Fundamental Techniques and Algorithms:

Introduce two basic problems that arise in the parallel solution of numerous problems:

1) The first problem is known as the prefix computation problem.

2) The second problem is called the list ranking problem.

Prefix Computation:

Let ε be any domain in which the ~~library~~ binary associative.

An operator $\oplus$ is said to be associative if for any three elements $x, y,$ and z from ε,

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

that is, the order in which the operation $\oplus$ is performed does not matter.

It is also assumed that $\oplus$ is unit time computable and ε is closed under this operation

i.e for any $x, y \in \varepsilon$, $x \oplus y \in \varepsilon$.

The prefix computation problem on ε has an input n elements from ε,

say $x_1, x_2 ----, x_n$

The problem is to compute the n elements

$x_1, x_1 \oplus x_2, ---- x_1 \oplus x_2 \oplus ---- x_n$.

The o/p elements are often referred to as the prefixes.

The prefix computation problem can be solved in O(n) time sequentially.

Work optimal algorithms are known for the prefix computation problem on many models of parallel computing.

present a CREW PRAM algorithm.

We emplyee the devide and conquer stoutegy to devise the prefix algorithm.

Let us the input be $x_1, x_2, ---, x_n$.

Without loss of generality assume that n is an intergoal power of 2.

We first present an n-processor and O(log n) time algorithm.

List Ranking:

List ranking plays a vital role in the P solution of several graph problems.

The i/p to the problem is a list given in the form of an array of nodes.

A node consist of some data and pointer to its right neighbor in the list.

The nodes need not occure in any order in the input.

The problem is to compute for each node in the list the number of nodes to its right.

called the rank of the node.

Since the data contained in any node is irrelevent to the list ranking problem,

We assume that each node contains only a pointer to its right neighbor.

The rightmost node's pointer field is zero.

PRAM Algorithm :

Parallel Random Access Machine Algorithm

Study algorithms for parallel machines (i.e., computers with more than one processor)

There are many applications in day-to-day life that demand real-time solutions to problems.

for example,

Weather forecasting has to be done in a timely fashion.

In the case of severe 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.

Programs written for such applications have to perform an enormous amount of computation.

In the ~~forecasting~~ Medical example, thousands of rules have to be tried.

In the forecasting example, large-sized matrices have to be operated on.

Even the fastest single-processor machines may not be able to come up with solutions within tolerable time limits.

Parallel Machines offer the potential of decreasing the solution times enormously.

MESH ALGORITHMS:

A mesh is an $a \times b$ grid.
In which there is a processor at each grid point.

The edge correspond to communication links and are bi directional.

Each processor of the mesh can be labeled with a tuple (i, j) where $1 \leq i \leq a$ and $1 \leq j \leq b$.

Every processor of the mesh is a RAM with some local memory.

Each processor can perform any of the basic operations such as addition, substraction, multiplicatio comparisions, local memory access and so on, in one unit of time.

The compatation is assummed to be synchronous;

There is a global clock and in every time unit each processor completes its intended task.

We consider only square meshes, for which $a = b$.

1) Data Concentration with mesh & Hypercube.

1) Data concentration Mesh:

- In a p-processor interconnection network assume that there are d<p data items distributed arbitrarily with at most one data item per processor.

- The problem data concentration is to move the data into the first d processors of the network one data item per processor, This problem also known as packing.

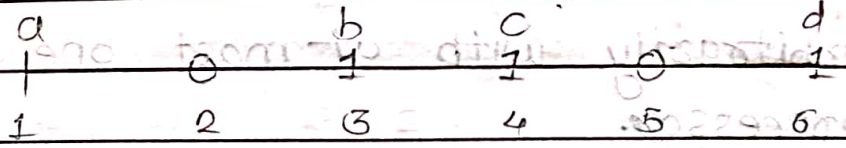
- In the case of a p-processor linear array we have to move the data item the processors 1, 2, 3, ------, d on a mesh, we might require the data items to move according to any indexing scheme of our choice. for example, the data could be moved into the first $(d/\sqrt{p})$ rows.

- Data concentration on any network is achieved by first performing a prefix computation to determine the destination of each packet & then routing the packet using an appropriate packet routing algorithm.

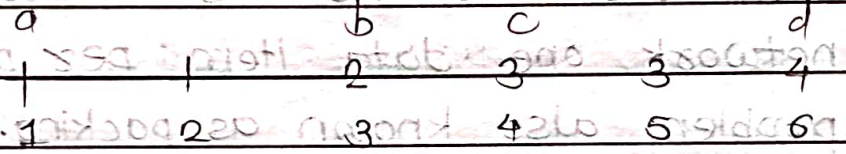
- let L be a p-processor linear array with d data item. To find the destination of each data item we make use of a variable x. If processor i has a data item. then it set $x_i = 1$ otherwise it set $x_i = 0$

Example consider six-processor linear array in there is an item in processor 1, 3, 4 & 6 then $(x_1, x_2, x_3, x_4, x_5, x_6) = (1, 0, 1, 1, 0, 1)$ &

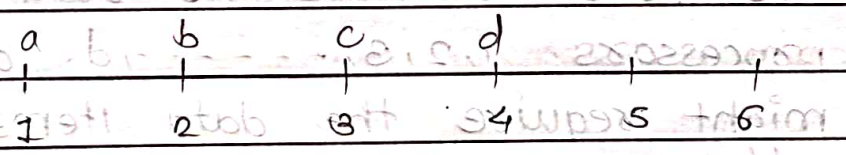
$(y_1, y_2, y_3, y_4, y_5, y_6) = (1, 1, 2, 3, 3, 4)$. So the items will be sent to the processors 1, 2, 3 & 4 as expected.



initial data location



Prefix computation



final data location

2) Describe & give example of prefix computational model with PRAM.

- Let ε be any domain in which the binary association operator $\oplus$ is defined. An operator $\oplus$ is said to be associative if for any three element x, y, & z from ε $((x \oplus y) \oplus z) = (x \oplus (y \oplus z))$; that is the order in which the operation $\oplus$ is performed does not matter.

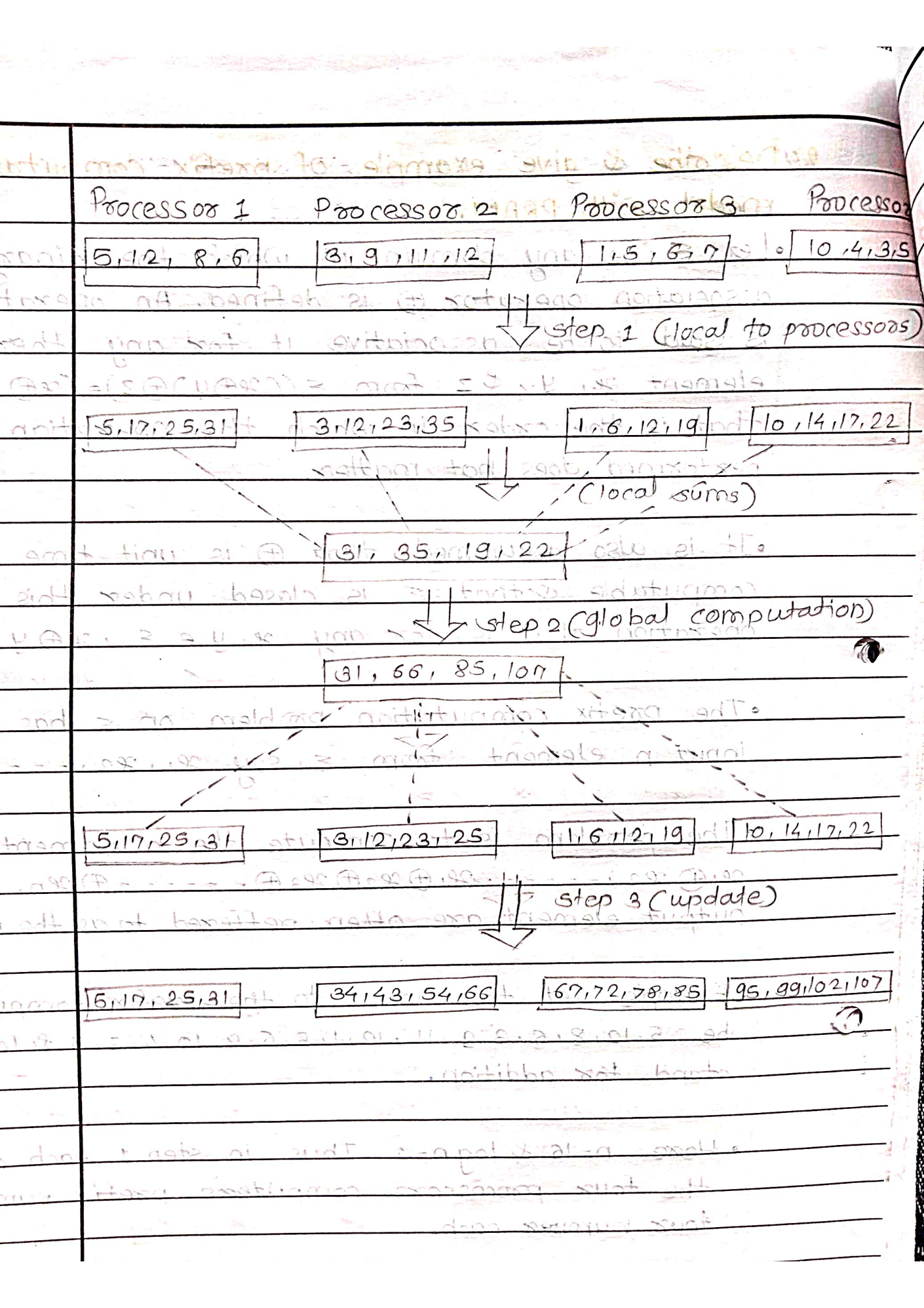
- It is also assumed that $\oplus$ is unit time computable & that ε is closed under this operation that is, for any $x, y \in \varepsilon$, $x \oplus y \in \varepsilon$.

- The prefix computation problem on ε has as input n element from ε, say $x_1, x_2, \ldots x_n$.

- The problem is to compute the n element x_1, $x_1 \oplus x_2, \ldots x_1 \oplus x_2 \oplus x_3 \oplus \ldots \oplus x_n$. The output element are often reffered to as the prefixes

- Example let the input to the prefix computation be 5, 12, 8, 6, 3, 9, 11, 12, 1, 5, 6, 7, 10, 4, 3, 5 & let $\oplus$ stand for addition.

- Here, $n=16$ & $\log n=4$. Thus in step 1 each of the four processors computers prefix sum on four number each.


Processor 1
Processor 2
Processor 3
Processor
5, 12, 8, 6
3, 9, 11, 12
1, 5, 6, 7
10, 4, 3, 5
step 1 (local to processors)
5, 17, 25, 31
3, 12, 23, 35
1, 6, 12, 19
10, 14, 17, 22
(local sums)
31, 35, 19, 22
step 2 (global computation)
31, 66, 85, 107
5, 17, 25, 31
3, 12, 23, 25
1, 6, 12, 19
10, 14, 17, 22
step 3 (update)
5, 17, 25, 31
34, 43, 54, 66
67, 72, 78, 85
95, 99, 102, 107

Q) Describe Broadcasting on MESH with an example.

- The problem of broadcasting in an interconnection network is to send a copy of a message that originates from a particular processor to a specified subset of other processors.
- Unless otherwise specified this subset is assumed to consist of every other processor.
- Broadcasting is a primitive form of interprocessor communication & is widely used in the design of several algorithm.
- Let C be a linear array with the processors 1, 2, - - - P Also let m be a message that originats from processor 1. Message m can be broadcast to every other processor as follows.
- Node 1 send a copy of m to processor 2, which in turn forward a copy to processor 3 & so on. This algorithm takes P-1 steps & this runtime is the best possible.

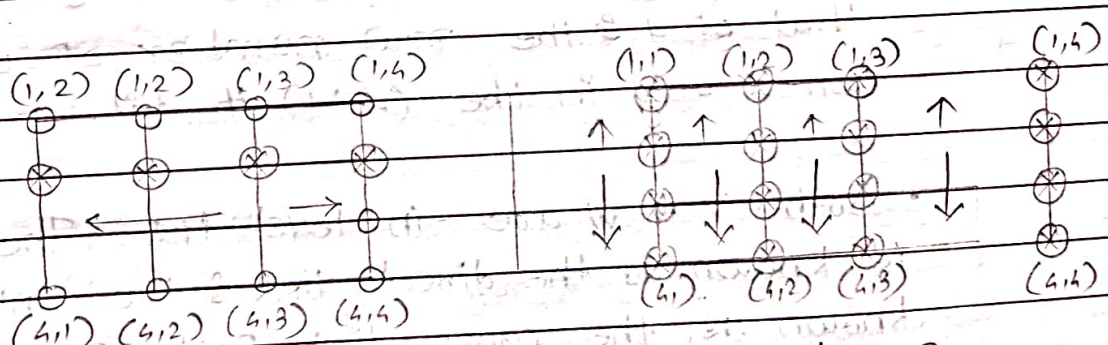


Fig: Broadcasting in a mesh.

4) Short note on Butterfly Network ?

- The butterfly network is closely related to the hypercube, Algorithm designed for the butterfly can sily adapted for the hypercube & vice-versa.

- In fact, for several problem it is easier to develop algorithm for the butterfly & then adapt them to the hypercube.

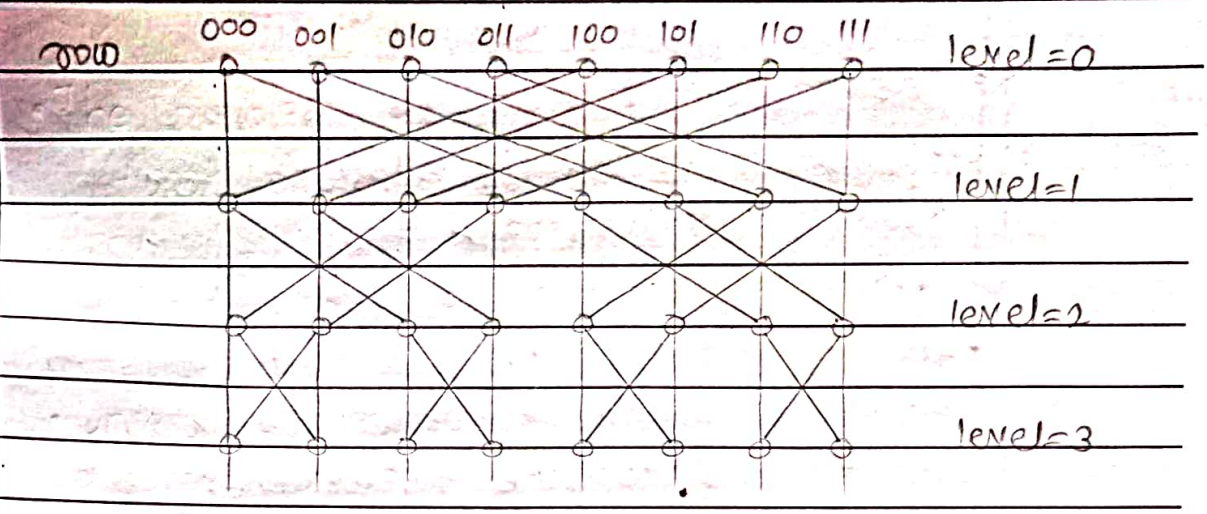
- A d-dimensional butterfly, denoted Bd, has $P=(d+1)2^d$ processors & $d2^{d+1}$ link Each processor in Bd can be represented as a tuple $\langle r,l \rangle$ Where $0 \leq r \leq 2^d-1$ & $0 \leq l \leq d$.

- The variable r is called the row of the processor & l is called the level of the processor.

- A processor $u=\langle u,l \rangle$ in Bd is connected to two processor in level $l+1$ (for $0 \leq l < d$).

- These two processor are $v=\langle r, l+1 \rangle$ & $w=\langle r^{(l+1)}, l+1 \rangle$. The row number of v is the same as that of u & the row number of w differs from r only in the $(l+1)$ th bit.

- Both v & w are in level $l+1$. The link (u,v) is known as the direct link & the link (u,w) is known as the cross link. Both of these link are called level $(l+1)$ links.



A three dimensional butterfly

- In above fig the processor <011,1> is connected to the processors <011,2> & <001,2>. Since each processor is connected to at most four other processor, the degree of Bd (for any d) is four & hence is independent of the size (p) of the network.

- If u is any processor in level 0 & v is any process or in level d, there is a unique path between u & v of length d.

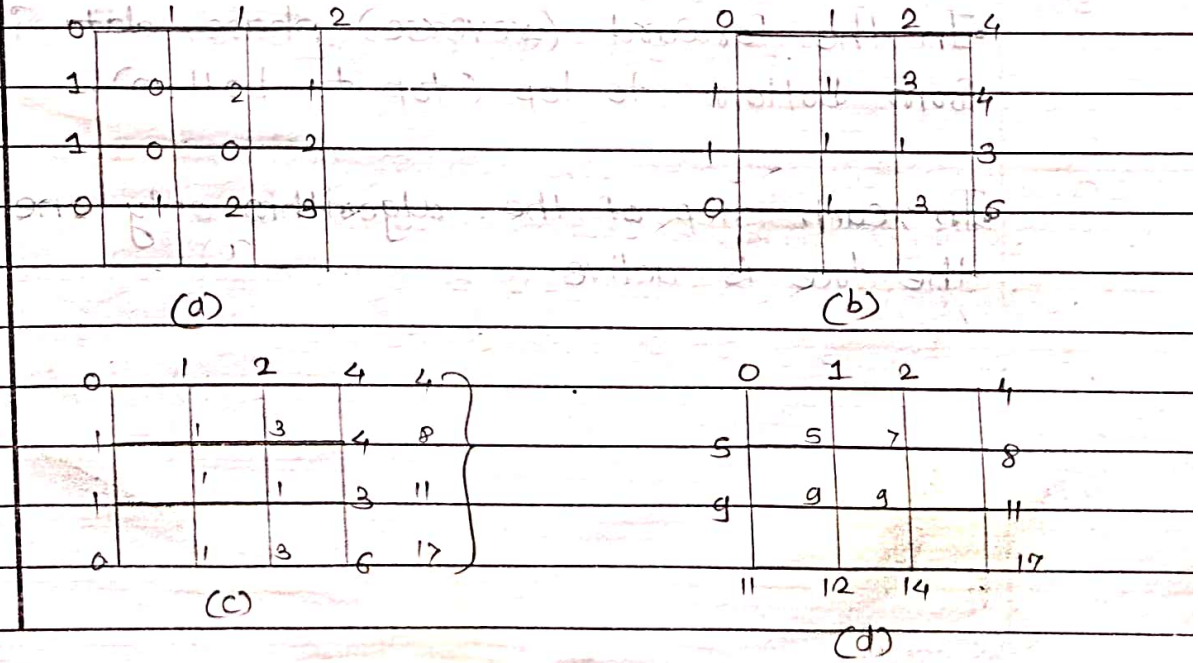
- let $u=\langle r,0\rangle$ & $v=\langle r',d\rangle$. The unique path is $\langle r,0\rangle, \langle r,1\rangle, \langle r_2,2\rangle, \ldots\ldots \langle r',d\rangle$ where r_1 has the same first bit as r', r_2 has the same first & second bits as r'. & so on.

- Any algorithm that runs on Bd is said to be a normal butterfly algorithm if any given time, processors in only one level participate in the computation.

Q) Explain prefix sum computation with the help of mesh & hypercube.

Prefix Computation Mesh:

- Let $\mathcal{E}$ be any domain in which the binary associative unit time computable operator $\oplus$ is defined. Recall that the prefix computation problem on $\mathcal{E}$ has as input n element from $\mathcal{E}$, say, $x_1, x_2, \ldots, x_n$.
- The problem is to compute the n elements $x_1, x_1 \oplus x_2, \ldots, x_1 \oplus x_2 \oplus x_3 \oplus \cdots \oplus x_n$.
- The output element are often referred to as the prefixes for simplicity, we refer to the operation $\oplus$ as addition.
- Consider the data on the 4X4 mesh & the problem of prefix sums under the row major indexing scheme. In phase 1, each row computes its prefix sums. In phase 2, prefix sums are computed only in the fourth column. finally in phase 3 the prefix sum are updated.



① Prefix computation on a mesh:

- Phase 1: Row i (for $i = 1, 2, 3 \ldots \sqrt{p}$) computes the prefixes of its $\sqrt{p}$ element. At the end, the processor (i, j) has $y(i,j) = \sum_{q=1}^{j} x(i,q)$.

- Phase 2: Only column $\sqrt{p}$ computes prefixes of sums compute in phase 1. Thus at end, processor $(i, \sqrt{p})$ has $z(i, \sqrt{p}) = \sum_{q=1}^{i} y(q, \sqrt{p})$. After the computation of prefixes (shift them down by one processor i.e. have processor $(i, \sqrt{p})$ send $z(i, \sqrt{p})$ to processor $(i+1, \sqrt{p})$ (for $i = 1, 2 \ldots, \sqrt{p}-1$).

- Phase 3: Broadcast $z(i, \sqrt{p})$ in row $i+1$ (for $i = 1, 2 \ldots \sqrt{p}-1$). Node j in row $i+1$ finally updates its result to $z(i, \sqrt{p}) \oplus y(i+1, j)$.

② Prefix Computation hypercube:

- We again make use of the binary tree embedding to perform prefix computation on Hd. Let x_i be input at the i^{th} leaf of a 2^d-leaf binary tree. There are two phase in the algorithm, namely, the forward phase and the reverse phase.

- In the forward (reverse) phase, data items flow from bottom to top (top to bottom).

- In each step of the algorithm only one level of the tree is active.

forward Phase:

- The leaves starts by sending their data up to their parents. Each interval processor on receipt of two items (say y from its left child & z from its right child) computes $w = y \oplus z$, stores a copy of y & w to its parrent.

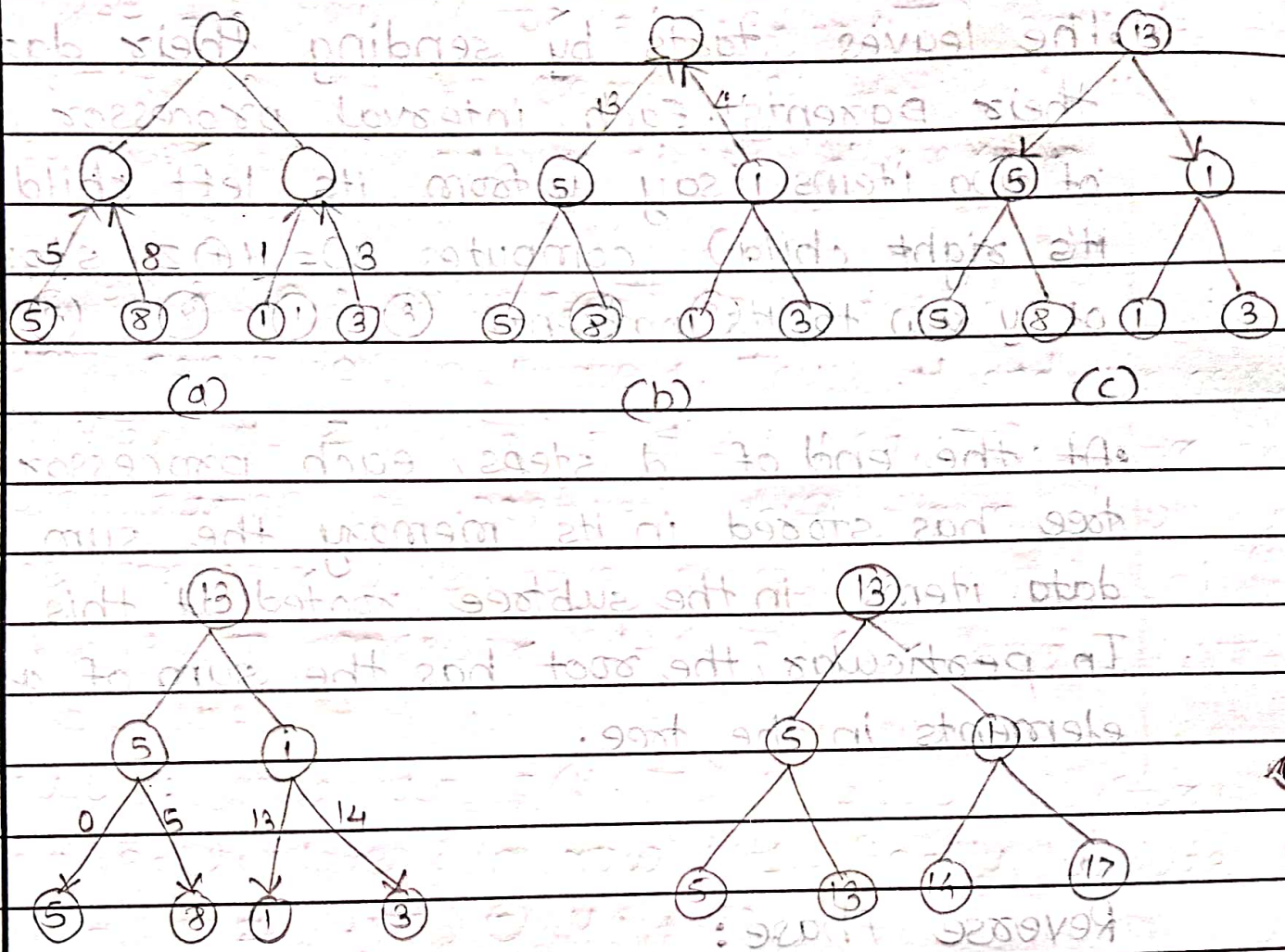
- At the end of d steps, each processor in the tree has stored in its memory the sum of all the data items in the subtree rooted at this processor. In particular, the root has the sum of all the elements in the tree.

Reverse Phase:

- The root starts by sending zero to its left child & its y to its right child. Each internal processor on receipt of a datum (say q) from its parrent sends q to its left child & $q \oplus r$ to its right child. When the ith leaf gets a datum q from its parent, it computes $q \oplus x_i$ & stores it as the final result.

- Let $\mathcal{E}$ be the set of all integers & $\oplus$ be the usual addition. Consider a four-leaf binary tree with the following input 5, 8, 1, 3.

- In step 1, the leaves send their data up, In step 2, the internal processors send 13 & 4, respectively, storing 5 & 1 as their y-values. In step 3 the root & send 0 to the left & 13 to the right.



Prefix Computation on a binary tree.