Unit-4

(AC-handwritten Note-1)
Page-76.

Date
Page

* Mesh sorting. Sheer sorting. (Numerical)
* Mesh algo for max. selection with n^2 process.
* Mesh algo for max. selection with n processor will it work optimal? When it will be optimal.
* Prefix computation on Mesh.
* Broadcasting problem on Mesh.
* Packet routing on Mesh. Demonstrate broadcasting on Mesh

* Explain embedding of networks.
* Explain process of embedding binary tree on hypercube and calculate Expansion, Dilation and Congestion.
* ~~Odd-even merge~~ Data concentration on hypercube.
* Explain hypercube. features of hypercube
* algo to solve prefix computation on hypercube.
* algo to solve broadcasting problem of hypercube.
*

* Process of Creating butterfly network.
* Prefix computation in a butterfly network
* Odd-even merge sort in butterfly net. (time complexity)
*

Butterfly network →

Mesh Algorithm

→ A mesh is a axb grid where there is a process at each grid point. The edges correspond to communication links and are bidirectional.

→ Each processor in a mesh can be labelled with a tuple (i,j) where $1 \le i \le a$ and $1 \le j \le b$.

→ Every processor of the Mesh is a RAM with same local Memory. Hence each processor can perform any of the basic operation like add, subtract, multiply, compare, etc.

→ The computation is assumed to be synchronous.

→ A closely related model for Mesh is linear array consist of p processor named (1,2,---p)

1. process i is connected to i-1 & i+1 for $2 \le i \le p-1$.
2. Processor 1 is connected to 2 and p is connected to p-1.

→ Processor 1 and p are called boundary processor.

→ Processor i-1 and i+1 are called left and right neighbour of i.

→ We consider square Mesh for a=b as shown in below fig

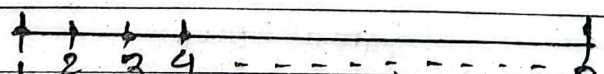


fig: linear array

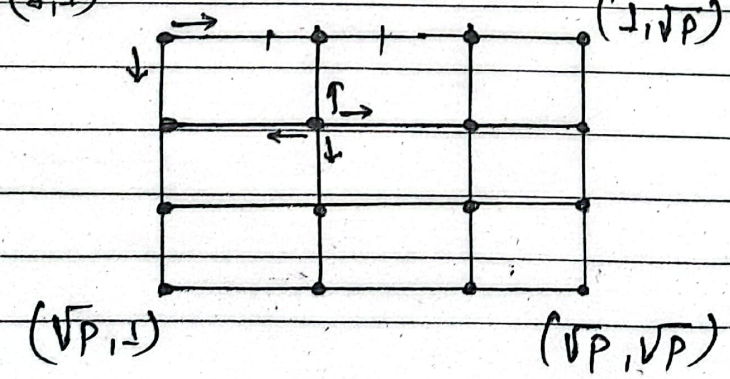


fig: Mesh.

Broadcasting In Mesh:

→ In a case of a $\sqrt{p} \cdot \sqrt{p}$ Mesh broadcasting can be done in two phases:

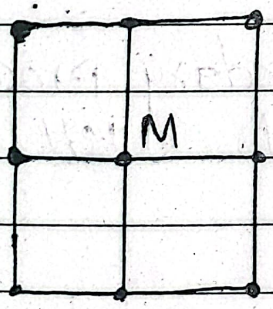
1. If (i,j) is the processor of Message (M) origin the.
   1. M could be broadcast to all the ~~proje~~ processor in row i.
   2. broadcasting of M is done in each column.

It takes

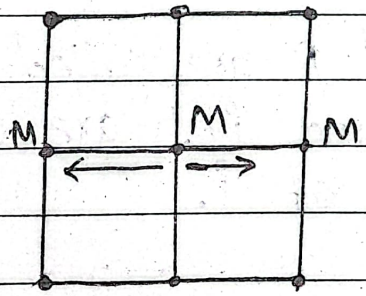
$2(p-1)$ steps $= O(p)$.

Example Consider a 3×3 mesh with Message M originating at (2,2) then broadcast is done at

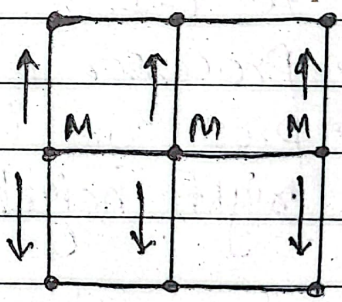
1. broadcast to node of row 2.
2. broadcast to node at each column.



(i) Message originates.



(ii) Message broadcast at same row.



(iii) Message broadcast at each column.

Packet Routing in Mesh:

→ mesh is mxn grid, → each grid point has processor.
→ communication link are bi-directional.
→ each process can be labelled by tuple (i,j),
→ each process has local memory.
→ mesh has global clock for synchronization.

→ Packet routing is a primitive inter-process communication operation.
→ each packet has its origin an destination.
→ In a Mesh of pxp, q can be an arbitrary packet with (i,j) as origin and (u,v) as destination.
→ Then packet can travel in two steps:
1. Travel along j (column) to row u.
2. Travel along row u to destination (u,v)

→ If a packet with origin (1,1) has destination (P,P) then:
- Step 1: is done in (P-1) step and
Step 2 also take (P-1) Step
Hence the lower bound of the worst case routing time of any algorithm is:

$$(P-1) + (P-1) = 2P-2 = 2(P-1).$$

However, if all the packets originated at col 1 are destined for row P/2 then processor (P/2, 1) gets two packet at every time step and it can sent only one, the other has to wait.
∴ Hence queue size is needed as large as P/2.

Prefix Computation on Mesh

→ Consider a $\sqrt{p} * \sqrt{p}$ Mesh.

→ prefix computation can be done in following steps:

Step 1: Prefix computation row-wise.
Row i (for $i = 1, \ldots \sqrt{p}$) compute prefix of p elements.

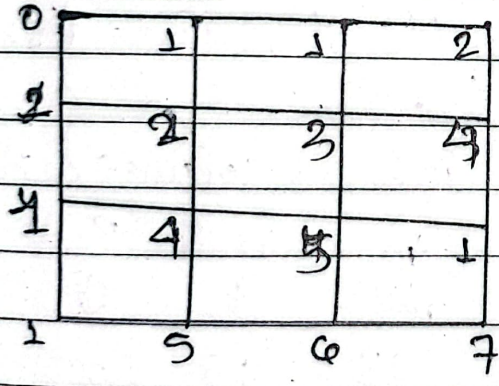
Step 2: prefix computation column wise. last column only.
only $\sqrt{p}$ column computes the prefix of sum computed on step 1 and store on local memory.

step 3: shift 1 down to column wise. last column only.
Shifting :→ $(i, \sqrt{p}) \longrightarrow (i+1, \sqrt{p})$

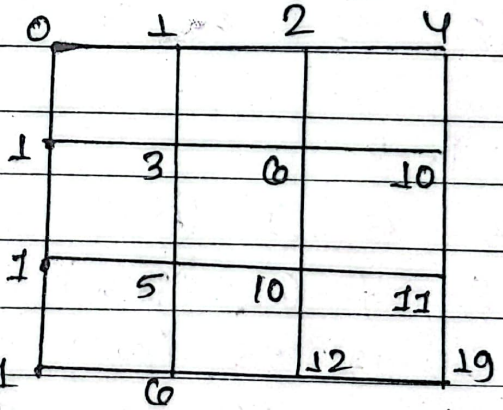
step 4: Broadcast row wise to each processor.
~~Broadcast~~

{0, 1, 1, 2, 1, 2, 3, 4, 1, 4, 5, 1, 1, 5, 6, 7}.

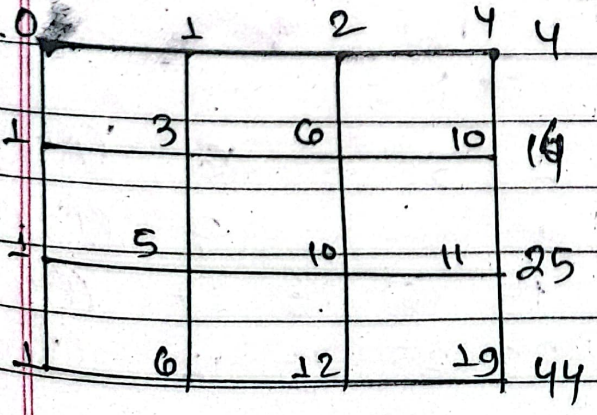
~~ste~~ Mesh:



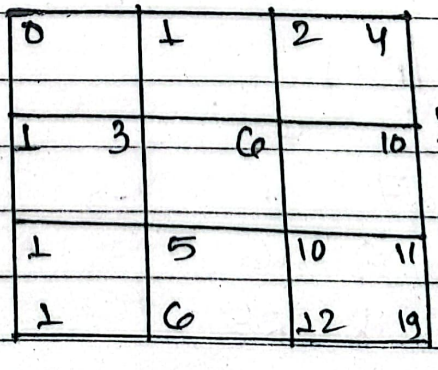
Step 1:



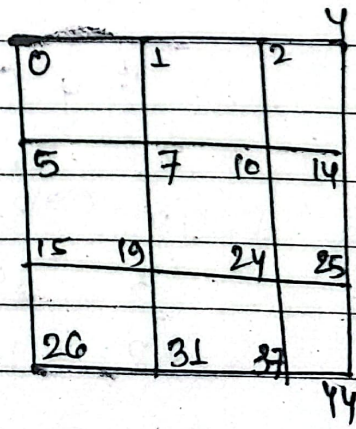
Step 2:



Step 3:



Step: 4



Hence: {0, 1, 2, 4, 5, 7, 10, 14, 15, 19, 24, 25, 26, 31, 37, 44} #

Sheer Sort (Mesh Sorting)

Algorithm:

```
for (i=1; i≤logp+1 ; i++)
{
    if i is even
    Sort the column (top to bottom)
    else
    Sort the row (alternative in order) snake.
```

Example:

15	12	8	32
7	13	6	17
2	16	19	25
18	11	5	3

i=1.

8	12	15	32	(Asc)
17	13	7	6	(Des)
2	16	19	25	(Asc)
18	11	5	3	(Desc)

↓

i=2 (colwise / Asc)

2	11	5	3
8	12	7	6
17	13	15	25
18	16	19	32

i=3

2	3	5	11	(Asc)
12	8	7	6	(Desc)
13	15	17	25	(Asc) ⇐
32	19	18	16	(Desc)

↓

i=4 (col / Asc)

2	3	5	6
12	8	7	11
13	15	17	16
32	19	18	25

⇒

i=5

2	3	5	6	(Asc)
12	11	8	7	(Desc)
13	15	16	17	(Asc)
32	25	19	18	(Desc)

#

Complexity: $O(\sqrt{p} \log p)$.

Mesh Algorithm for Maximum selection with 'n' processors: Will it work optimal ?.

The Mesh algorithm for maximum selection with n-processor is a randomized algorithm,
The steps involved are:

1. The randomized algorithm chooses a random sample S, from x of size $n^{1-\epsilon}$ where $\epsilon = 0.6$ is sufficient.

2. Two elements l_1 and l_2 are identified from S, l_1 & l_2 are also known as Splitter.

3. The keys l_1 & l_2 are such that the element to be selected has a value between l_1 & l_2 with high Probability and the number of keys in range $[l_1, l_2]$ is small.

4. Now, Partition x into x_1, x_2, x_3 where
   1. $x_1 = \{ x \in x \mid x < l_1 \}$
   3. $x_3 = \{ x \in x \mid x > l_2 \}$
   2. $x_2 = \{ x \in x \mid l_1 \leq x \leq l_2 \}$.

5. We also count $|x_1|, |x_2|, |x_3|$, and if $|x_1| < i \leq |x_1| + |x_2|$ the element to be selected lies in x_2, if this is true we proceed further or else we start all over again.

6. The element to be selected will be $(i - |x_1|)^{th}$ smallest element of x_2:

→ The above process of sampling and elimination is repeated until the appropriate number of remaining keys is $\leq n^{0.4}$.

→ After this we perform an appropriate selection from out of the remaining keys using sparse enumeration sort.

→ Analysis:
- first step takes $O(1)$ time on Mesh.
- prefix computation takes $O(\sqrt{p})$ time.
- concentration and sparse enumeration sort takes $O(\sqrt{p})$ time.

Hence, selection from $n=p$ keys can be performed in $O(\sqrt{p})$ time on $\sqrt{p} * \sqrt{p}$ mesh.

Embedding of network:

→ Mapping of one network to another is called embedding for example network such as ring, mesh and binary tree can be shown to be subgraph of hypercube

→ let $G(V_1, E_1)$ and $H(V_2, E_2)$ are any two connected networks, an embedding of G to H is mapping of V_1 to V_2.

→ Embedding are important,
to simulate one network on another, using an embedding of G to H, an algorithm designed for G can be simulated on H.

- following terminologies are used in embedding:

1. Expansion:
   It is obtained as V_2/V_1.

2. Dilation:
   length of longest path any link of one network structure is mapped with another.

3. Congestion:
   Number of path on the H_y corresponding to the links of G that it is on.
   (max. no. of times an individual path of S_2 is traversed for mapping S_1).

Example: Embedding of binary tree: Prefix Computation

We can embed a binary-tree to hypercube in many ways.

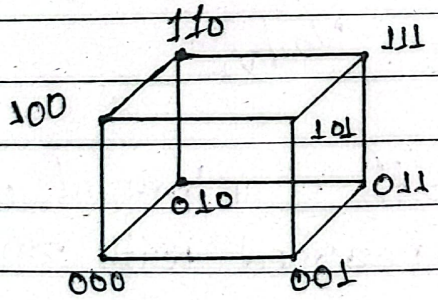
→ A p-leaf binary tree, whose $p = 2^d$ can be embedded into H_d.

→ A full binary tree with d-levels has $(2^{d+1} - 1)$ processor however d-dimension hypercube has 2^d processor

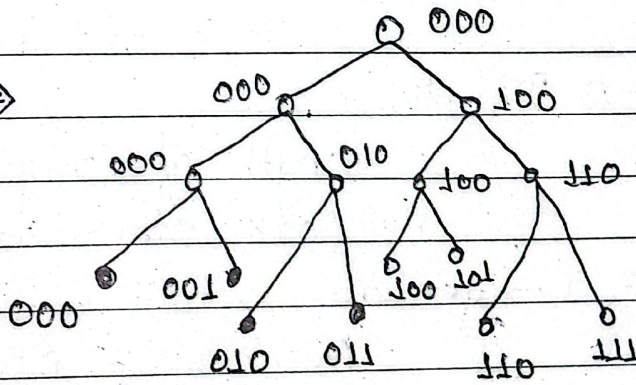
→ Hence, more than one Processor of binary tree has to be mapped to a single Processor of hypercube.

→ If tree leaves are 0, 1, 2, ---- P-1 then leaf is mapped to i^{th} processor of H_d.

→ Each processor of T is mapped to the same processor of H_d as its leafmost descendant leaf.



⇒



Embedding of Hypercube.

1. Expansion: $\frac{15}{8}$ → node on binary tree → node on hypercube

2. Dilation: 5

3. Congestion: 4

Hypercube:

Data Concentration on hypercube:

⇒ A hypercube is a d-dimensional structure, numbered using d-bits, Hence there are 2^d processor in d-dimensional hypercube, represented as H_d.

→ A hypercube has several variants like butterfly, shuffle-exchange network and cube connected cycle

→ Algorithm for hypercube can be adapted for butterfly network and vice-versa. Hence algorithm for butterfly network can be ~~by~~ applied to hypercube as well.

Data Concentration on hypercube:

→ on a hypercube H_d assume there are $k<p$ data item distributed arbitrarily with atmost one datum per processor. Then the problem of data concentration is to move the data into processor $0, 1, \ldots k-1$, of H_d

→ for this, we map the given hypercube to a butterfly network and then apply the concentration algorithm for butterfly network.

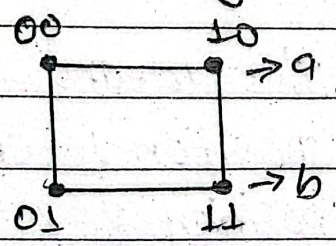


fig: a 2-D hypercube.

Consider a 2-d hypercube that has two data items at 10 & 11 processor be a & b respectively.

Then we map the 2-D hypercube to 2-d butterfly net.

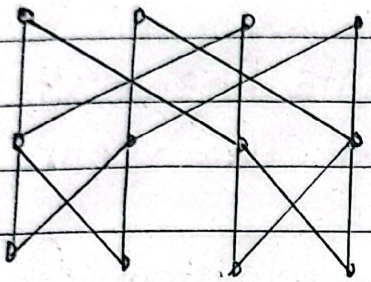


fig: mapping 2-D hypercube to 2-d butterfly network.

Step 1: The first step in data concentration is to compute prefix sum and to do so, we map the butterfly network to a binary tree of depth 2.

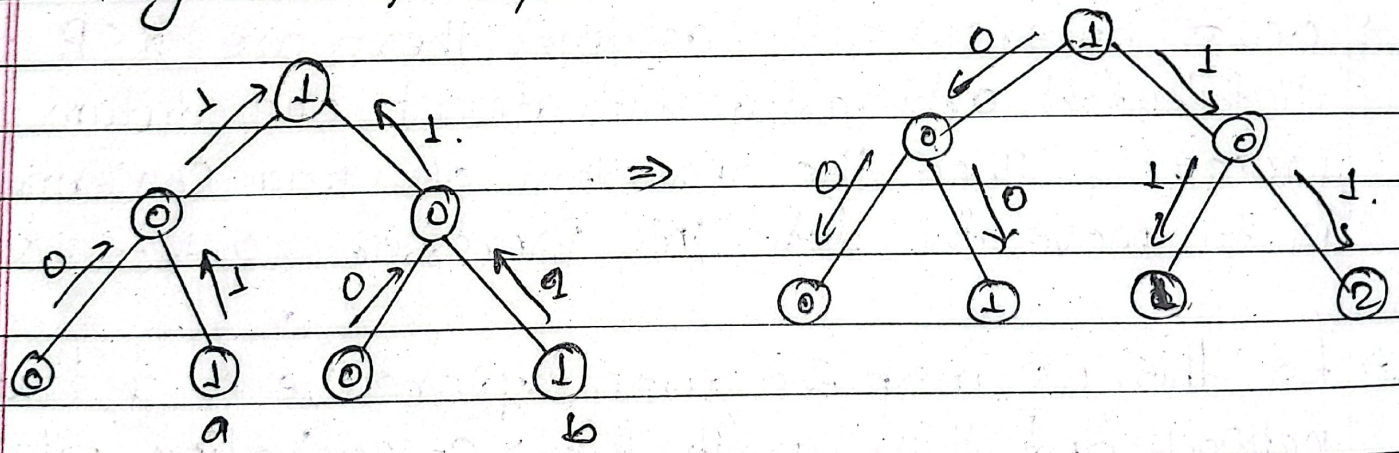
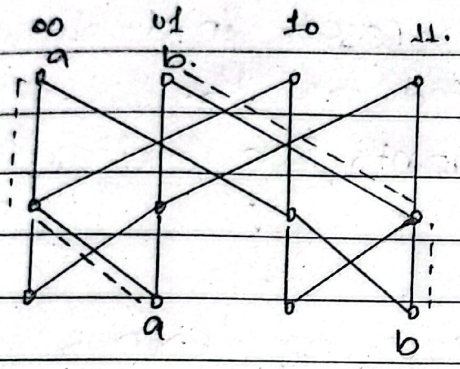


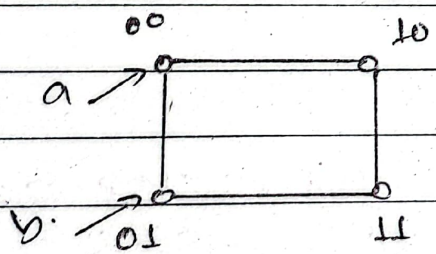
fig: Prefix Computation on binary tree

The prefix sum is {0, 1, 1, 2}. This means that the element a must be moved to position 1 and the element b must be moved to position 2.

In second phase, using the graph greedy approach the packets are routed to their destination. The path followed is shown dashed line.



finally we map the butterfly network to hypercube.



Hypercube.
features of hypercube.

→ A hypercube is a d-dimensional structure, numbered using d-bits. Hence there are 2^d processor in d-dimensional hypercube. Which is represented as H_d.
→ a hypercube may have variants like butterfly network, shuffle-exchange network, etc.

→ we define hypercube as following terms:
1. Dimension.
2. Coding of Processor.
3. Hamming Distance
4. Diameter.

Example: for d = 3 i.e. dimension = 3
no. of vertex = $2^d = 2^3 = 8$
no. of Processor = 8.

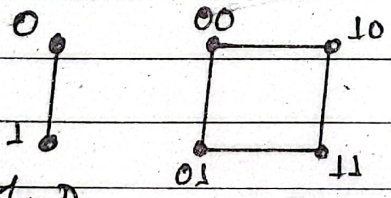
→ Coding of Processor = 000, 001, 010, 011, 110, 100, 101, 111,
→ Coding should be carried out in such a way that the difference of code of two adjacent Processor should be 1, which is called hamming distance.
→ The diameter of d-dimensional hypercube is d.
→ The bisection width of d-dimensional hypercube is $2^{(d-1)}$.
→ Hypercube is highly scalable architecture. Two d-dimensional hypercube can be easily combined to form a d+1- dimensional hypercube.

Characteristic of hypercube:

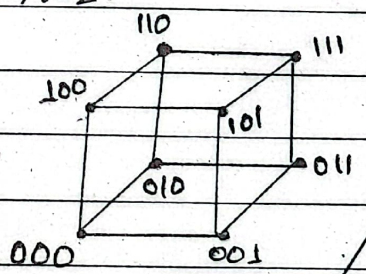
→ each processor code should alter by 1 bit only.
→ Degree of H_d = No. of interconnected processor of reference processor.
→ Processor coding is calculated by humming distance with no. of position change.
→ Sequential :
at one unit time any process can only communicate to one of it's neighbour.
→ Parallel :
at one unit time a processor can communicate with all of its neighbour.

features of hypercube:

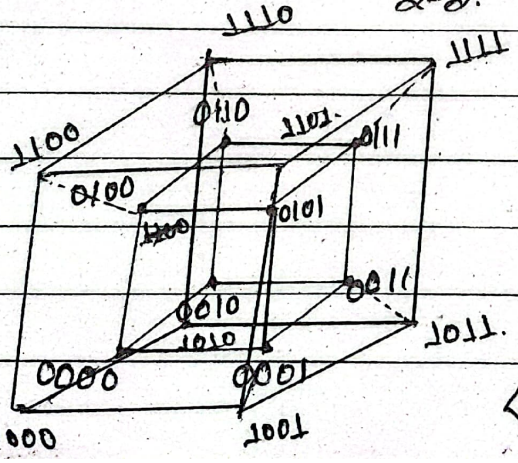
→ The diameter of hypercube is $\log p$.
→ A hypercube can be generated by combining different hypercubes: $H_d = H_{d_1} \times H_{d_2}$.



1-D. 2-D.



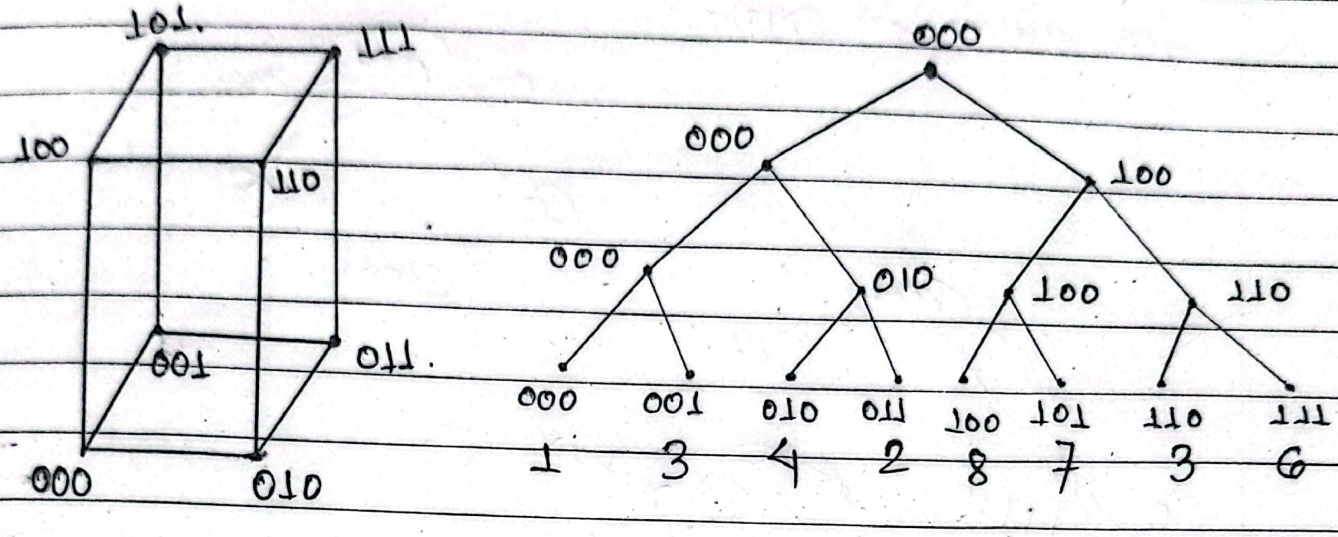
3-D.



4-D.

0	000	1	000
0	001	1	001
0	010	1	010
0	011	1	011
0	100	1	100
0	101	1	101
0	110	1	110
0	111	1	111

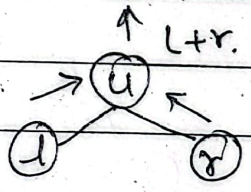
Prefix Computation on hypercube:



Phase I :

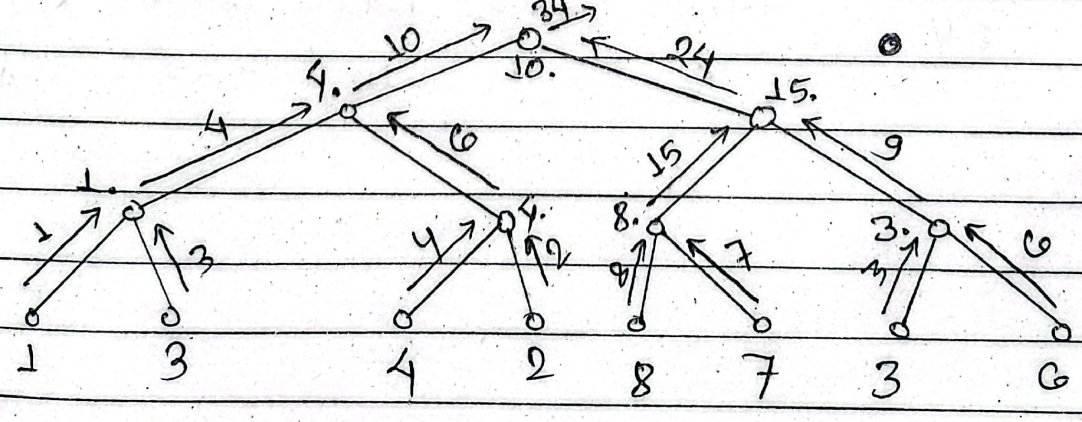
1: Place the values of the H_d to binary tree by mapping its pattern.

2. The leaf node will pass the value to the upper (u) (l,r)



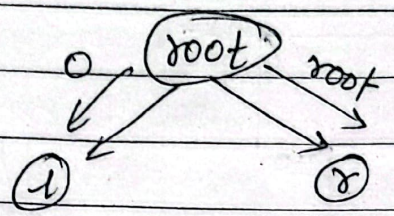
3. 'u' node will transfer the 'l+r' value to ~~its~~ the upper node. and store the received ~~value~~ left value.

4. The process will continue recursively until it reaches the root node.

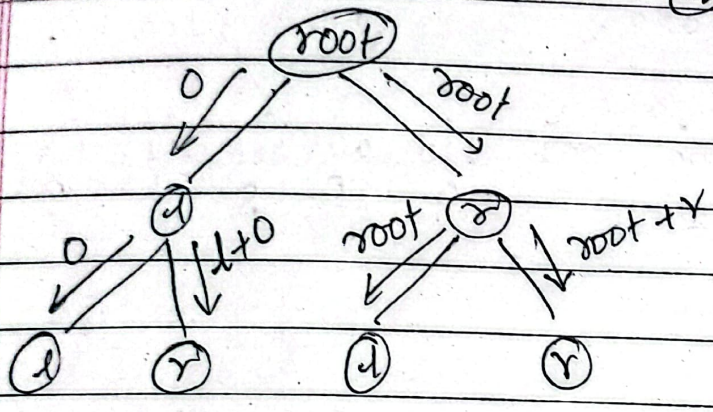


Phase II:

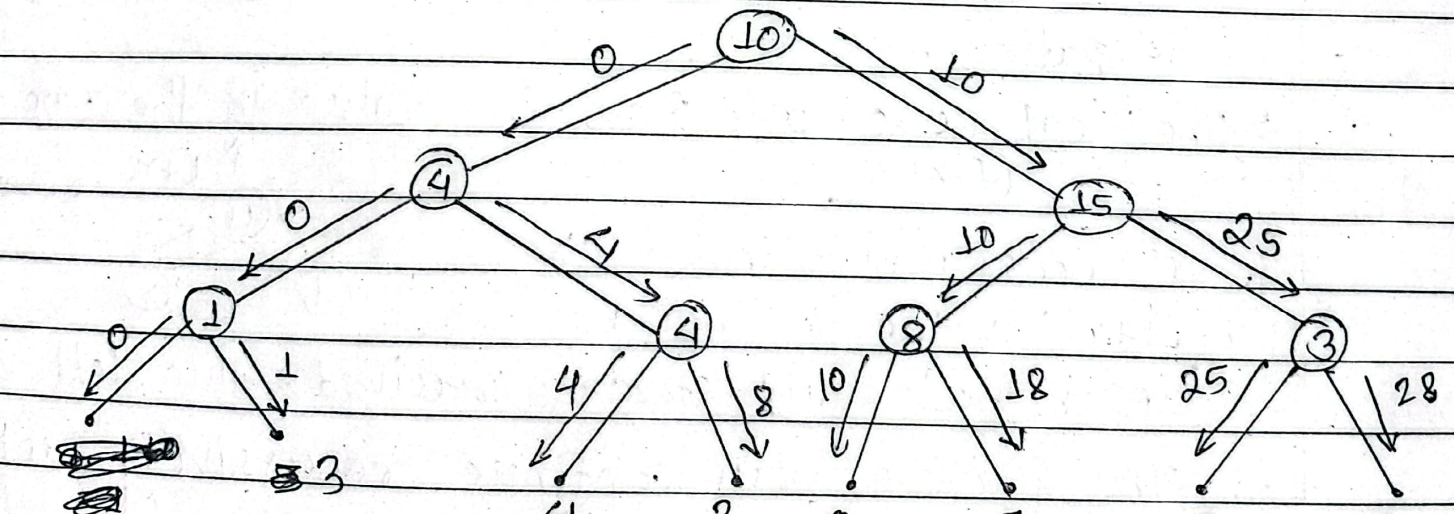
1. Starting from the root node it transfer 0 to left and it's value to right.



Step1

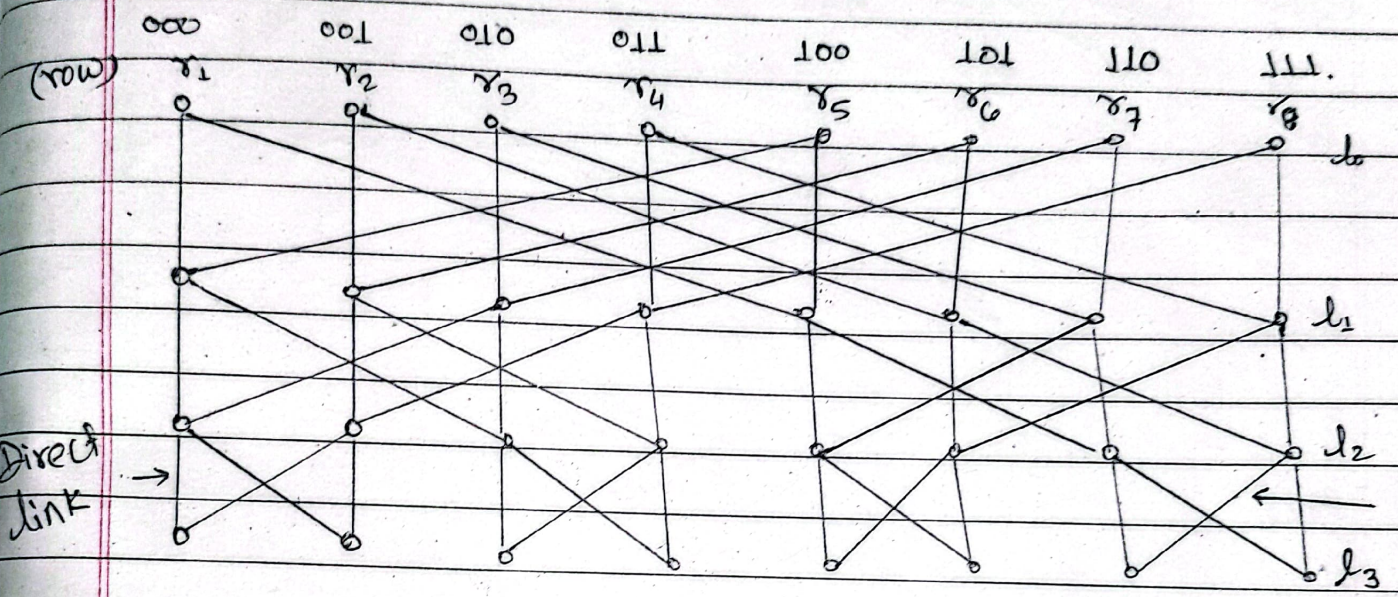


#



Butterfly network: (B_d)

→ B_d is also one of the embedding technique to solve problems of Sorting data connection, prefin, selection, etc. In a massive parallel connected Network.



3d- butterfly network

1) Vertical line.
2) cross link
3) Process (p) = 32 → $(d+1)2^d = 4 \cdot 2^3 = 32$.

total links = $d.2^{d+1}$

$3 \times 2^{3+1} = 3 \times 2^4 = 3 \times 16 = 48$

→ A processor u is connected to v and w; where;

$v = \langle r, l+1 \rangle$ → direct connection.

$w = \langle \quad , l+1 \rangle$ → cross connection.

$u = \langle r, l \rangle$

$0 \leq r \leq 2^d - 1.$

$0 \leq l \leq d.$

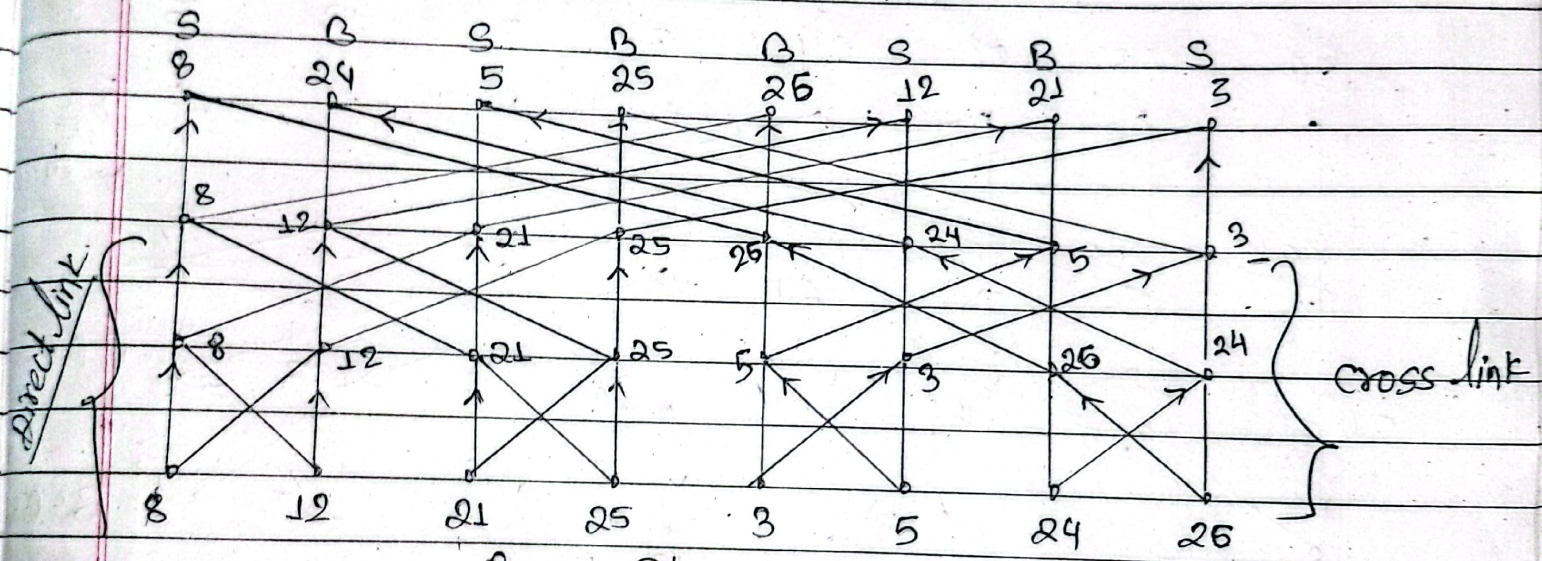
Odd-even merge Sort (Butterfly Network)

→ It consist of two phases.

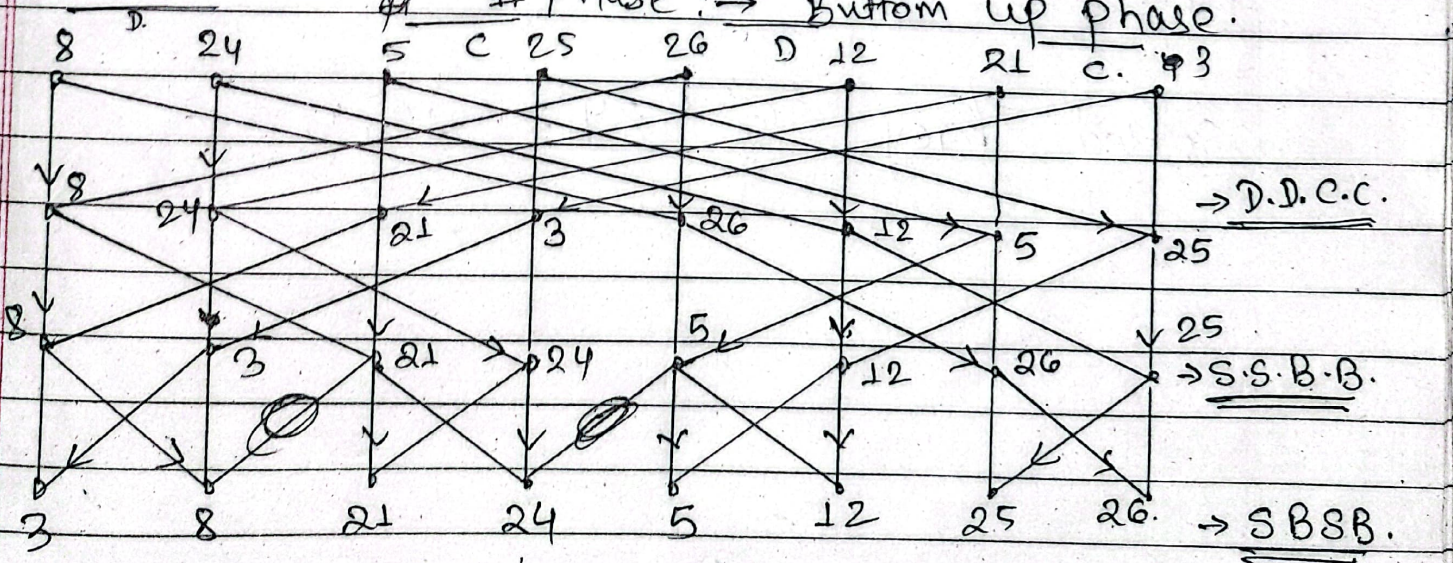
Phase I: separate odd and even into
[x_1 & x_2 given Sequence to O_1, E_1, O_2, E_2]

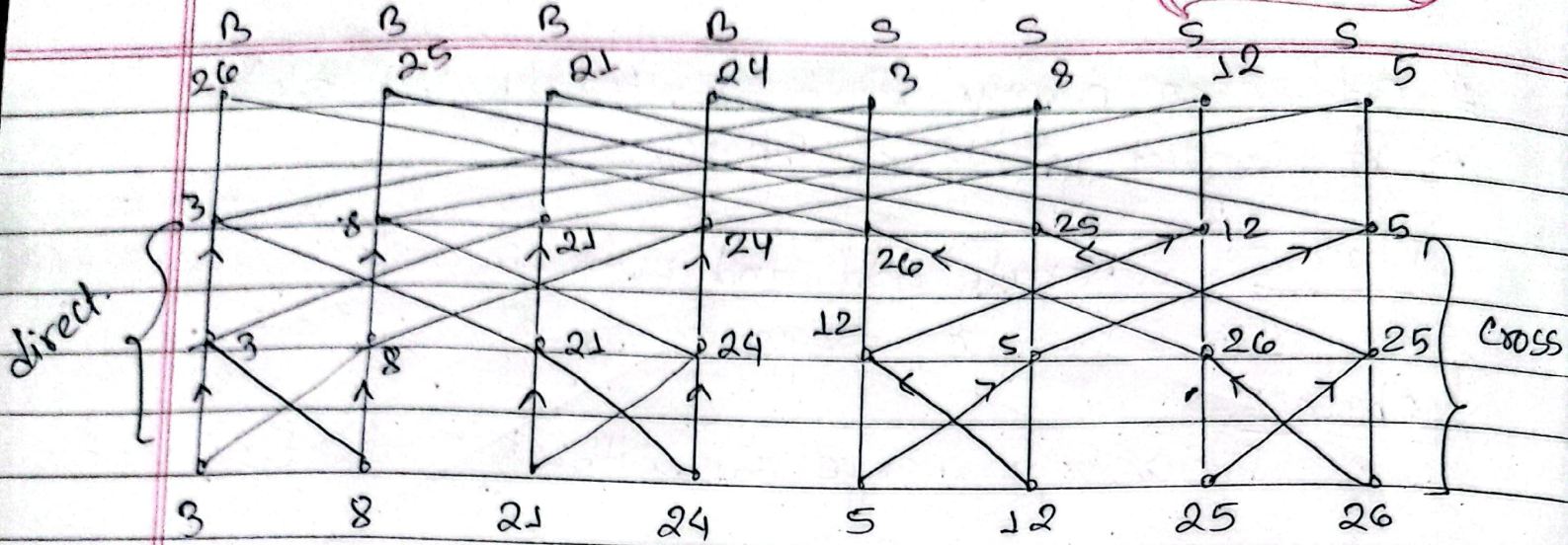
Phase II:
Recursively merge O_1 with O_2 and E_1 with E_2

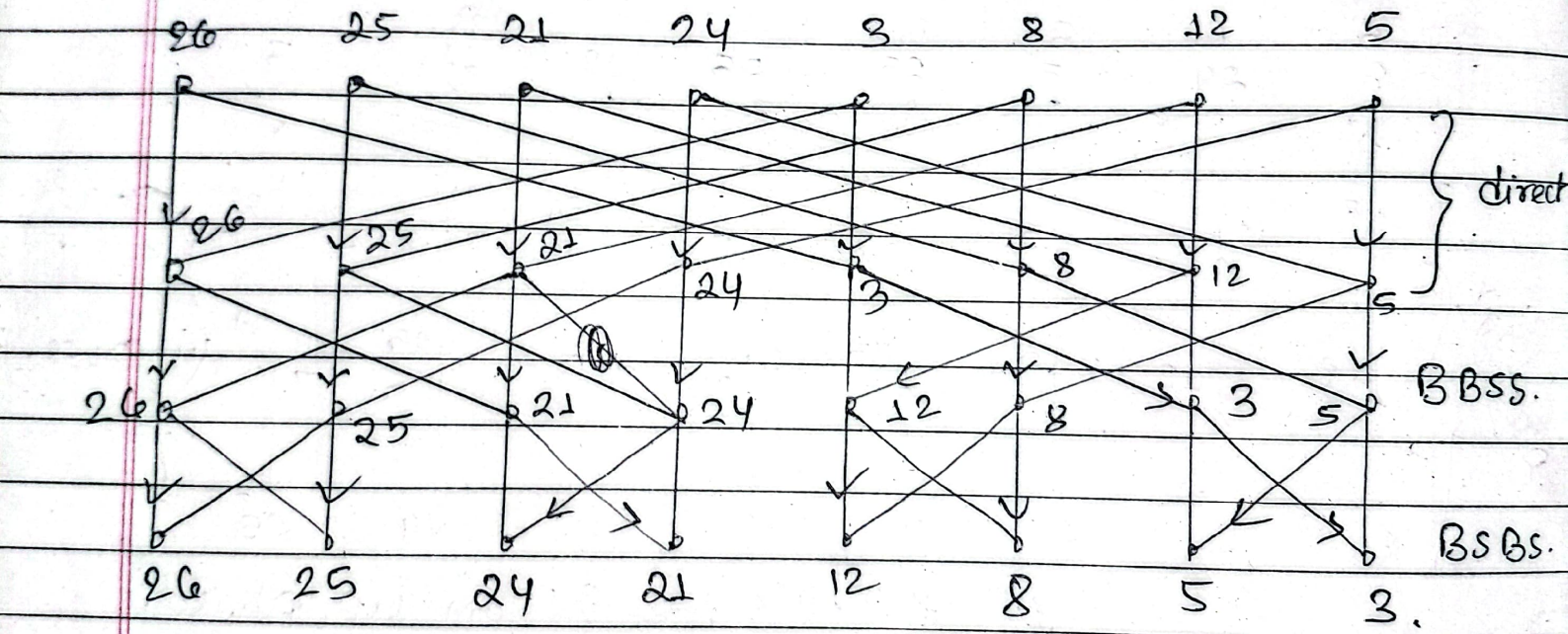
Question = 8, 12, 21, 25, 3, 5, 24, 26.



Phase :→ Buttom up phase.







Sorted Array : 26, 25, 24, 21, 12, 8, 5, 3

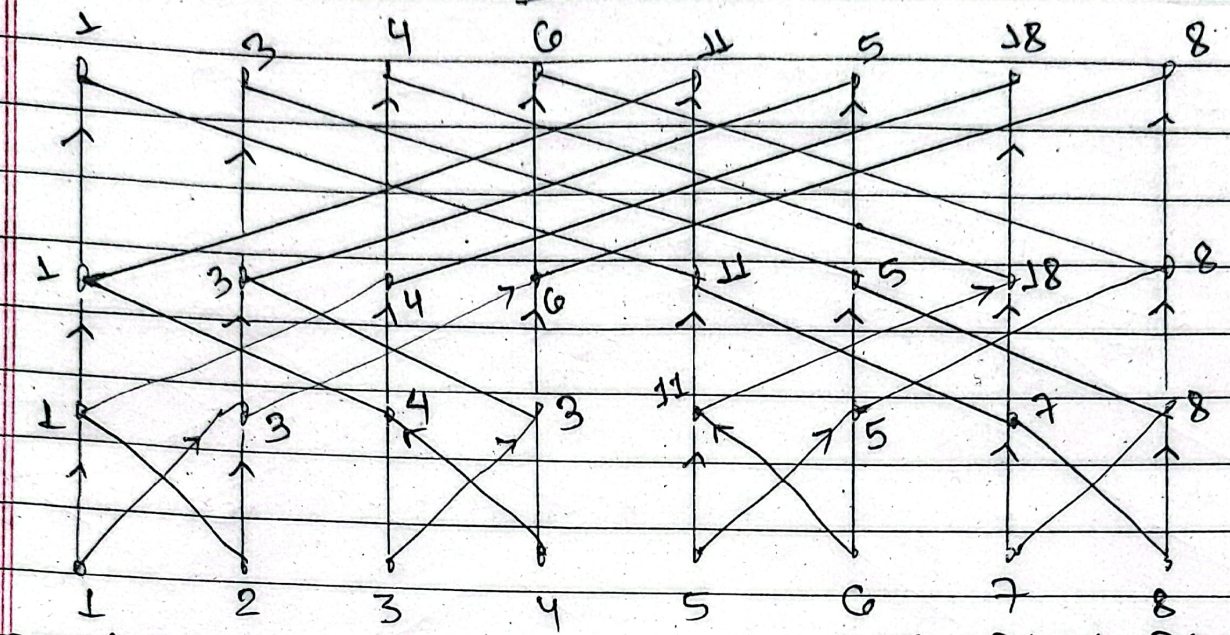
Prefix Computation

Phase I

Step I: Direct, Sum, Cross, Cross, Sum, Cross, Direct, Direct.

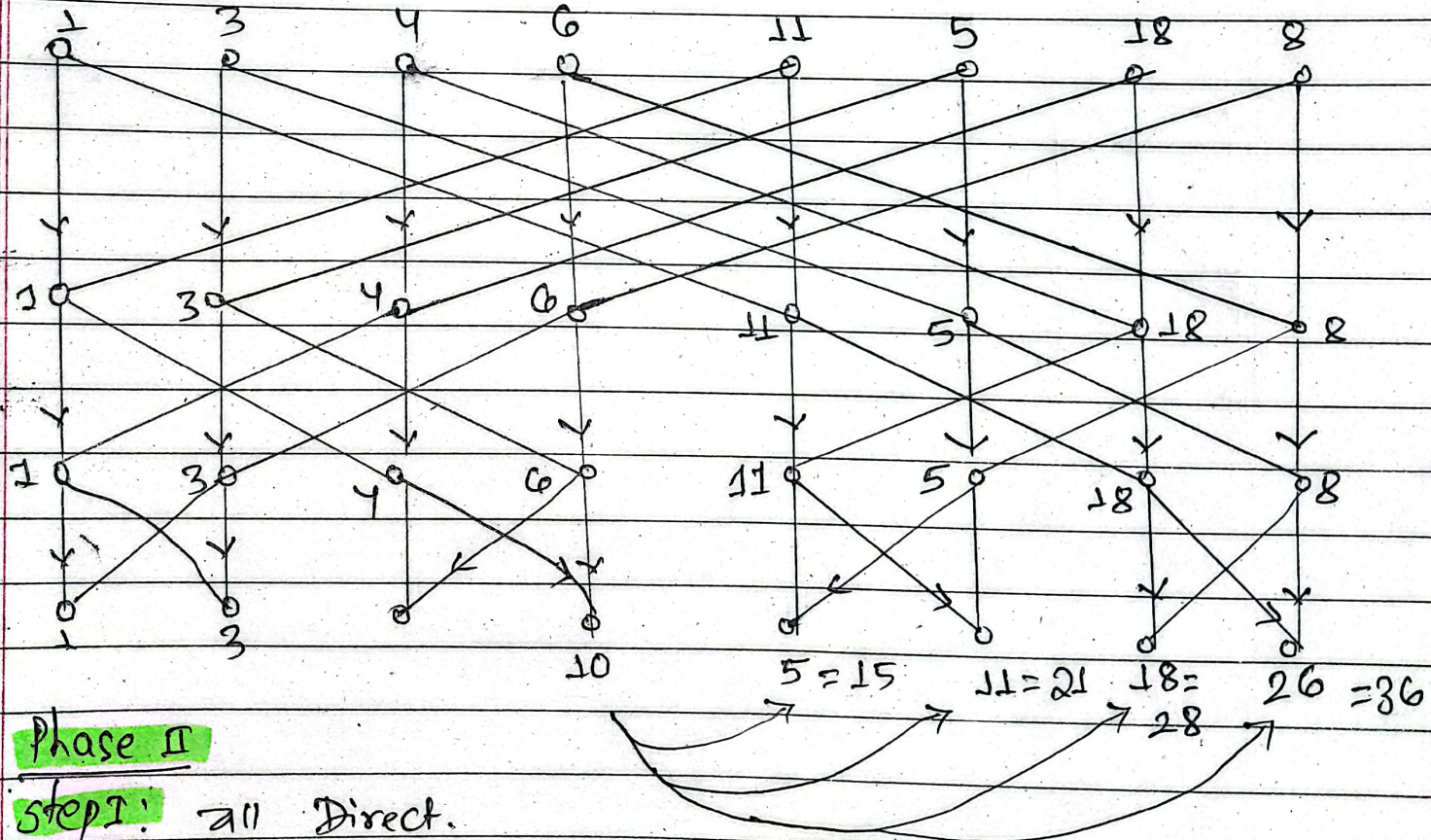
Step II: D, D, D, Sum, D, D, Sum, D.

Step III: all Direct.



Direct Cross Cross Sum Cross Direct Direct.

~~Cross~~ (Sum)



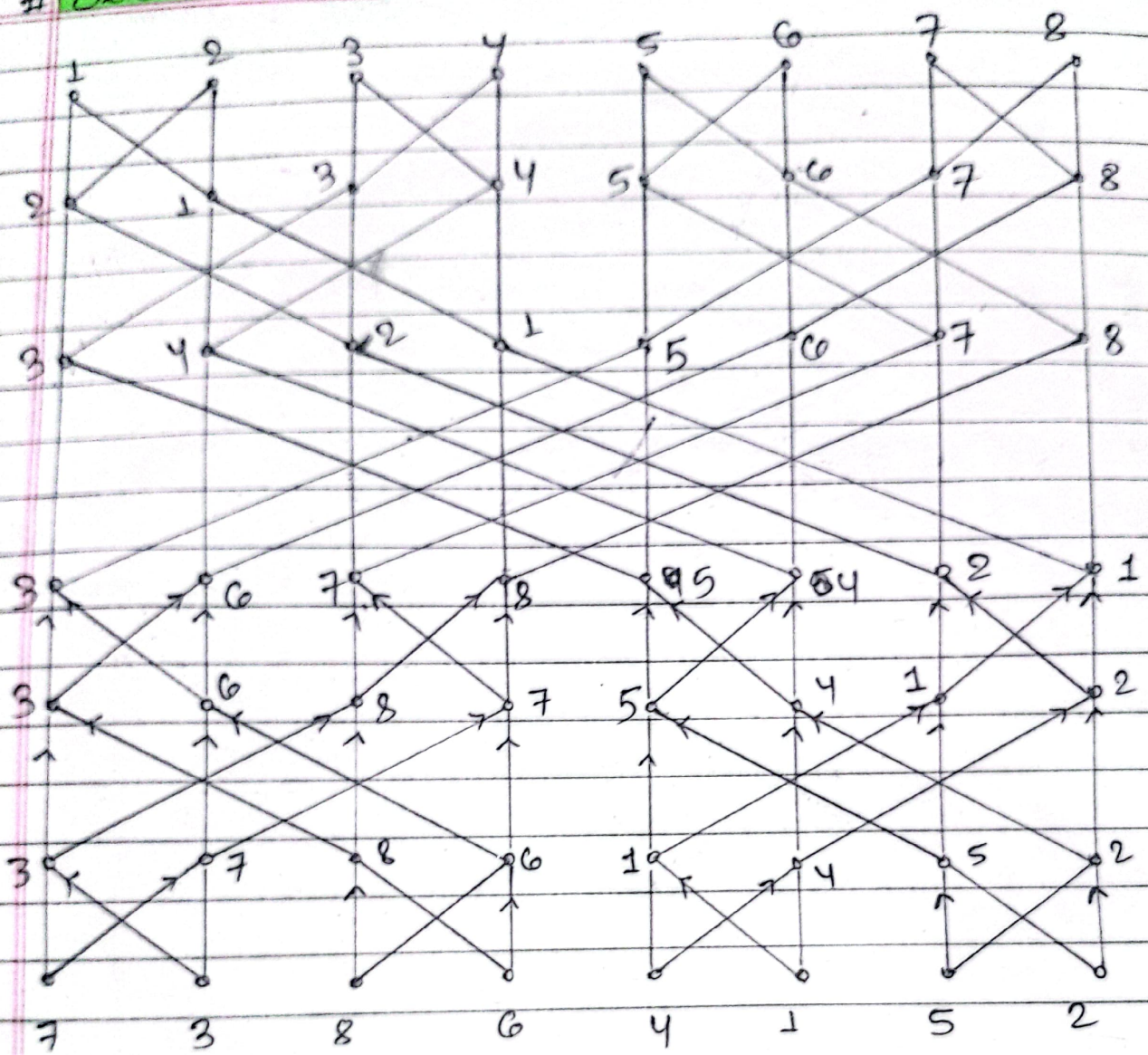
Phase II

Step I: all Direct.

Step II: all Direct.

Step III: D, D, Cross, Sum, Cross, Cross, Direct, Sum.

Selection Sort:



Step 1: Cross, Cross, Direct, Direct. Cross, Cross, Direct, Direct.

Step 2: Small Small Big Big Big Big Small Small.

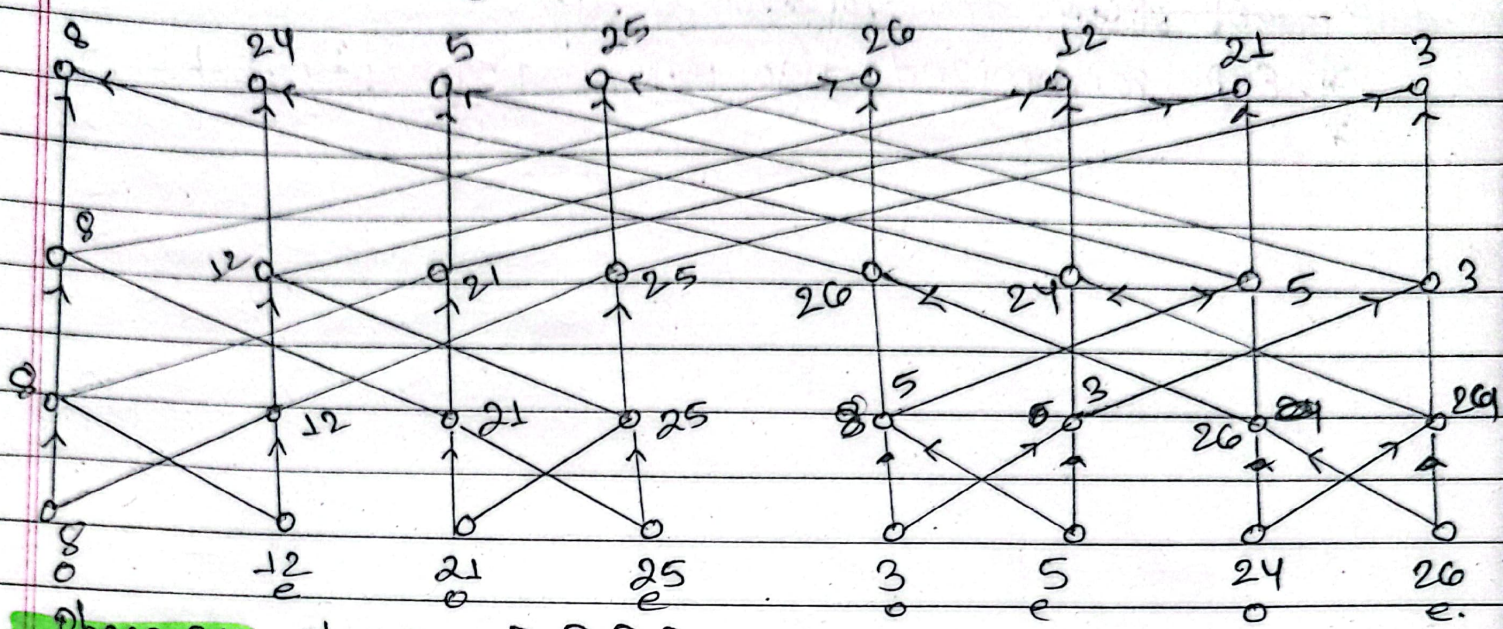
Step 3: Small Big Small Big Big Small Big Small

Step 4: Small Small Small Small Big Big Big Big.

Step 5: Small Small Big Big Small Small Big Big.

Step 6: Small Big Small Big Small Big Small Big.

Odd Even Merge.

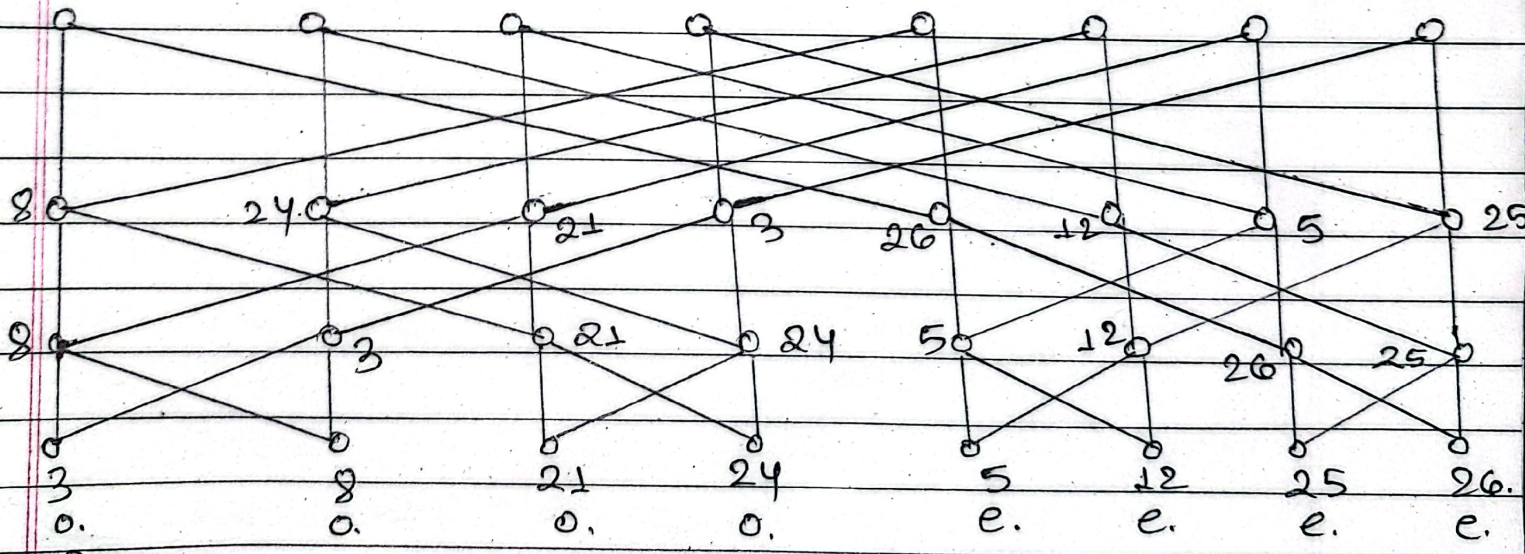


Phase I: Step 1: D.D.D.D. C.C.C.C.

Step 2: D.D.D.D. C.C.C.C.

Step 3: S.B.S.B. B.S.B.S.

[8 D 24] [5 C 25] [26 D 12] [21 C 3]



Phase II: Step 1: D.D. C.C. D.D. C.C.

Step 2: S.S. B.B. S.S. B.B.

Step 3: S. B S B. S. B. S. B.