = 6$

$x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 2 * 3 * 1 * (-2) = -12$

$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 2 * 3 * 1 * (-2) * (-4) = 48$

So, the output = (2, 6, 6, -12, 48)

Example 10.6: Let $\sum$ be the set of integers and $\oplus$ be the minimum operator. The input to the prefix computation problem is: 5, 8, -2, 7, -11, 12. Obtain the prefixes.

$\oplus$ is the minimum operator and it is associative:

Let $(x_1, x_2, x_3, x_4, x_5, x_6) = (5, 8, -2, 7, -11, 12)$

x_1	$= 5$
$x_1 \oplus x_2 = 5, 8$	$= 5$
$x_1 \oplus x_2 \oplus x_3 = 5, 8, -2$	$= -2$
$x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 5, 8, -2, 7$	$= -2$
$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = 5, 8, -2, 7, -11$	$= -11$
$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 = 5, 8, -2, 7, -11, 12$	$= -11$

So, the output = (5, 5, -2, -2, -11, -11)

Observe that the above prefix computational problems can be solved in O(n) time sequentially.

Now, let us see "How to solve prefix computation problems using parallel algorithms?" The prefix computation problems can be solve using work-optimal algorithms, on many models of parallel computing with less time. It can be shown that the efficiency of work-optimal algorithm that solves the prefix computational problem is $\theta(1)$ and speedup of such algorithms is $\theta(n / \log n)$.

Now, let us "Write the prefix algorithm using divide and conquer strategy" Let the input be $(x_1, x_2, x_3 \ldots\ldots x_n)$. Without loss of generality assume that n is an integral power of 2. We use n-processor machine which takes time O(log n) as shown below:

Example 10.7: Let the input to the prefix computation problem is 12, 3, 6, 8, 11, 4, 5, 7 and let $\oplus$ be addition. Obtain the prefixes using divide and conquer with n = 8 and p = 8.

Step 1: If n = 1, one processor output x_1.

Step 2:

a) Let the first n/2 processors compute the prefixes $x_1, x_2, x_3 \ldots\ldots x_{n/2}$ recursively and let $y_1, y_2, y_3 \ldots\ldots y_{n/2}$ be the result.

b) Let the rest of n/2 processors compute the prefixes $x_{n/2+1}, x_{n/2+2}, x_{n/2+3} \ldots\ldots x_n$ recursively and let $y_{n/2+1}, y_{n/2+2}, y_{n/2+3} \ldots\ldots y_n$ be the result.

Step 3: Note that the first half of the final answer is same as $y_1, y_2, y_3, \ldots\ldots, y_{n/2}$ and second half of the final answer is $y_{n/2+1}, y_{n/2+2}, y_{n/2+3}, \ldots\ldots, y_n$. Now, the second half of the processors read $y_{n/2}$ concurrently from the shared memory and update their answers which takes O(1) time

Example 10.8: Let the input to the prefix computation problem is 12, 3, 6, 8, 11, 4, 5, 7 and let $\oplus$ be addition. Obtain the prefixes using divide and conquer with n = 8 and p = 8.

As per the above algorithm in step 2.a, the first n/2 processors i.e., processors 1 to 4 compute the prefix sums of 12, 3, 6, 8 whose result is obtained as shown below:

$x_1 = 12$
$x_1 \oplus x_2 = 12 + 3 = 15$
$x_1 \oplus x_2 \oplus x_3 = 12 + 3 + 6 = 21$
$x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 12 + 3 + 6 + 8 = 29$

So, the output = (12, 15, 21, 29)

As per the above algorithm in step 2.b, the remaining n/2 processors i.e., processors 5 to 8 compute the prefix sums of 11, 4, 5, 7 whose result is obtained as shown below:

$x_5 = 11$
$x_5 \oplus x_6 = 11 + 4 = 15$
$x_5 \oplus x_6 \oplus x_7 = 11 + 4 + 5 = 20$
$x_5 \oplus x_6 \oplus x_7 \oplus x_8 = 11 + 4 + 5 + 7 = 27$

So, the output = (11, 15, 20, 27)

In step 3 of the algorithm, the processors 1 to 4 don't do anything. The processor 5 to 8 update their result by adding 29 (which is the final result of step 2.a) to every prefix sum

(11, 15, 20, 27) to get the result i.e.,

+29 +29 +29 +29

↓ ↓ ↓ ↓

(40, 44, 49, 56)

So, the final output = (12, 15, 21, 29, 40, 44, 49, 56)

The algorithm given in example 10.7 is not work-optimal for the prefix computation problem. The work-optimal algorithm can be obtained as shown below:

- In a work-optimal algorithm the number of processors used = $\frac{n}{\log n}$
- The number of inputs to each processor = $\frac{n}{\log n}$
- Now use the algorithm given in example 10.7 and compute the prefixes of the reduced input. Note that every processor will be in charge of computing log n final answers.
- Finally compute all n prefixes.

The work-optimal logarithmic time prefix computation can be written as shown below:

Step 1: Processor i (where i = 1 to $\frac{n}{\log n}$) computes the prefixes of its log n elements in parallel

Step 2: A total of $\frac{n}{\log n}$ processors use algorithm given in example 10.7 to compute the prefixes of $\frac{n}{\log n}$ elements.

Step 3: Each processor updates the prefixes it computed in step 1.

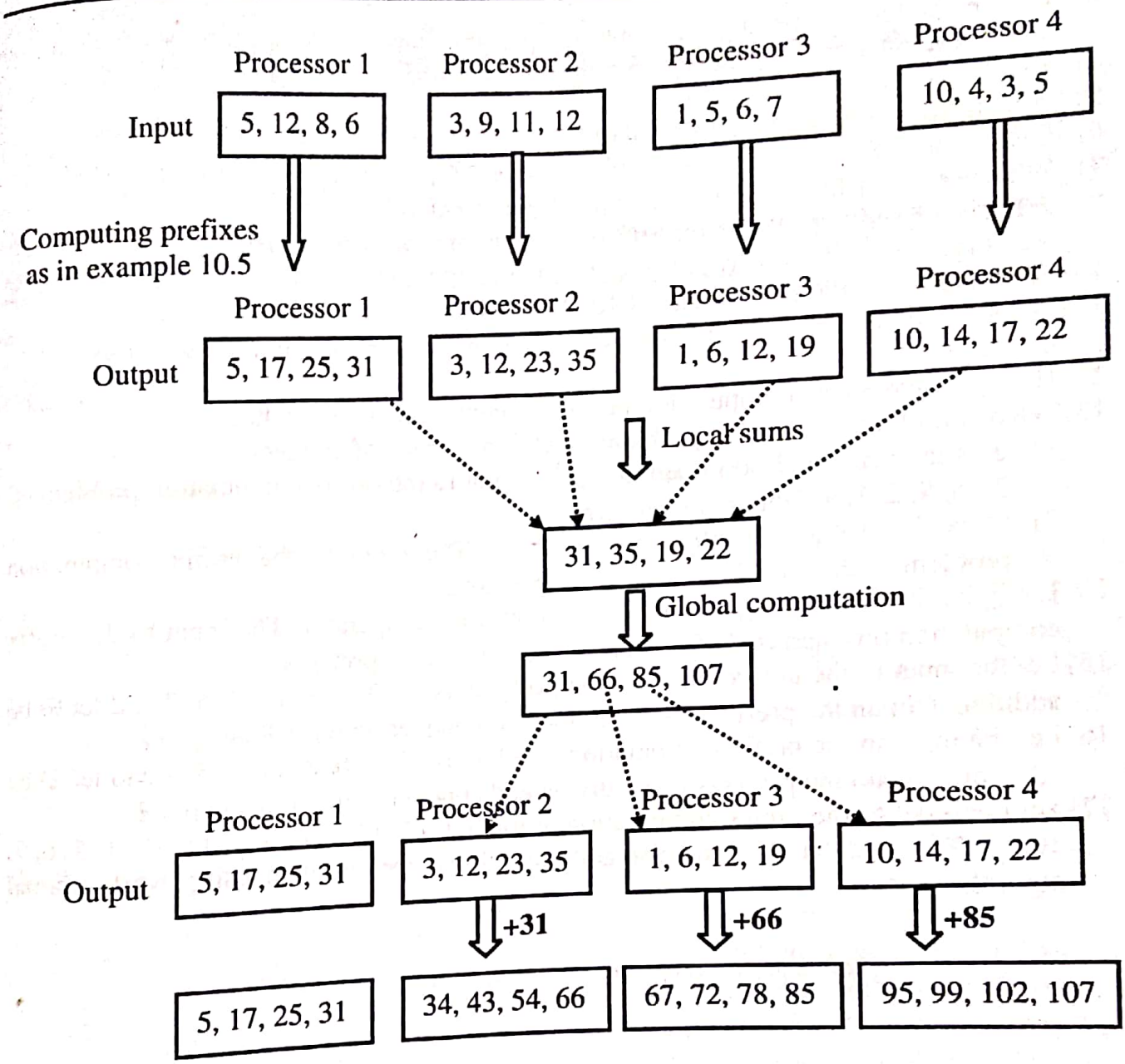
Example 10.9: Let the input to the prefix computation problem be 5, 12, 8, 6, 3, 9, 11, 12, 1, 5, 6, 7, 10, 4, 3, 5 and let ⊕ stand for addition. Solve the problem using work-optimal algorithm.

Given the following information:

- Input: 5, 12, 8, 6, 3, 9, 11, 12, 1, 5, 6, 7, 10, 4, 3, 5
- n = number of elements in input = 16
- Operation ⊕ to be performed = addition

To solve the problem using work-optimal algorithm method let us compute the following:

- Number of processors = $\frac{n}{\log n} = \frac{16}{\log 16} = \frac{16}{\log_2 16} = 4$
- Number of input for each processor = $\frac{n}{\log n}\ \frac{16}{\log 16} = \frac{16}{\log_2 16} = 4$
- So, 4 processors are used with each processor computing prefixes for 4 inputs. This can be done as shown below:



So, final output: 5, 17, 25, 31, 34, 43, 54, 66, 67, 72, 78, 85, 95, 99, 102, 107

Exercises:

1) What is the procedure to be followed by any parallel computer while performing an activity?
2) How the algorithms are classified based on the machines for which they are written?

3) Define the terms: speedup, asymptotic speedup, linear speedup, total work done by algorithm, efficiency of the algorithm, work optimal"
4) What is super-linear speedup?"
5) How to solve the problem in parallel?
6) Obtain the maximum speedup when p = 10 and for various values of f = 0.5, 0.1, 0.01
7) What are the different types of computational models?
8) What is a fixed connection model? What is shared memory model?
9) What are the different ways of resolving read and write conflicts
10) What is EREW PRAM? What is CREW PRAM? What is CRCW PRAM?
11) What are the two basic problems that arise in the parallel solution of numerous problems?
12) How to solve prefix computation problems using sequential method?
13) What is prefix computation problem? Let $\sum$ be the set of integers.
    1) $\oplus$ is the usual addition operation. The input to the prefix computation problem is: 3, -5, 8, 2, 5, 4. Obtain the prefixes.
    2) $\oplus$ is the usual multiplication operation. The input to the prefix computation problem is: 2, 3, 1, -2, -4. Obtain the prefixes.
14) Let $\sum$ be the set of integers and $\oplus$ be the minimum operator. The input to the prefix computation problem is: 5, 8, -2, 7, -11, 12. Obtain the prefixes.
15) Let the input to the prefix computation problem is 12, 3, 6, 8, 11, 4, 5, 7 and let $\oplus$ be addition. Obtain the prefixes using divide and conquer with n = 8 and p = 8.
16) Let the input to the prefix computation problem is 12, 3, 6, 8, 11, 4, 5, 7 and let $\oplus$ be addition. Obtain the prefixes using divide and conquer with n = 8 and p = 8.
17) Let the input to the prefix computation problem be 5, 12, 8, 6, 3, 9, 11, 12, 1, 5, 6, 7, 10, 4, 3, 5 and let $\oplus$ stand for addition. Solve the problem using work-optimal algorithm.