2/1/20

Parallel Algorithms

"To Reduce Time Complexity"

Syllabus :
- intro
- dense mat
- decomp & map
- sorting
- searching & select.
- Graph Algos

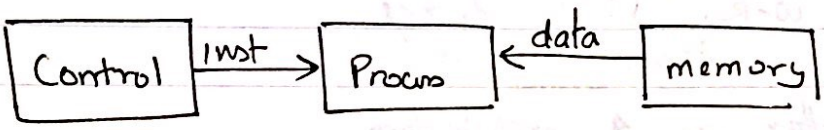
intro
- need.
- Models of comp.
- Analysis PA.
- expressing PAs.

Models of Computations

SISD	MISD	SIMD	MIMD
Single inst	Multiple inst	Single inst	Multi Inst
Single data	Single data	Multi dat	Multi Data

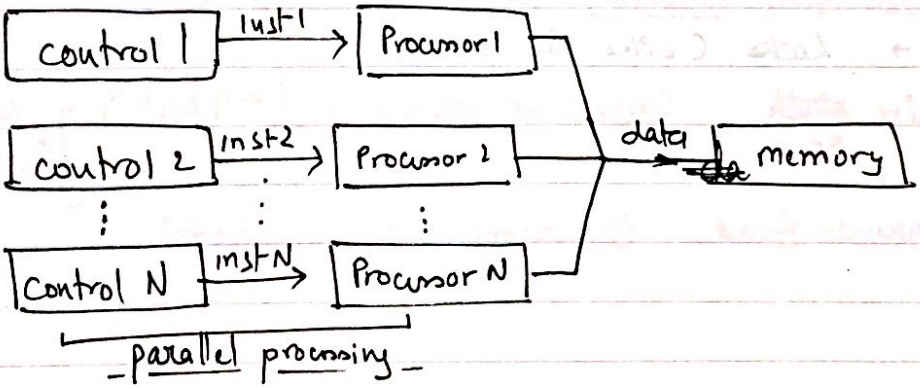
SISD

Single instruction Stream, Single data Stream.

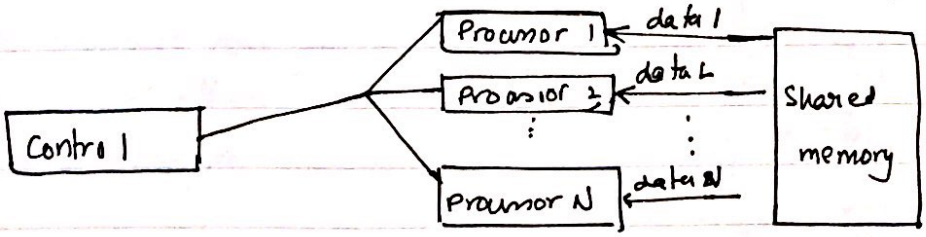


MISD

Multiple instruction stream, single data Stream



SIMD : Single instruction stream, multiple data stream



SIMD : is vulnerable to race condition, to avoid this, each processor has some local memory.
: The processor NO & ID is encoded when into the instruction

→ Shared Memory (SM-SIMD computer)
→ Parallel Random Access Machine (PRAM)

to do : processor$_i$ ⟷ processor$_j$

→ i processor$_i$ —write→ S. memory
ii processor$_j$ —Read→ S. memory
] =

⟹ errors → some other processor$_k$ writes data in between
i.e R-W, W-R, W-W errors
↑ ↑

Based on the errors there are 4 architectures.

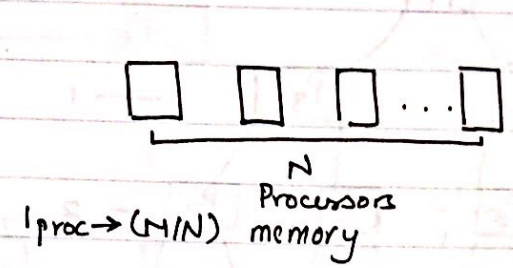
fool proof
→ Exclusive Read, Exclusive write (EREW)
→ basically critical section / locks.
→ Concurrent Read, Exclusive write (CREW)
→ Locks (like in DBMS)

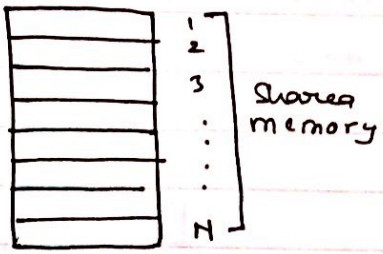
danger zone
→ Exclusive ~~Read~~ Read, Concurrent ~~Read~~ write (ERCW)
→ Concurrent Read, concurrent write (CRCW)

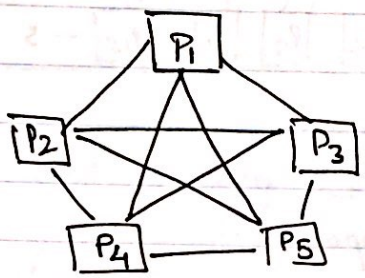
] → cause race condition issues.

9/1/18

Interconnection Network SIMD computer





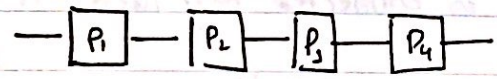


[connected to [n-1] other procs]

Fully Interconnected Network

\# There is 5 types of computers based on connections

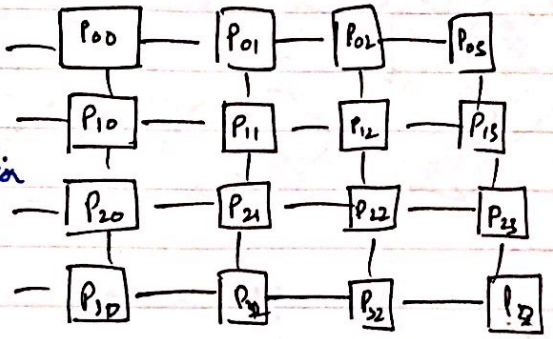
1> Linear Array (1D)



→ Connected linearly
→ Serial execution of tasks

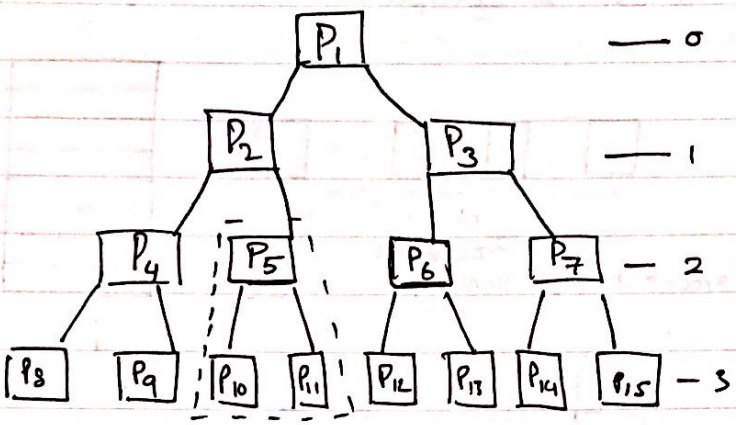
2> 2D Array (Matrix)

nxm mat N procs
→ All processors are equidistant from each other. [∵ of attenuation of signal strength, time delay]

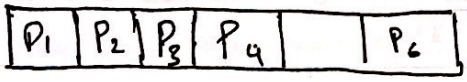


3> Tree Connection

d levels & 4



In memory



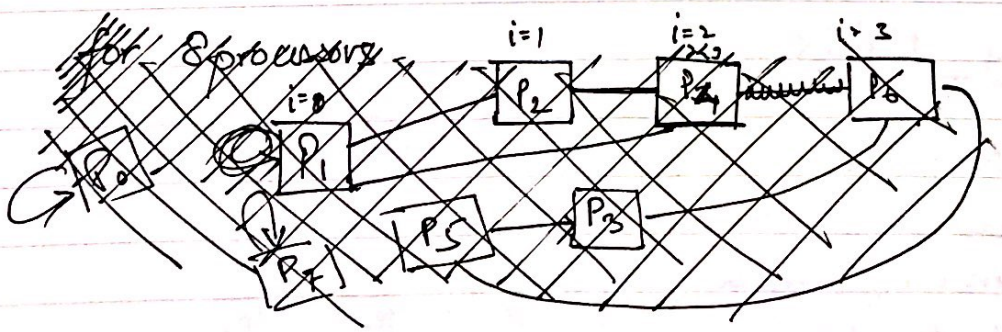
Stale when  is dropped.

→ only complete binary trees

4> Perfect Shuffle Connection

given N processors . if Pi is connected to Pj then it follows ..

$$j = \begin{cases} 2i & \text{for } 0 \le i < N/2 - 1 \\ 2i + 1 - N & \text{for } N/2 \le i \le N-1 \end{cases}$$



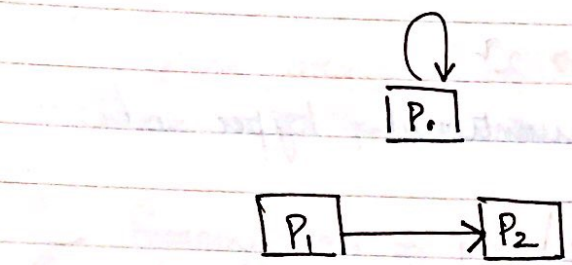
N/2 - 1 = 3

i = 4

8 + 1 - N

100 !

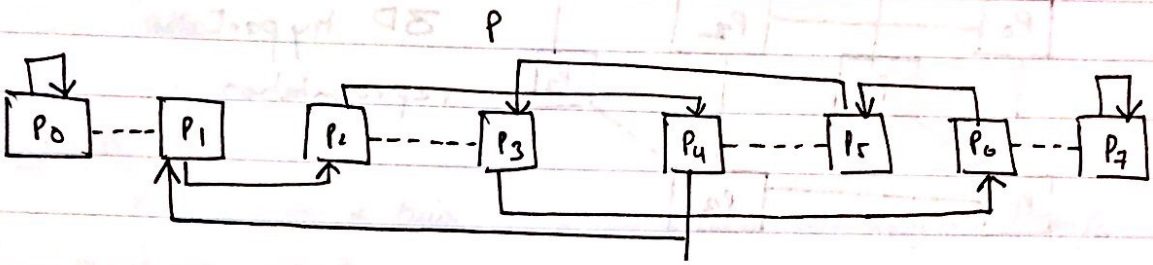
6 12+1



$i=0 \quad 0 \leq 3 \quad \therefore \quad 2i = 0$

$i=1 \quad 1 \leq 3 \quad \therefore \quad 2i = 2 \quad P_1 \rightarrow P_3$

$i=2 \quad 2 \leq 3 \quad \therefore \quad 2i = 4 \quad 2 \rightarrow 1$



These links are called ~~exchange~~ shuffle links, these facilitate exchange in data

Every alternate / even processor is connected to its successor, these are called exchang links.

Shuffle-Exchang Links.

↳ The final set of links.

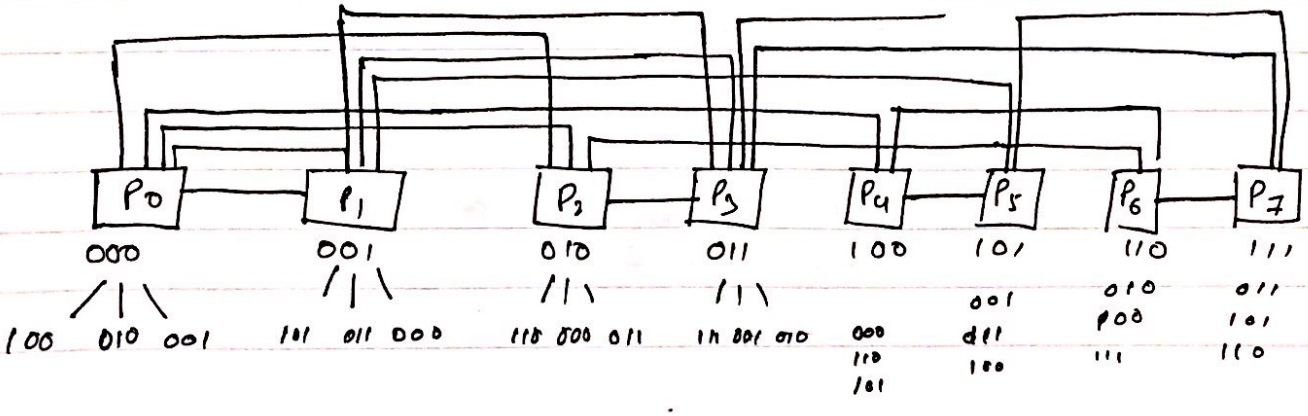
5> Cube Connection

N processors

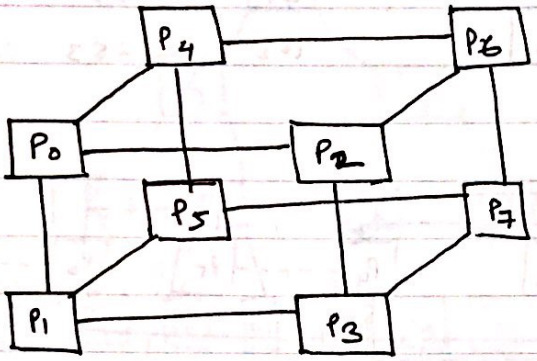
→ N procesos are represented as bits

eg N=04 => 00, 01, 10, 11

→ The interconnections are calculated as :

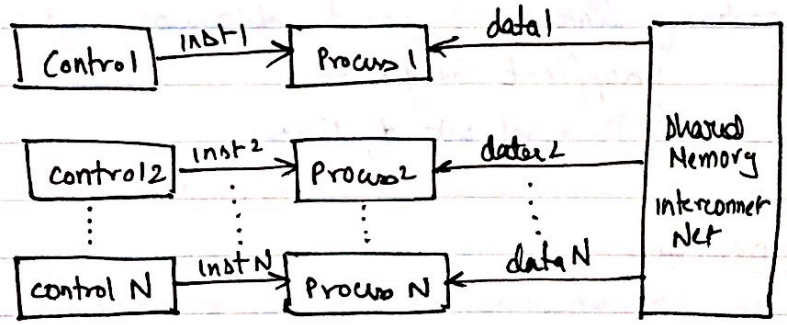


The processors are shown as $N \rightarrow 2^q$ and the layout is a q dimensional hyper cube.



3D Hypercube representation.

MIMD : Multiple Instruction Multiple data



Also known as distributed systems.

Analyzing Parallel Algorithms

→ Time → Space → Resources [processors]

i. Running Time of program
ii. No of processors used
iii. Cost.

> Running Time

→ Two types of time ie processing time and assigment/transfer

→ Counting Steps

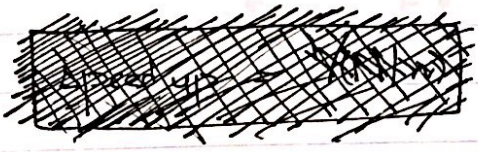
1) Computational Step
2) Routing Step

Sp

$$\text{speed up} = \frac{\text{time in serial execution}}{\text{time in parallel execution}}$$

Q A file of n entries is searched by an algorithm running on a CREW-SIMD computer with N processors calculate the speed up.

[∵ at one an N/n size chunk is checked by the N processors]



$$S.up = (N)/(N/N)$$

$$\Rightarrow \quad \text{Speed Up} = N$$

16/1/18

Matrix Operations

- Mat-Mat mult
  - Mesh
  - cube
  - CRCW
- Mat-vec Mult
  - Linear Array
  - Tree.

UNIT-2

1> Normal MM

$$A_{2\times3}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad B_{3\times4}\begin{bmatrix} B_{11} & B_{12} & B_{13} & B_{14} \\ B_{21} & B_{22} & B_{23} & B_{24} \\ B_{31} & B_{32} & B_{33} & B_{34} \end{bmatrix}$$

$$\begin{bmatrix} c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}\cdots \\ \vdots \end{bmatrix}$$

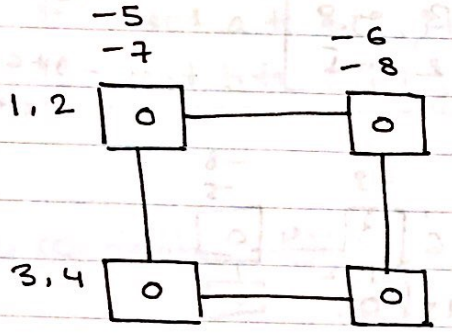
This is an $O(n^3)$

Matrix - Matrix Multiplication

1. Mesh Multiplication

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} -5 & -6 \\ -7 & -8 \end{bmatrix}$$

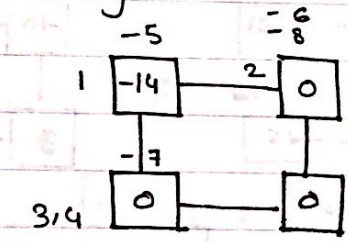
$$C = \begin{bmatrix} -19 & -22 \\ -43 & -50 \end{bmatrix}$$



1.

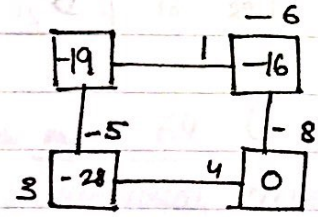
$C_{ij} = \{0\}$

2. Mult boundary elet



2x-7 , more 2 to next processor, 7 to next.

3. Mult boundary elet & propagate

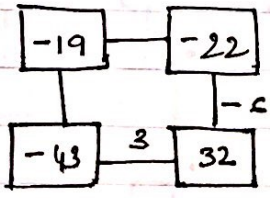


2x-8 , more 2, x 8

1x-5+(-14) more #, -5

4x-7 + (0) more 4 , discard 7

4. Mult boundary and propagate

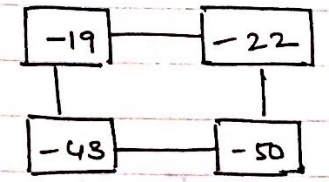


1*-6+(-16) more -6

-5x3+(-28) more 3 discard -5

5.

5.



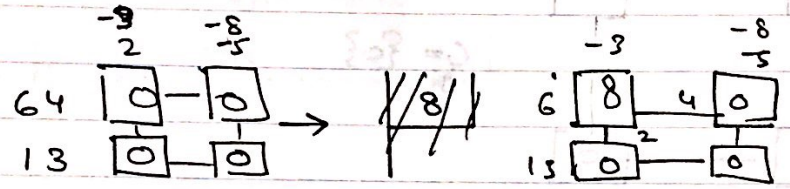
end

Example $A = \begin{bmatrix} 6 & 4 \\ 1 & 3 \end{bmatrix}$ $B = \begin{bmatrix} -3 & -8 \\ 2 & -5 \end{bmatrix}$

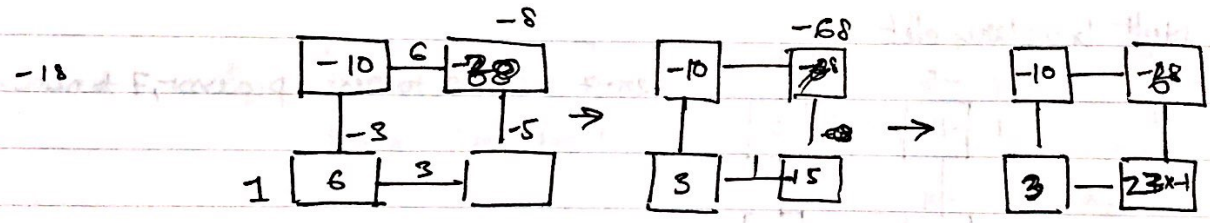
$-48 + -20 = 68$

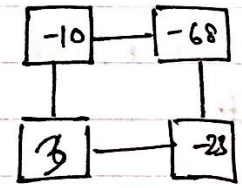
$-3 + 6 = 6$

$-8 + -15 = -23$



$-48 + 20$ $\quad -20 + -48$



$\Rightarrow$ 

$C = \begin{bmatrix} -10 & -68 \\ 3 & -23 \end{bmatrix}$

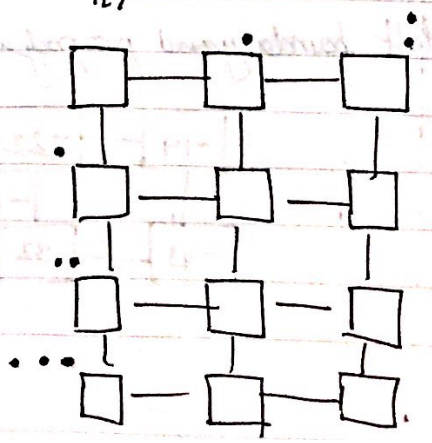
#2

$A_{m \times n} \times B_{n \times k} = C_{m \times k}$

$4 \times 5 \quad 5 \times 3 \quad 4 \times 3$

$b_{[i]}$

$a_{[i]}$



"•" Then one time lag in unit time i.e one • is one unit of delay.

Algorithm

```
MeshMult()
1. multiply the boundary element
2. res ± cij
3. send a to p(i, j+1) unless j = k
4. send b to p(i+1, j) unless i = m
```

time complexity : O(n)

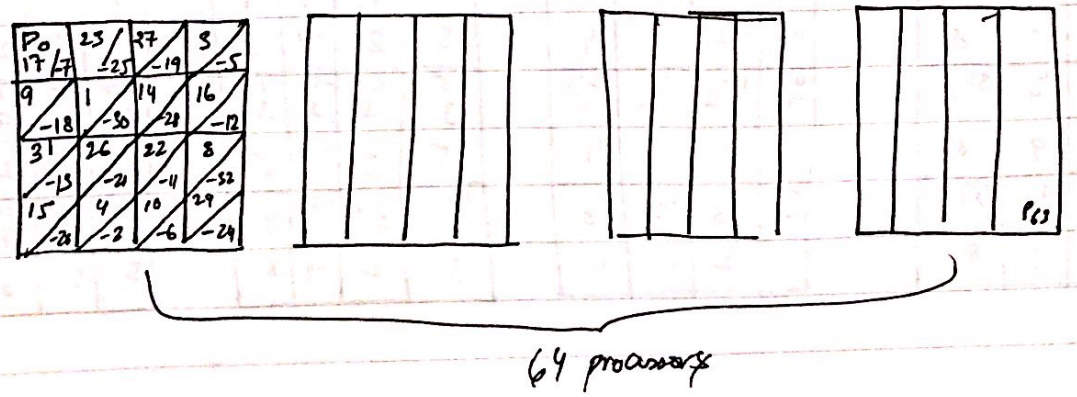
→ Increased no of processors, thus increased cost.

2. Cube Multiplication

$$A_{m \times n} \quad 4 \times 4 = \begin{bmatrix} 17 & 23 & 27 & 3 \\ 9 & 1 & 14 & 16 \\ 31 & 26 & 22 & 8 \\ 15 & 4 & 10 & 29 \end{bmatrix} \qquad B_{n \times k} \quad 4 \times 4 = \begin{bmatrix} -7 & -25 & -19 & -5 \\ -18 & -30 & -28 & -12 \\ -13 & -21 & -11 & -32 \\ -20 & -2 & -6 & -24 \end{bmatrix}$$

No of processors $n^3 = 64$ (2^6)

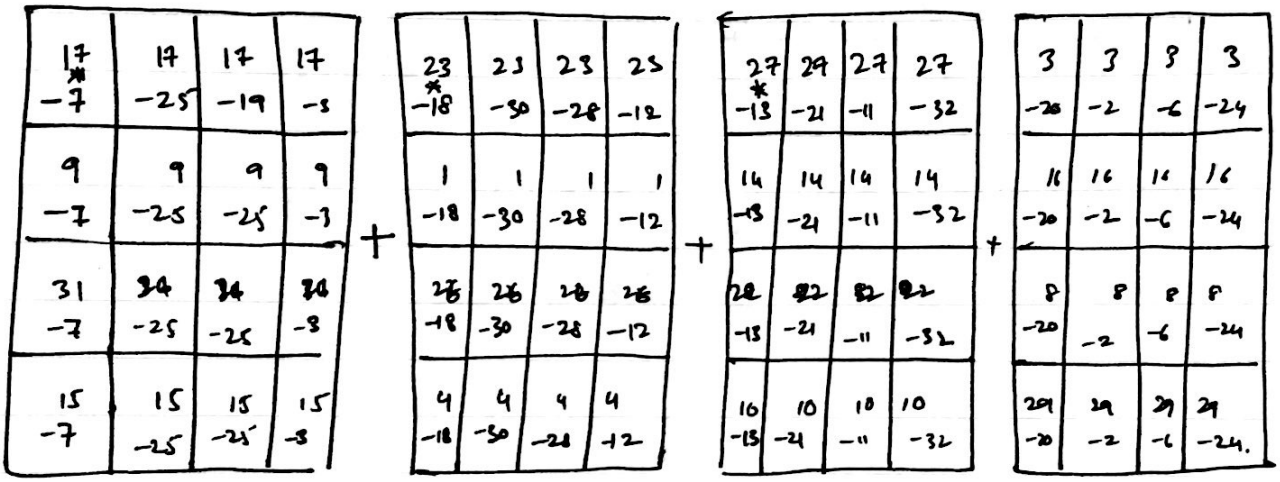
This is a 6 dimensional cube (when visualized)



17	23	27	3
-7	-25	-19	-3
9	1	14	16
-18	-30	-28	-12
31	26	22	8
-15	-21	-11	-32
15	4	10	29
-20	-2	-6	-24

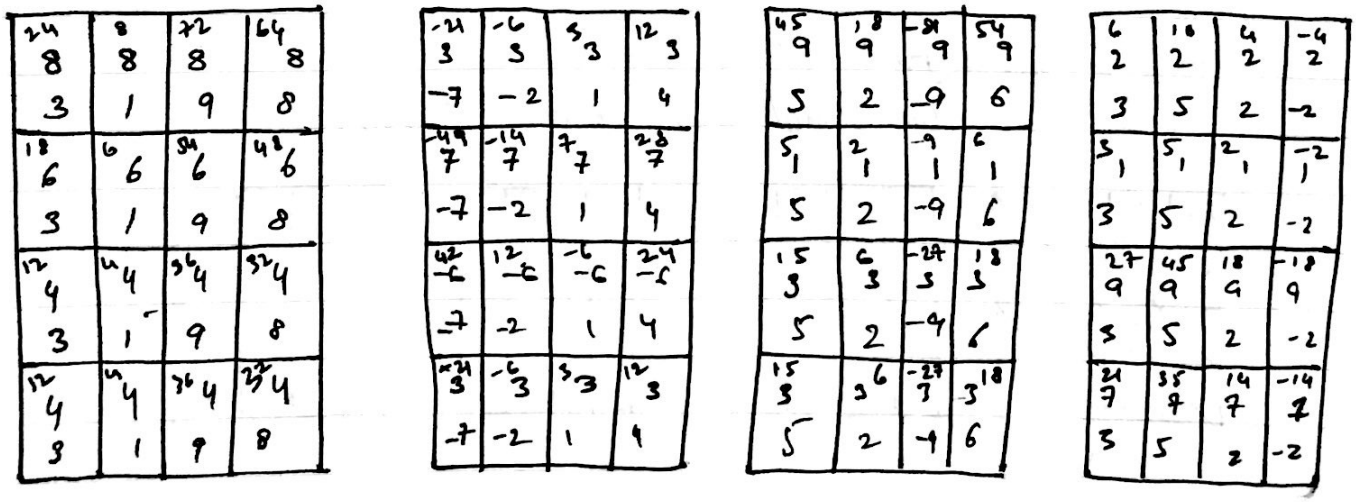
[the 16 pairing

The processor distribution of elements:



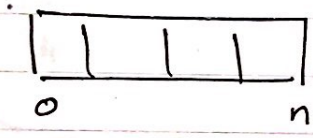
1. multiply all internal elements.
2. add up all the matrices.

Example:

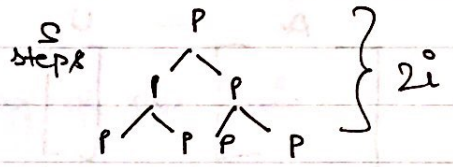


$$\begin{bmatrix} 54 & 30 & -2 & 126 \\ -23 & -1 & 54 & 80 \\ 96 & 67 & 21 & 8 \\ 27 & 39 & 26 & 48 \end{bmatrix}$$

Serial addition vs parallel addition



$n-1$ operations

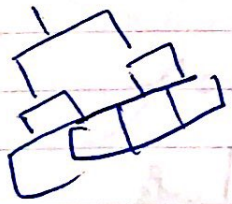


$$\rightarrow \sum_{i=0}^{s-1} 2^i = \frac{2^s - 1}{1} = 2^s - 1 \text{ steps}$$

$$n - 1 \leq 2^s - 1$$

$$n \leq 2^s$$

$$\Rightarrow s \geq \log(n) \Rightarrow \boxed{\Omega(\log(n))}$$



22/1/18

CRCW multiplication

time complexity $O(1)$

cost = n^3 [$p(n) \times t(1) = O(n^3) * O(1)$]

Matrix-Vector Multiplication

$$\underset{m \times n}{[A]} \times \underset{n \times 1}{[U]} = \underset{m \times 1}{[V]}$$

→ result is a col^n vector

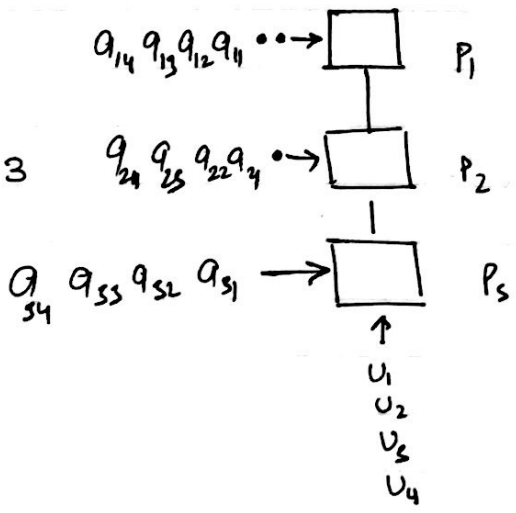
$$v_i = \sum_{j=1}^{n} a_{ij} \cdot v_j \qquad (1 \le i \le m)$$

1. Linear Array :

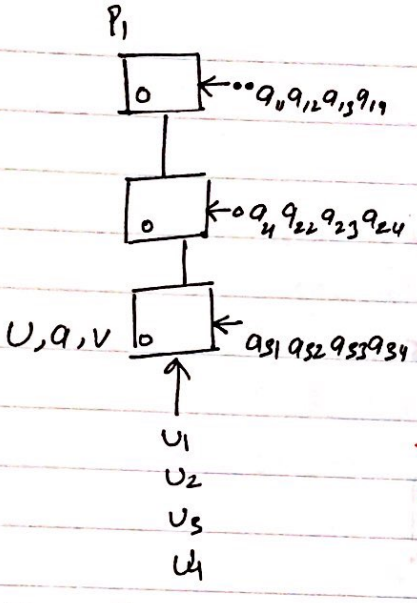
$P_1, P_2, \ldots, P_m$ // processors

$v_i = 0$

$n = 4 \mid m = 3$



init all or 0



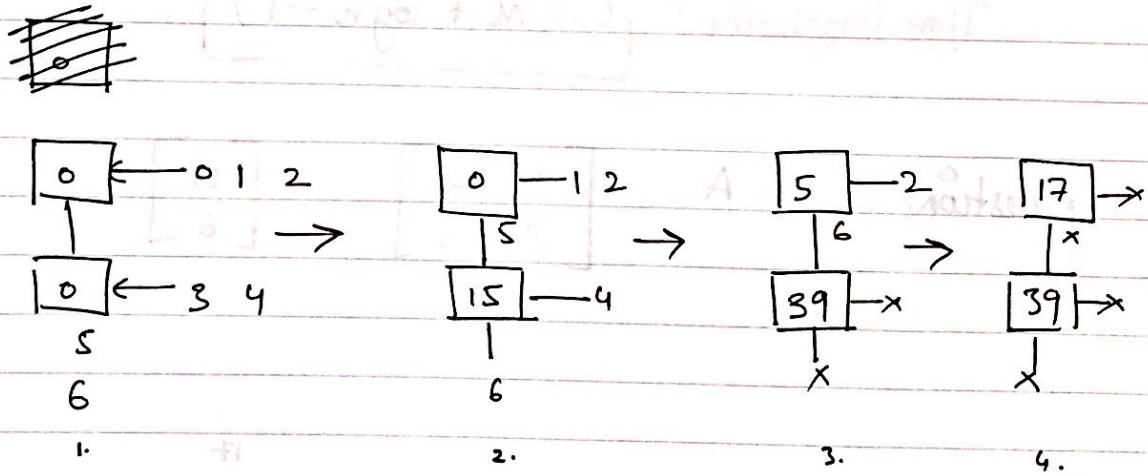
U → in commiy U
a → in commiy a
V → resultant

Time complexity: $O(m+n-1)$

-1 due to lag of one unit

Question

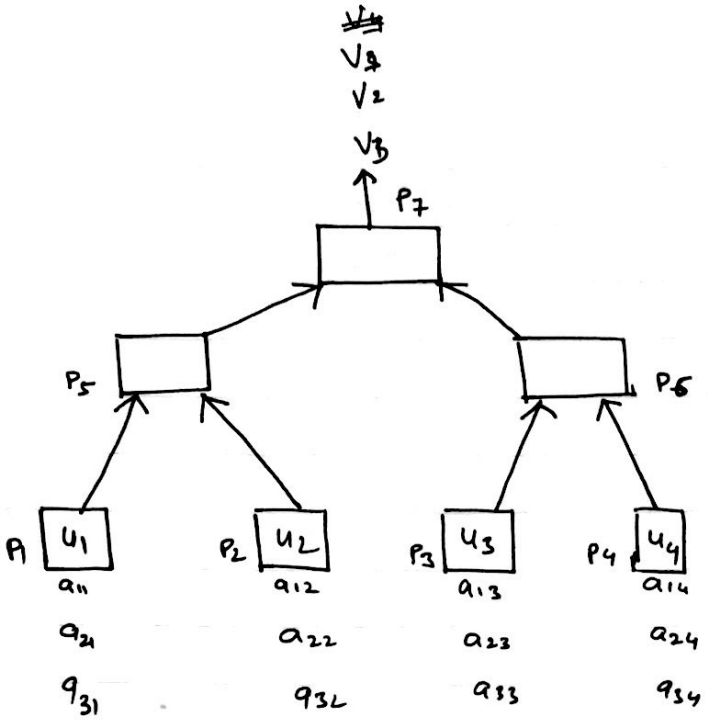
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad U = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \quad V = \begin{bmatrix} 17 \\ 39 \end{bmatrix}$$



Tree Multiplication

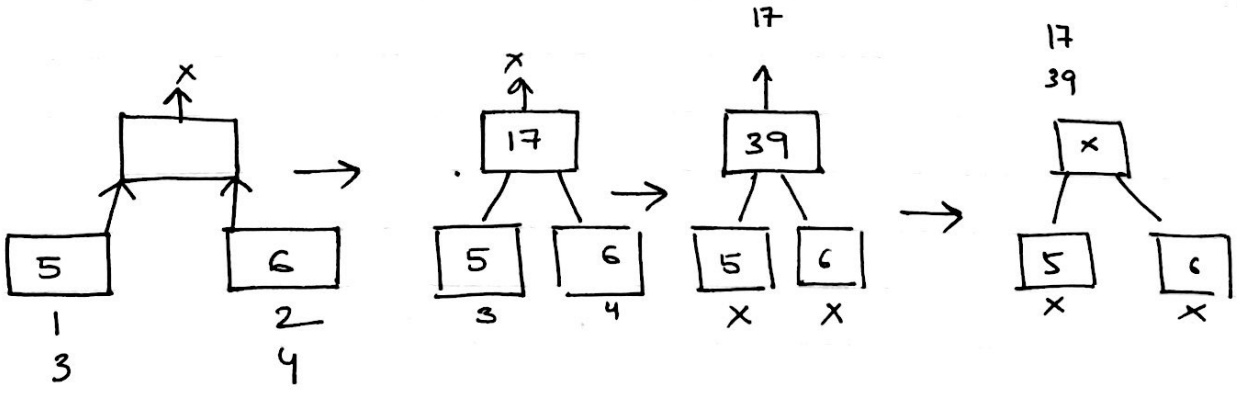
for $m = 3, n = 4$

$$A \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} U_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = V$$



Time Complexity : $O(m + \log n - 1)$

Question A $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\begin{bmatrix} 5 \\ 6 \end{bmatrix}$



UNIT 3 Sorting-

1. Hyper Quick Sort → Quick Sort.
   → Parallel Quick Sort.
   → Hyper Sort.
2. Merge Sort
3. Bitonic Merger Sort
4. Odd-Even Transposition
5. Enumeration Sort
   - → CRCW model.
   - → CREW model.
   - → EREW model.

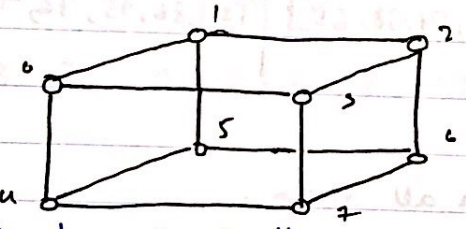
Time complexity : $O(\log n)$

: Finding the pivot [the selection of ~~random~~ median element in parallel is a computationally expensive in parallel].

Hyper Quick Sort

"quick sort on a hypercube."

1. divide unsorted ~~elements into~~ list among each nodes
2. select some median @ node ~~from~~ & Broadcast to all nodes



2. from node 0 broadcast the median value
3. split each list locally & exchange halves with higher dimension
4. Repeat 2,3 in parallel until dimension 0
5. Sort the unsorted list sequentially using quick sort

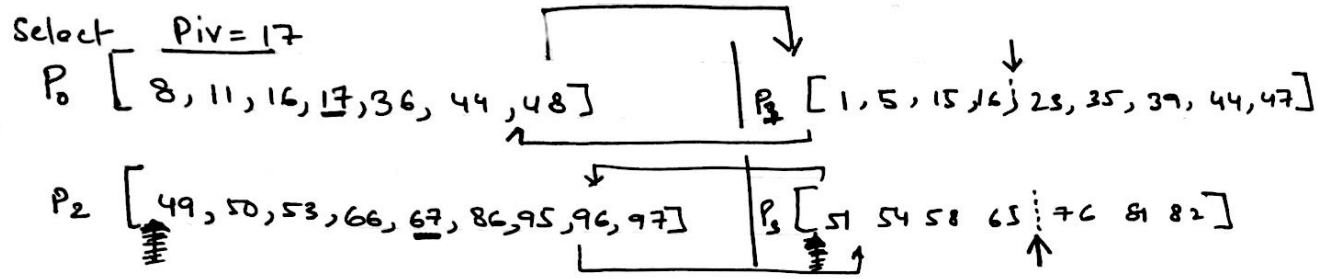
P_0	97, [48], 16, 8, 66, 96, 17, 49	P_1	58 76 54 39 82 47 65 57
P_2	11, 50, 53, 95, 36, 67, 86, 44	P_3	35, 16, 81, 1, 44, 23, 15, 5
P_0'	8, 16, 17, 48, 49, 66, 96, 97	P_1'	39, 47, 51, 54, 58, 65, 76, 82
P_2'	11, 36, 44, 50, 53, 67, 86, 95	P_3'	1, 5, 15, 16, 23, 35, 44, 81

Considering P_0 & P_2

Swap (lower (P_0), upper (P_2))

considering P_1 & P_3

Swap (lower(P_1), upper(P_3))



Select Piv =

P_0 [8, 11, 16, 17, 1, 5, 15, 16]	P_1 [36, 44, 48, 23, 35, 39, 44, 47]
P_2 [49, 50, 53, 66, 67, 51, 54, 58, 65]	P_3 [86, 95, 96, 97, 76, 81, 82]

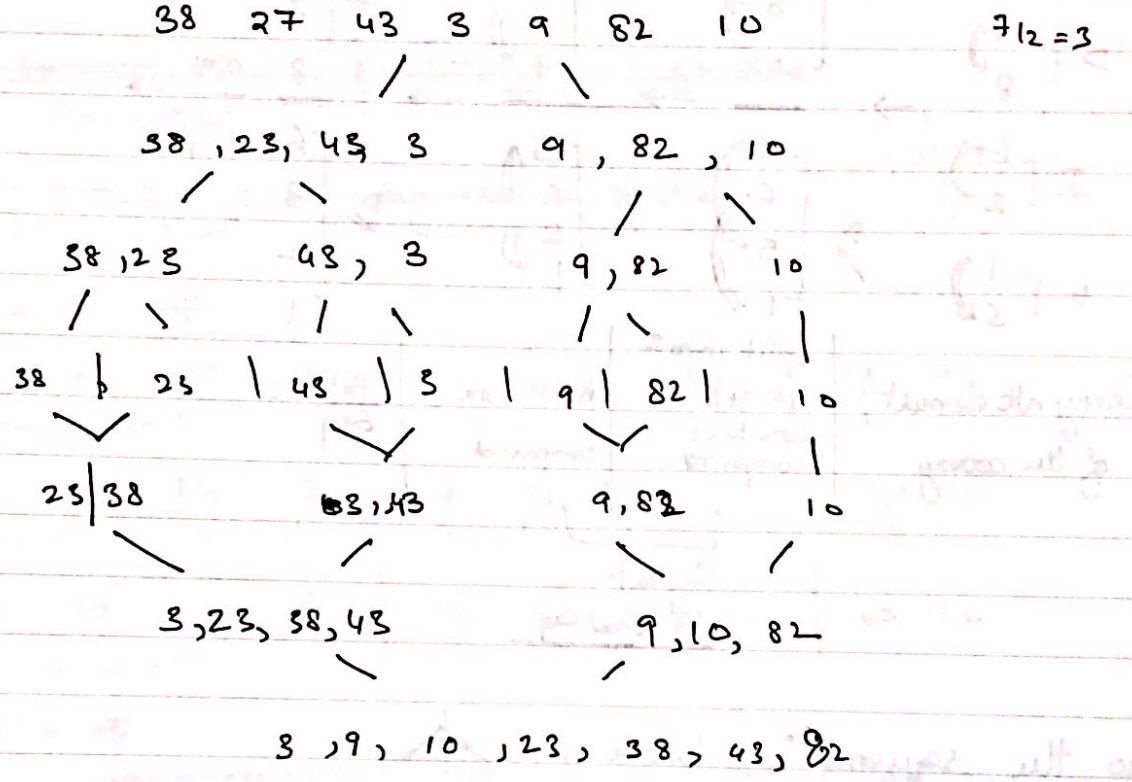
1) Seq Quick sort on all nodes.

Time Complexity : $O\left(\frac{N}{P}\log\left(\frac{N}{P}\right)\right)$

5/2/18

Merge Sort

Eg

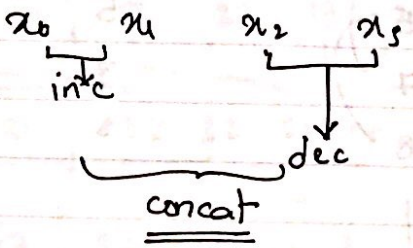


Bitonic Sort

The seq must be bitonic i.e increasing till n/2 and then decreases

The seq is ~~bit~~ ~~bitonic~~ bitonic when
$a[0, \ldots, n-1]$ $0 \leq i \leq n-1$ And $x_0 \leq x_1 \cdots \leq x_i$, $x_i \geqslant x_{i+1} \geqslant x_{n-1}$

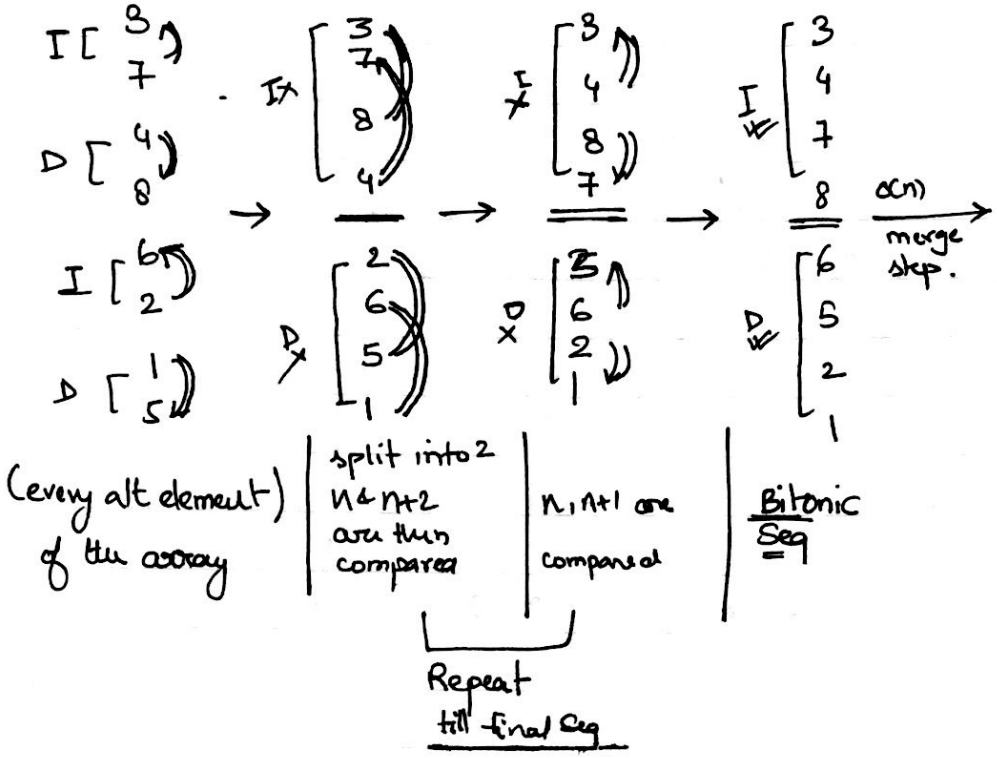
Then



① ② ⑤

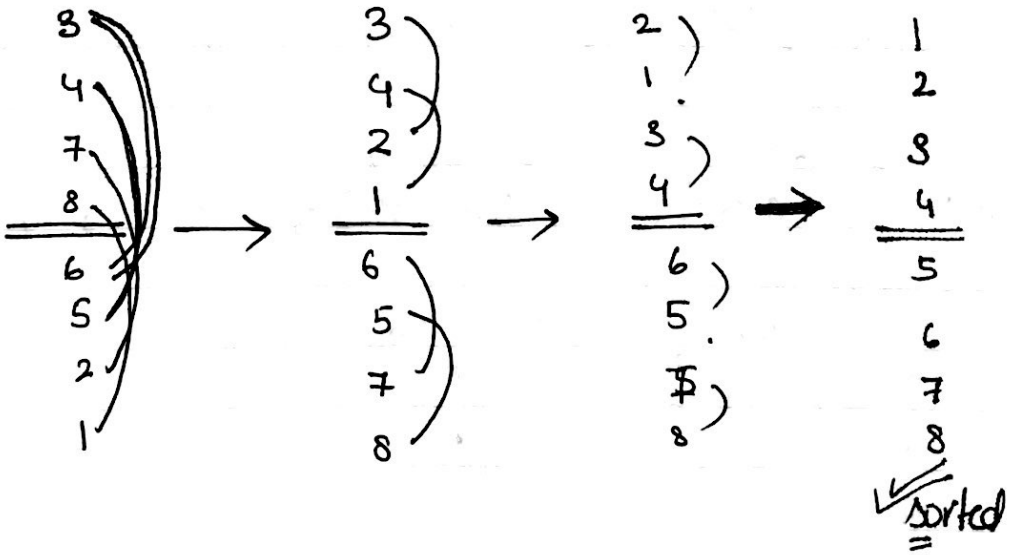
)→ comparision

eg



Now the sequence is bitonic i.e 

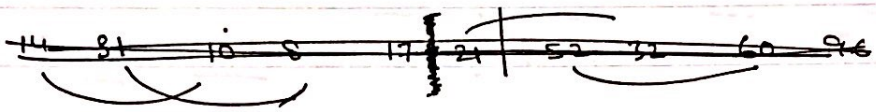
Sorting

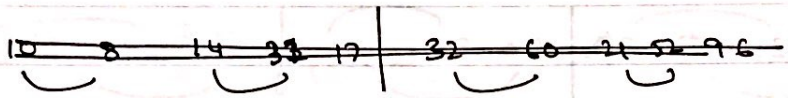


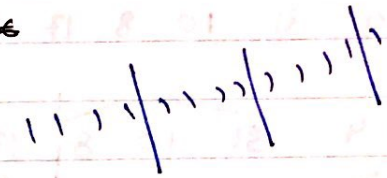
i.e ceil($\log_2^{(n)}$).

Qus

14 31 10 8 17 21 52 32 | 60 96



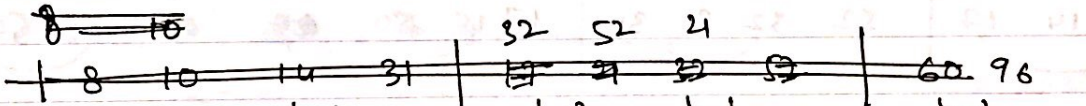


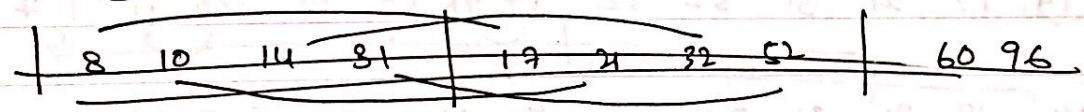


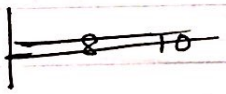
14 31 10 8 17 21 52 32 | 60 96

10 8 14 31 17 21 52 32 | 60 96

8 10 14 31 | 17 21 32 52 | 60 96







14 31 10 8 | 17 21 52 32 | 60 96

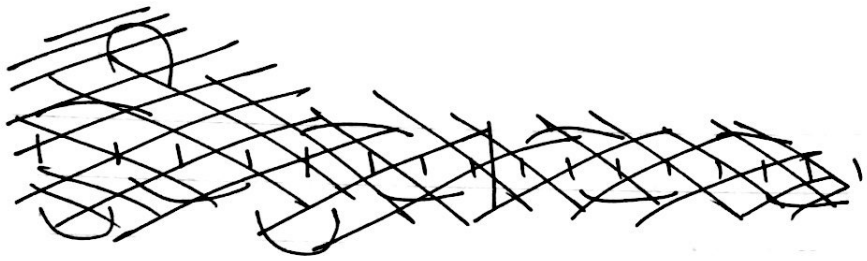
10 8 14 31 | 52 32 17 21 | 60 96

8 10 14 31 | 52 32 21 17 | 60 96

8 10 14 17 | 52 32 21 31 | 60 96

8 10 14 17 | 21 31 52 32 | 60 96

8 10 14 17 | 21 31 31 52 | 66 96



I D I D I D XI D

9 14 31 10 8 17 21 52 32 60 96 11 1 67 45 50 7

14 31 10 8 | 17 21 52 32 | 60 96 11 1 | 45 67 50 7

10 8 14 31 | 17 21 52 32 | 60 96 11 1 | 50 67 45 7

8 10 14 31 | 17 21 35 52 | 96 60 11 1 | 67 50 45 7

8 10 14 31 | 52 32 21 17 | 1 11 60 96 | 67 45 50 7

8 10 14 17 | 52 32 21 31 | 67 45 50 96 | 1 11 60 7

8 10 14 17 21 31 52 32 | 67 96 60 45 50 11 1 7

8 10 14 17 21 31 32 52 | 96 67 60 45 50 11 7 1

8 10 14 17 | 21 11 7 1 | 96 67 60 45 | 50 31 32 52

8 10 14 17 | 7 1 21 11 | 60 45 96 67 | 50 52 32 31

8 10 14 17 | 1 7 11 21 | 45 60 67 96 | 50 52 32 31

8 10 11 17 | 1 7 14 21 | 50 60 67 96 45 52 32 31

1 7 11 8 7 14 21 | 67 96 50 60 45 52 32 31

Time Complexity : $\Theta(\log^2(n))$

21/2/18

Odd-Even Transposition

n processors $P_1, P_2 \ldots P_n$

Sequence $S = \{s_1, s_2, \ldots, s_n\}$

1. All odd no. processors P_i obtain x_{i+1} from P_{i+1}
   if $x_i > x_{i+1}$:
   swap ()
2. Repeat (1.) using even no processors
3. when $x_i < x_{i+1} \quad \forall \ 1 \leq i \leq n-1$
   stop()

$S = \{6, 5, 9, 2, 4, 3, 5, 1, 7, 5, 8\}$

1. odd pos
   $\{5, 6, 2, 9, 3, 4, 1, 5, 5, 7, 8\}$
2. Even
   $\{5, 2, 6, 3, 9, 1, 4, 5, 5, 7, 8\}$
3. odd
   $\{2, 5, 3, 6, 1, 9, 4, 5, 5, 7, 8\}$
4. Even
   $\{2, 3, 5, 1, 6, 4, 9, 5, 5, 7, 8\}$

Decomp & Mapping Techniques
→ DB query processing
→ 15 puzzle Problem
→ Parallel Discrete Event Sim
→ Image Dithering
→ Dense LU factorization
• PPT after mid sems.

MID SEMS
I. INTRO
- Need for ll compr
- Models of comp
- Analysing Pa.
- Expressing Pa
II. Dense Matrix Algo
a) M·V mult
b) M·M mult
III. Sorting
- Hyper quick
- Merge
- Bitonic Merge
- Odd even transp.

{ 2, 3, 1, 5, 4, 6, 5, 9, 5, 7, 8 }
{ 2, 1, 3, 4, 5, 5, 6, 5, 9, 7, 8 }
{ 1, 2, 3, 4, 5, 5 5, 6, 7, 9, 8 }
{ 1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9 } → stop.

Time Complexity : $O(n)$
Cost of algo : $O(n^2)$

5.14.18

Enumeration Sorting

Enumerate how many elements come before any element. The position each element S_i of S in the sorted sequence is determined by computing C_i

i.e the no of elements $< S_i$

1. CRCW

uses n^2 processors

Eg S 2 4 3

C 0 2 1

⇒ 2 3 4

1. if $S_i = S_j$,
   if $i > j$
   $S_i > S_j$
2. After all C_i are computed, S_i is placed @ $C_i + 1$

Eg> $S = \{5\ 2\ 4\ 5\}$ $P = n^2 = 16$

S

5
2
4
5

	1	2	3	4
1	5,5	5,2	5,4	5,4
2	2,5	2,2	2,4	2,5
3	4,5	4,2	4,4	4,5
4	5,5	5,2	5,4	5,5

C

2
0
1
3

due to 1. Step 1

S

2
4
5
5

Sorted

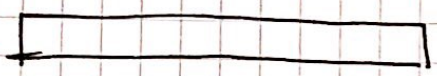
Complexity Time $O(1)$

Cost $O(n^2)$

Searching

18/4/18

① EREW

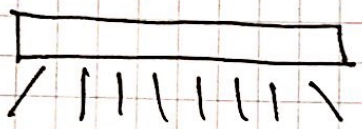


$|S| = n$

N processors

① Broadcast $O(\log(N))$

$1 \leq N \leq n$



N Sequences of S in each seq do Binary Search.

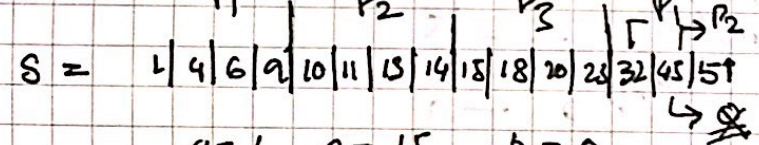
$$\text{Time} = \boxed{O\left(\log \frac{n}{N}\right)}$$

worst case $= O(\log(n))$

② CREW

q, r, k, g, j

k ← stores the position of found X



(let N = 3)

q = 1 r = 15 k = 0

$$g = \log(n+1) / \log(N+1) = 2$$

$$j = (q-1) + i(N+1)^{g-1} = 4$$

$S_4 - x$

$q < 45$

$S_{14} = x$

$$O\left(\frac{\log(n+1)}{\log(N+1)}\right)$$

$$= O\left(\log_{N+1}(n+1)\right)$$

Random

ER EW
ERCW
CREW
CRCW
Tree
Mesh

▷ ErEw

1 Broad Cast. — log (N)
2 Sequential Search — O (n/N)
3 Store O(log (N))

2) ERCW

1] Broad cast + Search
2
5 Store — O(1)

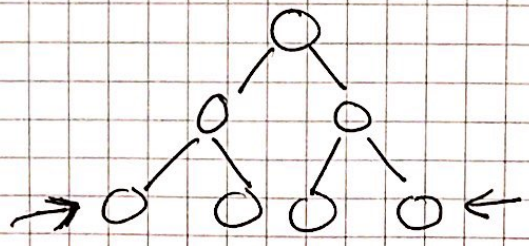
3) CREW

1 — O(1)
2 — O(n/N)
3) — log(N)

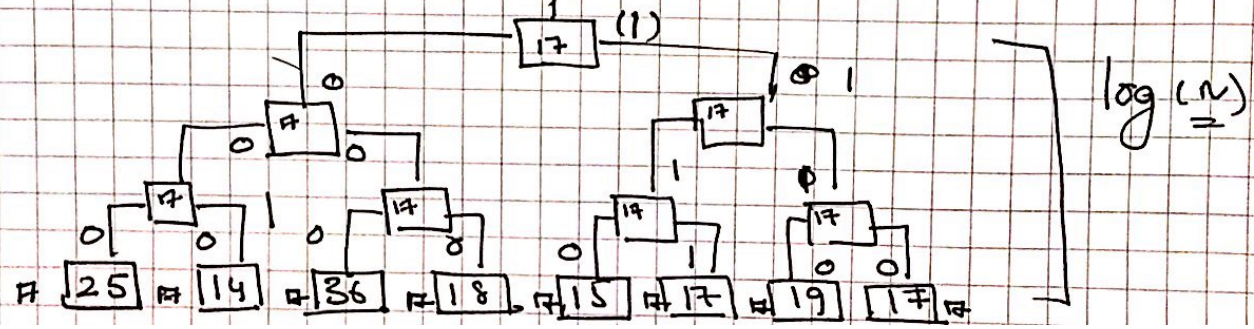
4) CRCW

1) — O(1)
2 — O(n/N)
3) — O(1)

Tree



S = { 25, 14, 36, 18, 15, 17, 19, 17 } x = 17



23/4/18

Searching –

Mesh

ANS ← 0/1

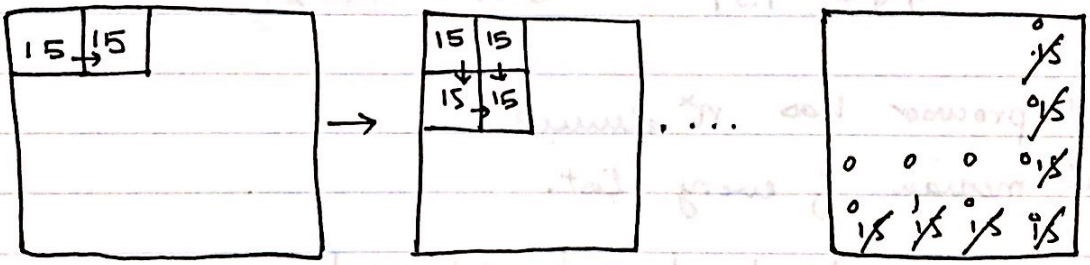
P(1,1)

1	2	3	4
8	7	6	5
9	10	11	12
16	15	14	13

P(4,4)

$x = 15$

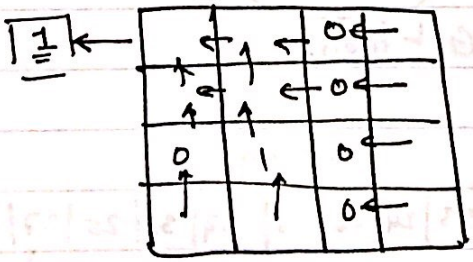
1. Search. (Broadcasting values)
2. Folding. (matching)



1. Search

(Read all steps in book).

2. Folding (x || y)



Complexity : ?

Selection

Sequence S, of n elements

$1 \le k \le n$ $\quad |S| = (n)$

k is the k^{th} smallest elements.

P_1 P_2 P_3 P_4

1.) 3 | 14 | 16 | 20 | 8 | 31 | 22 | 12 | 33 | 1 | 4 | 9 | 10 | 5 | 13 | 7 | 24 | 2 | 24 | 26 | 18 | 34 | 36 | 25 | 14 | 27 | 32 | 55 | 33

$n = 29$ | $k = 21$ | $N = 5$. $\quad P_5$

2.) $N = |S|^{1-x} \quad (x \in [0,1])$

3.) Each processor has n^x elements

4.) Find median of every list.

$N =$ | 14 | 9 | 7 | 25 | 32 |

take median of median

$= \boxed{14}$

$k \geq 14$

5. divide list into Less than / Eq / Grt than.

L | E | Greater.

3 | 8 | 12 | 1 | 4 | 9 | 10 | 5 | 13 | 7 | 2 | 14 | 14 | 14 | 16 | 20 | 31 | 22 | 53 | 24 | 26 | 18 | 34 | 36 | 25 | 27 | 32 | 35 | 33 |

↑

6. Since we have $k > 14$ we now work on the GT Eq

$N = |G|^{1-n}$ ⇒ $N = 3$

P1 | P2 | P3

| 16 | 20 | 31 | 22 | 33 | 24 | 26 | 18 | 34 | 36 | 25 | 27 | 32 | 35 | 37 |

(14)

| 22 | 26 | 32 |

M' = 26

L | E | G

| 16 | 20 | 22 | 24 | 18 | 25 | 26 | 31 | 33 | 34 | 36 | 27 | 32 | 35 | 33 |

Ind = 24

26 Ans