Unit 11.

Parallel and Distributed Computing.

Parallel Computing:

- also called Parallel Processing.
- there are multiple processors.
- each has its own assigned processor or task
- multiple computations are performed simultaneou
- have access to shared memory to exchange data and instructions between processor.

~~In par~~

In parallel computing, ~~the~~ all the processing units are supposed to share the common memory.

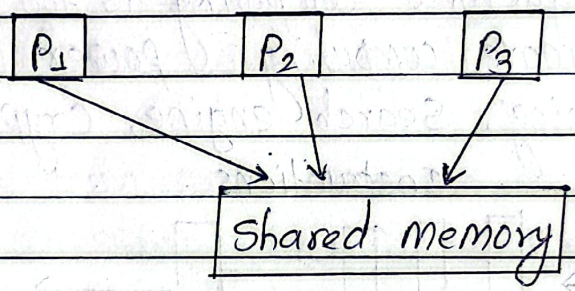


fig: Parallel Computing.

Distributed Computing:

(Processors)

- Divides single task between multiple Computers.
- All computers works together to achieve common goal. Hence they work as single entity.
- Each processor has its own private memory.
- Information is exchanged by passing message between processor. (RPC, RMI)

RPC: (Remote Procedural call) (follows Procedural progg)

RMI: (Remote method Invocation) (follows OOP)

In distributed computing, all the processing units are graphically distributed and consist of their own local private memory.

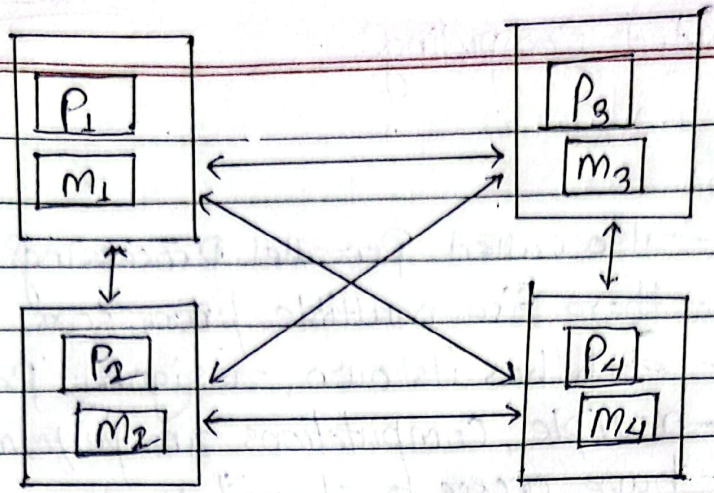


fig: Distributed Computing

Q. Need of Parallel Computing Paradigm:

Why do we need Parallel Computing?

- Serial / sequential computing is too slow.
- Need of more computing power.

(Data Mining, Search engine, Cryptography)

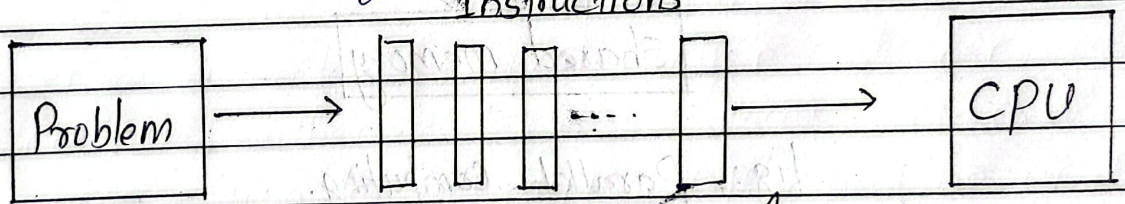


fig: Traditional Approach.

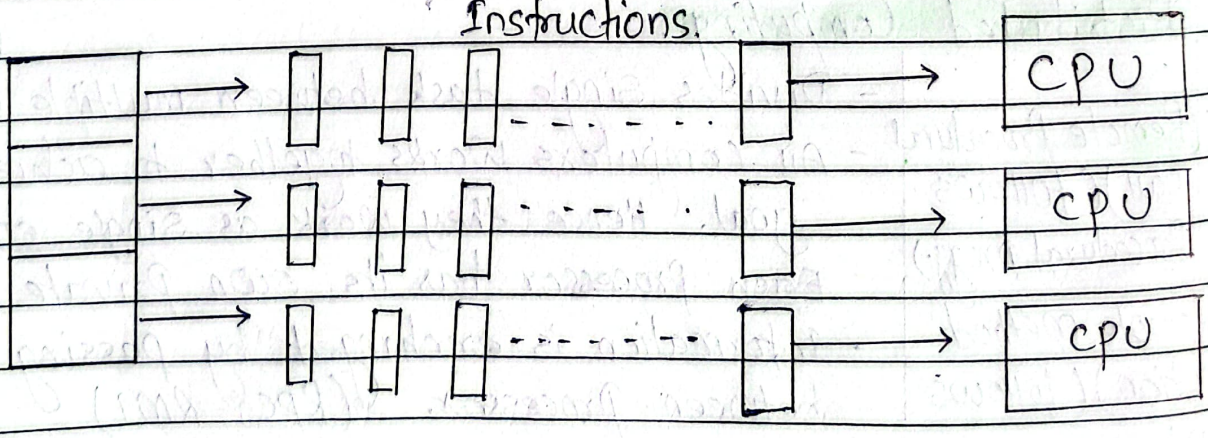


fig: Parallel Computing Approach.

Q. "Problem arise in distributed system when Processor need to share resources."

Date
Page

Issues on Shared Resources: (Race condition)

- Issue faced while sharing the resources by multiple processor is ~~the result~~ inconsistent result also known as race condition.

Race condition:

A condition of programming where its behavior depends on relative timing or interleaving of multiple threads or process.

It is a situation where,
- The final outputs produced depends on the execution order of instruction of different processes.
- Several processes compete with each other.

When more than one process is either executing the same code or accessing the same shared resource, then there is a possibility that the output is wrong. As several processes access and manipulate same data concurrently, the outcomes depends on the order in which the access takes place.

Thread-Safe:

If a program or data structure is free of race condition, when accessed by multiple-thread then it is called Thread-safe.

~~To avoid~~
* Mutual Exclusion → Interrupt Disablinge & locked variable.
* Strict Alteration.
* Sleep and Wakeup.

How to avoid Race Condition?

To prevent race condition from occuring, we can lock shared variables so that only one thread can have access to the shared variable at a time. which is called Mutual Exclusion.

Mutual Exclusion:

a property of process synchronization which states that "no two processes can exist in the critical section of any given point of time."

~~Impletin~~

Implementing Mutual exclusion with busy wait.

* Disabling interrupts: (Busy wait)

allowing a process to disable interrupt before it enters critical section and then enable interrupts when it leaves critical section. When a process disable interrupts, CPU will be unable to switch process and any other process won't have access to that resource which guarantees that the process can use the resource without interruption from any other process.

* locked variable: (Busy wait)

- assign a locked variable and switch its value based on the process.
- lets say assign 1 when the process enter critical section and 0 the rest of the time.

* Sleep and Wakeup (Busy Wait)

- if a process is not permitted to access Critical Section it make a system call i.e. Sleep. Then the process gets blocked and not scheduled, until another process call Wakeup.
- Wakeup is called when a process leaves Critical section if any other process have blocked.

Q. list of Parallel Processing Paradigm.

Q. four Parallel Processing Paradigm.

Parallel Processing Paradigm:

- Parallel System are more difficult to Program.
- The processes of multiple CPUs must be co-ordinated and Synchronized.
- The difficult part are CPUs.

Based on the no. of instruction and data Stream that can be Processed simultaneously, computing system are classified into four major categories.

flay

flynn's Taxonomy:

1. Single Instruction- Single Data
2. Single Instruction - Multiple Data
3. Multiple Instruction- Single Data
4. Multiple Instruction- Multiple Data.

* SISD:

- uniprocessor machine.
- capable of executing single instruction operation on a single Data Stream.
- Instruction are processed in Sequential manner.
- Computer adopting this model are called Sequential Comp.
- All the Instruction and data to be processed have to be stored in primary Memory.

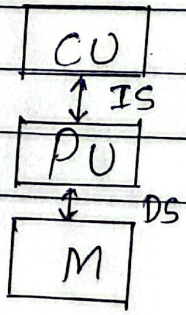


fig: SISD.

* SIMD:

- multiprocessor machine.
- Can perform (execute) same instruction on all CPU operating on different data stream.

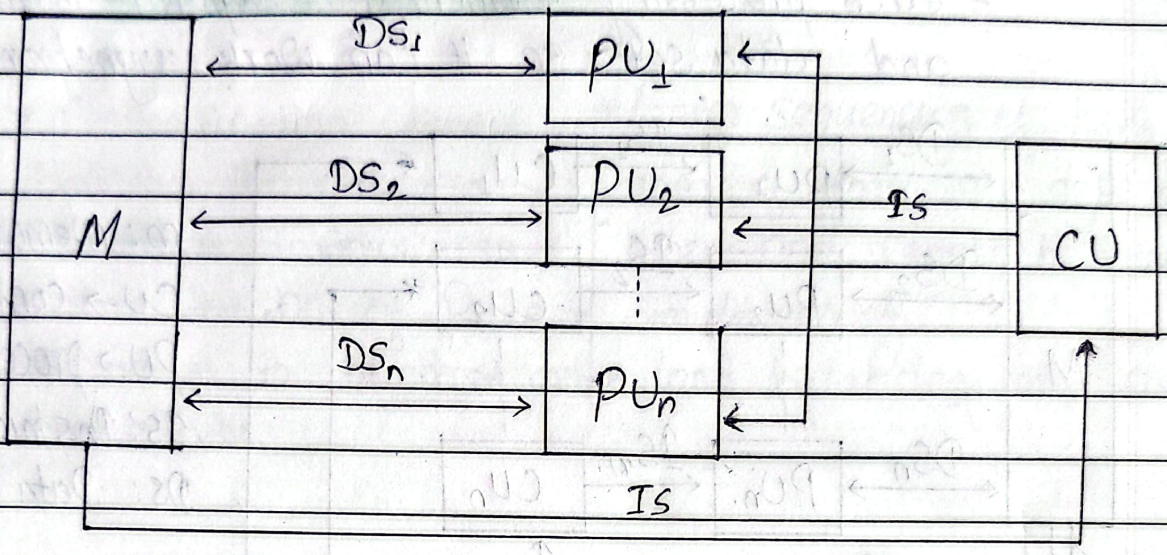
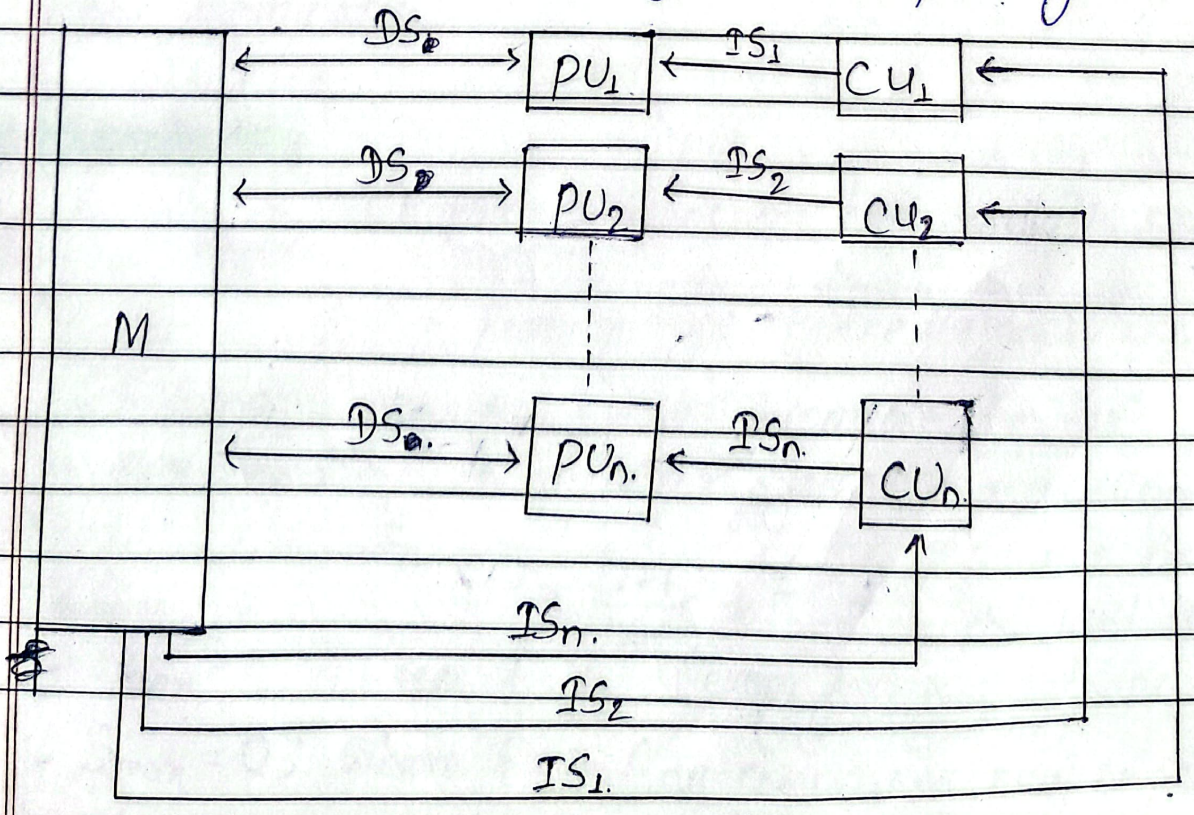


fig: SIMD.

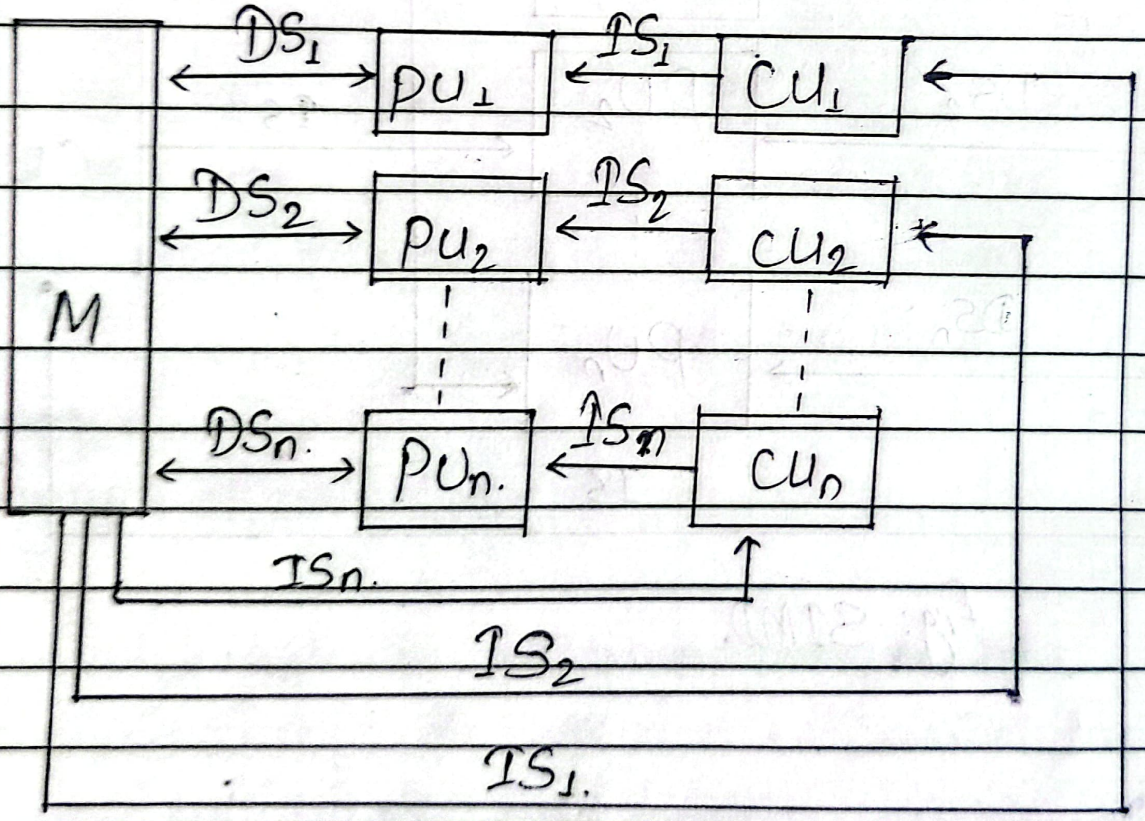
* MISD:

- multiprocessor Machine.
- Can ~~proc~~ perform (execute) multiple Instruction ~~on~~ (different Inst.) on different processing element operating on Same Data.



* MIMD:

- multiprocessor machine
- capable of executing multiple instruction on multiple data set.
- each processing element has separate instruction and data set so it can work asynchronously.



m: Memory.
CU → Control unit
PU → Processing unit
IS: Instruction Stream
DS: Data Stream.

Issues related to Parallelization:

Issues related to parallelization that do not arise in Sequential Programming are:

* Task Allocation.
* Communication Time.

* Communication time between the Processors. * becomes Serious issue when the no. of Processor increases to hundred or thousands.	* Proper sequencing of tasks when some of them are dependent to other and cannot be executed simultaneously. * load balancing or Scheduling.

Cost vs Performance Evaluation:

1. Execution Time:

- Time elapsed from the moment the algorithm start to the moment it terminates.
- In case of multiple Processor, it is the time elapse between the time the first processor begins and the last one terminates.

2. Speed up:

$$\text{Speed} = \frac{\text{Running Time of the best available sequential algorithm.}}{\text{Running time of the parallel algo.}}$$

- It is not always possible to decompose the task.
- Max Speed of N processor System in executing algo:

$$S_N \leq \frac{1}{f + \frac{(1-f)}{N}} \leq \frac{1}{f}$$ ∴ $f \rightarrow$ fraction of task that must be done sequentially

- $S_{max} = 1$; Where $f = 1$. (every task must be done sequentially.) (no speed up)
- $S_{max} = 0$; Where $f = 0$ (all computation must be parallel) (full speed)

3. Communication Penalty:

$$CP_i = \frac{E_i}{C_i}$$

where; E_i = total execution time spent by P_i

C_i = total time used for communication by P_i

Unit 1.2

1.2 Semantics of Concurrent Programming

- ~~refers to the~~
- Concurrent programming refers to the activation of two or more instruction at the same time.
- In programming language, semantics is the field of concerned with the mathematical study of the meaning of the programming language.
-
- Syntatic structure vs Semantic structure:
  - Subject + verb + object (grammatical pattern.

    * I eat rice (syntactically & semantically correct)
    * I rice eat (syntactically wrong).
    * I eat car (syntactically correct but semantically wrong)

1. level of granularity.
2. Sharing the clock.
3. Sharing the memory.
4. Pattern of interaction.

① ~~level of granularity~~:
② When several process operates concurrently, ~~and~~ it is important to determine whether they share same memory. i.e access to shared memory must be mutually exclusive.

When several processes operates concurrently, it is important to determine whether they share the common clock.

There exist two forms of interaction between processes:
A. Synchronization.
B. Communication.

A. Synchronization:

- defines chronological order between events taking place within different processes.
- two pattern of Synchronization:
  1. Mutual exclusion.
  2. Mutual admission.

B. Communication:

- defines transfer of information from one process to another.
- two models of communication.
  1. Synchronous communication.
  2. Asynchronous communication.

Synchronous communication	Asynchronous communication
All parties involved in the communication are present at the same time. eg. (phone call)	All parties doesnot requires to be present at the same time during comm.n eg: (E-mail.)

mutual exclusion

- defined by encapsulated sequence of actions whose execution is indivisible. - once a process starts executing this sequence, it may not be interrupted until the sequence is completed

mutual admission:

- defined by an encapsulated sequence of actions which must be executed by two process simultaneously if one process is ready and the other is not; it must wait until both are ready to give their individual attention.

8. Very Imp:

Date
Page

Semantic Definitions:

Three broad techniques of programming language Semantics:
1. Axiomatic Semantic Definition.
2. Operation Semantic Definition.
3. Denotational Semantic Definition.

1. Axiomatic Semantic Definition:

- defines the meaning of language construct by making statements in the form of axioms or inference rule.
- based on the Hoare's program, which is in the form of: (Hoare Triple)

$\{P\}\ S\ \{Q\}$ where, P → pre-condition.
Q → post-condition.
S → statement.

- If S statement is executed in state where p holds, then it terminates and Q holds.

eg:

① $\{x = 5\}$ $x = x + 1$. $\{x = 6\}$
P. True. S. Q must be true

② $\{J = 3 \text{ AND } K = 4\}$ $J = J + K$ $\{J = 7 \text{ AND } K = 4\}$

③ $\{x \geq 0\}$ While $(x != 0)$ do $x = x - 1$ $\{x = 0\}$.

④ $\{x \leq N - 1\}$. $x = x + 1$ $\{x \leq N\}$.

- Pre-condition defines the state of the program before execution
- Post-condition defines the state after the execution.
- P and Q are predicates that holds boolean.

Q. Partial correctness
Q. Total correctness

Partial Correctness:

- a program is partially correct with respect to pre-condition and post-condition provided that if the program is started with the values that makes the pre-condition True, and the resulting value makes the post-condition True, when the program halts (if ever).

Total Correctness:

- It requires that the algorithm terminates.

Sequence Statement Rule.

The semantics of sequence statement is defined by means of following rules:

- has the form:

$$\frac{H_1, H_2, \ldots\ldots H_n.}{H}$$

Where if, $H_1, H_2, \ldots H_n$ have all been verified, then, H is also valid.

eg.

$$\frac{\{P\}\ S_1\ \{Q\},\ \{Q\}\ S_2\ \{R\}}{\therefore \{P\}\ S_1, S_2\ \{R\}}$$

$$\frac{\{x>1\}\quad x=x+1\quad \{x>2\} \qquad \{x>2\}\quad x=x+1\quad \{x>3\}}{\therefore \{x>1\}\quad x=x+1,\ x=x+1\quad \{x>3\}}$$

$\{x>1\}\quad x=x+1\quad \{x>2\}$ (A)

(B) $\{x>0\}\quad x=-x\quad \{x<0\}$

$\{x>1\}\quad x=x+1,\ x=-x\quad \{x<0\}$

B must be A but if both gives equivalent meaning it is ok.

Alternative Statement Rule:

If - THEN:

$$\frac{\{P \text{ and } B\}\ S\ \{Q\} \qquad \{P \text{ and NOT } B\} \supset \{Q\}}{\therefore \{P\}\ \text{IF } B \text{ THEN } S \text{ END IF } \{Q\}}$$

If - ELSE:

$$\frac{\{P \text{ and } B\}\ S_1\ \{Q\} \qquad \{P \text{ and NOT } B\}\ S_2\ \{Q\}}{\therefore \{P\}\ \text{IF } B \text{ THEN } S_1 \text{ ELSE } S_2 \text{ END If } \{Q\}}$$

i.e

Loop:

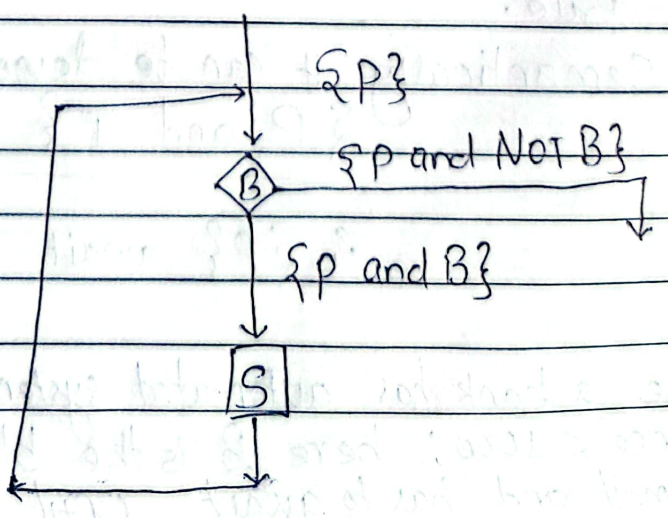
$$\frac{\{P \text{ and } B\}\ S\ \{P\}}{\therefore \{P\}\ \text{While } B \text{ do } S \text{ END while } \{P \text{ and NOT } B\}}$$

eg:

$$\frac{\{x>0 \wedge x<10\}\ \ x = x+1\ \ \{x>0\}}{\therefore \{x>0\}\ \text{while } x<10\ \text{Do } x = x+1\ \{x>0 \wedge x \geq 10\}.}$$

eg.

i.e.



Disjoint Parallel Program:

- two program S_1 and S_2 are said to be disjoint if none of them can change the variable accessed by the other.
- Denoted as $[S_1 \parallel S_2]$

eg:

$S_1 : x = z$	$S_1 : x = y + 1$
$S_2 : y = z$	$S_2 : x = z$
This is	This is not.

- Semantic of this statement can be denoted as:

$$\frac{\{P_1\}\ S_1\ \{Q_1\} \quad \{P_2\}\ S_2\ \{Q_2\}}{\therefore \{P_1 \text{ and } P_2\}\ [S_1 \parallel S_2]\ \{Q_1 \text{ and } Q_2\}}$$

Q. Semantic defⁿ of await-then rule.

Await Then Rule:

- await T then B, where T → condition and B → block of code.
- If condition T holds then block B is executed else the process is suspended until T becomes True.
- Semantically it can be denoted as:

$$\frac{\{P \text{ and } T\}\ B\ \{Q\}}{\therefore \{P\}\ \text{await } T \text{ Then } B\ \{Q\}}$$

- assume a bank has automated system that sends email if balance < 1000; here B is the block of code that sends the email and has to await until balance < 1000.

Operational Semantics:

- how a computation is performed.
- while axiomatic semantic capture the meaning of programming language constructs by
  - focusing on the effect of these constructs on the program's state.
  - operation semantic focuses on how the state of the program is affected.

- uses the notation $\langle P, S \rangle$ means semantics of program P at state S.

i.e.

$\langle z = x, x = y, y = z, [x \to 5, y \to 7, z \to 0] \rangle \Rightarrow \langle P_1, S_1 \rangle$

$\langle x = y, y = z, [x \to 5, y \to 7, z \to 5] \rangle \Rightarrow \langle P_2, S_2 \rangle$

$\langle y = z, [x \to 7, y \to 7, z \to 5] \rangle \Rightarrow \langle P_3, S_3 \rangle$

$\langle , [x \to 7, y \to 5, z \to 5] \rangle$

before starting the program, the state of the memory is $[x \to 5, y \to 7, z \to 0]$ and after the program has terminated, it is $[x \to 7, y \to 5, z \to 5]$, which is what the program says i.e: $z = x, x = y, y = z$.

eg: Syntax directed translation can also be represented:

Production	Semantic Rule.
$E \to E + T$	{ Print("+") }
$E \to E - T$	{ Print("-") }
$E \to T$	{ ~~Print~~ }
$T \to 0$	{ Print(0) }
$T \to 1$	{ Print(1) }
$T \to 9$	{ Print(9) }

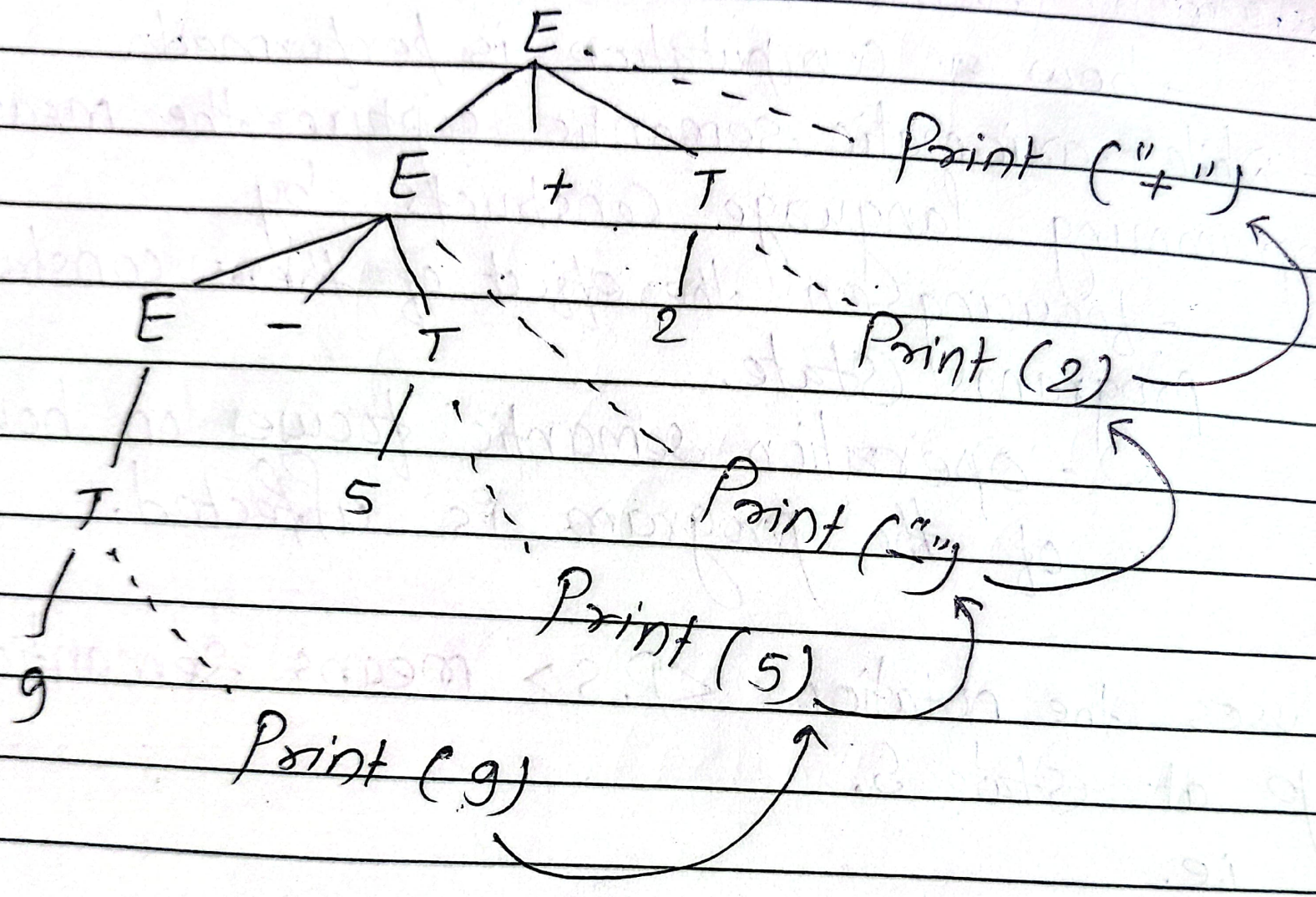
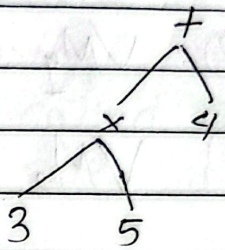


fig: translating 9-5+2 into 95-2+

Denotational Semantic Definition:

- each construct of the language is mapped into a mathematical object that define its meaning.
- the ideas D.S. is to associate an appropriate mathematical object with each phrase of the language.

eg: syntax tree for $3*5+4 \Rightarrow$



- traditionally, the emphatic bracket ($[\![,]\!]$) is used to represent the denotational Semantic definition

eg:

$E_1 + E_2 = plus(\text{Evaluate } [\![E_1]\!], \text{Evaluate } [\![E_2]\!])$

- also the ordered pair can be used to represent the program

eg:

fact (n) = if (n==0) then 1 $\Rightarrow$ can be viewed as $\langle n, n! \rangle$

else, n * fact (n-1).

eg:

$\{ \langle 0,1 \rangle, \langle 1,1 \rangle, \langle 2,2 \rangle, \langle 3,6 \rangle, \langle 4,24 \rangle, \langle 5, \overset{120}{\cancel{625}} \rangle, \ldots \ldots \}$.

Communicating Sequential Process:

- a process P can be represented as $n \rightarrow Q$

where $n \rightarrow$ event

$Q \rightarrow$ process.

ie. P as a Process which first engage in the event x and then behaves exactly as described by process Q.

eg: Vending Machine.

① $VM_0 \xrightarrow{def} (coin \rightarrow (candy \rightarrow stop))$

② $VM_1 \xrightarrow{def} (coin \rightarrow (candy \rightarrow VM_1))$

③ $VM_2 \xrightarrow{def} (coin \rightarrow (candy \rightarrow VM_2) \,||\, (notebill \rightarrow (toffee \rightarrow VM_2)))$.

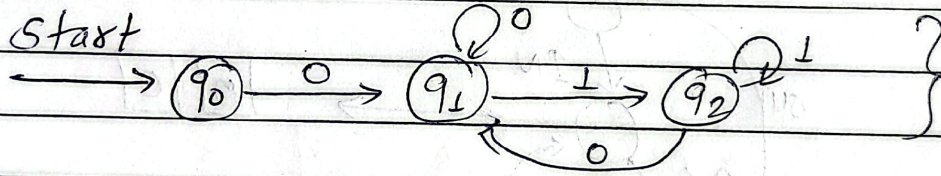
Unit 1.3

1.3 Process Algebra:

- System behavior generally consist of process and ~~algebra~~ data.
- Process are the control mechanism for the manipulation of the data.
- Process graph is the labelled transition system in which one state is selected as root state.
- for the mathematical reasoning, Process graph are expressed algebrically called process algebra

eg:

$0(0+1)^* 1$ → process algebra



} Process graph.

Process algebra studies two types of actions:

1. Observable action
2. Unobservable action.

explicit action

Implicit, without any event denoted by τ (tau)

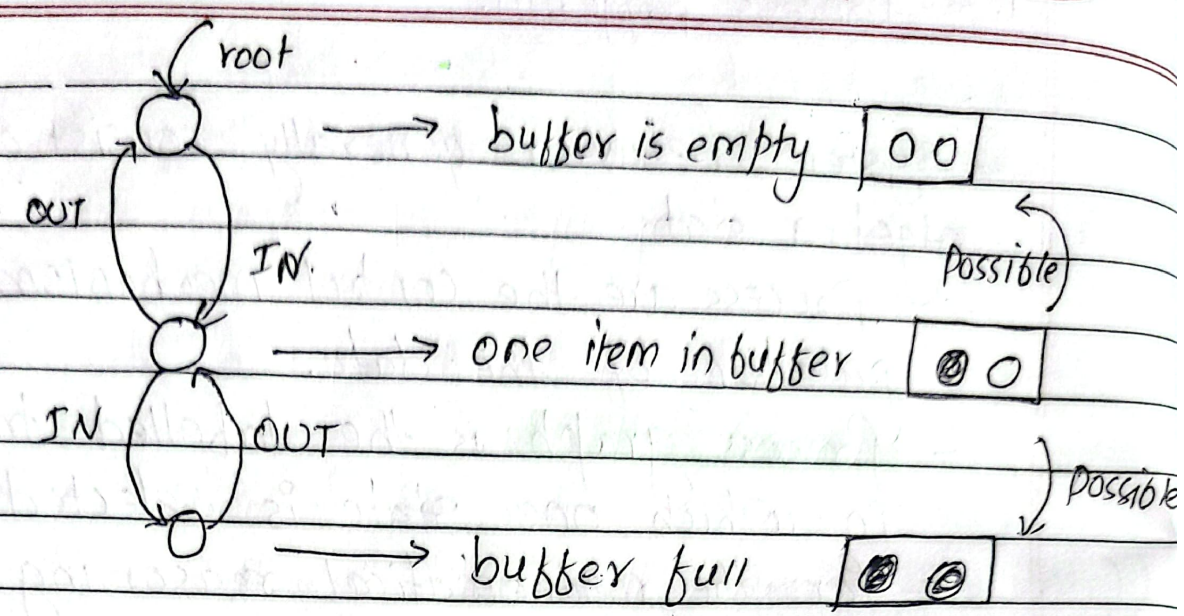


fig: Transition system representing two place buffer.

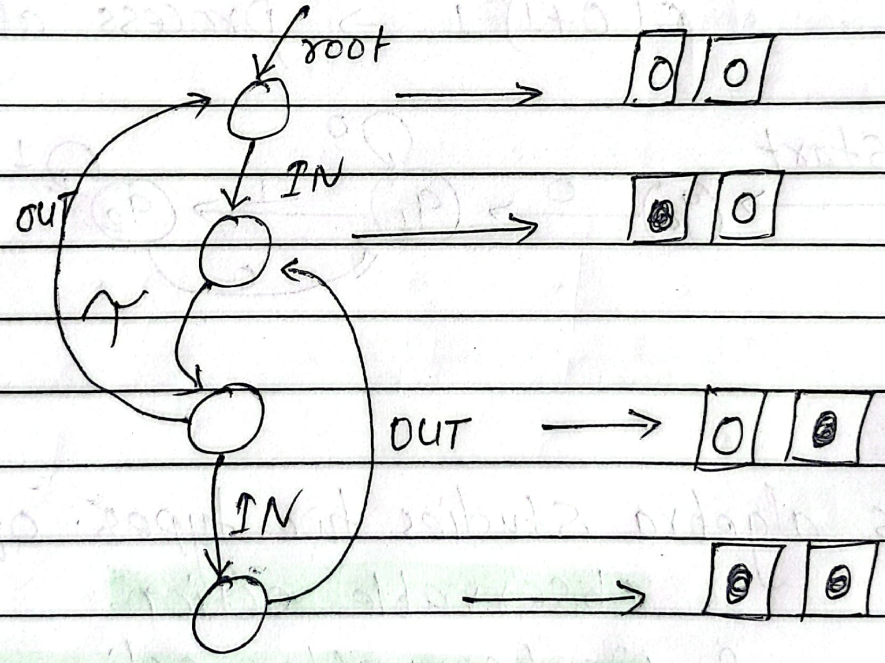


fig: Transition system representing two one place buffer.

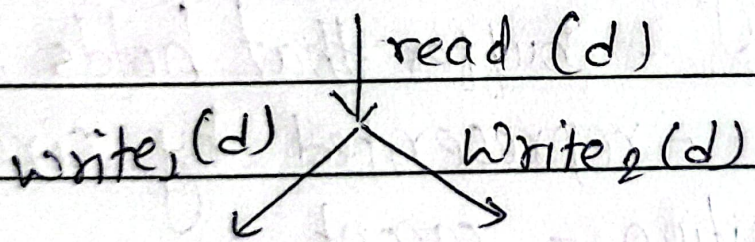
Q. Describe observation Bisimilarity.

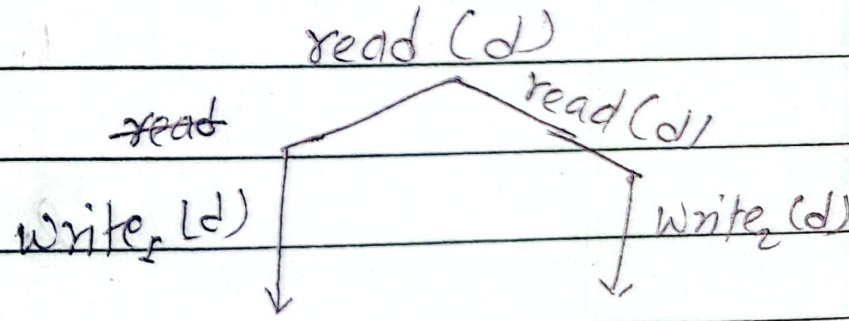
Observation Bisimilarity:

- two transition system A_1 and A_2 are Observation Bisimilar if they shows similar behavior

eg: - two place buffer and two one place buffer are

eg:



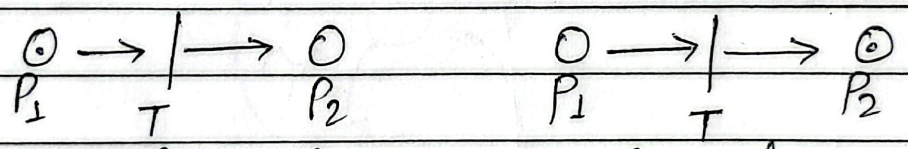


Q. define petrinets.

Petri- Nets:

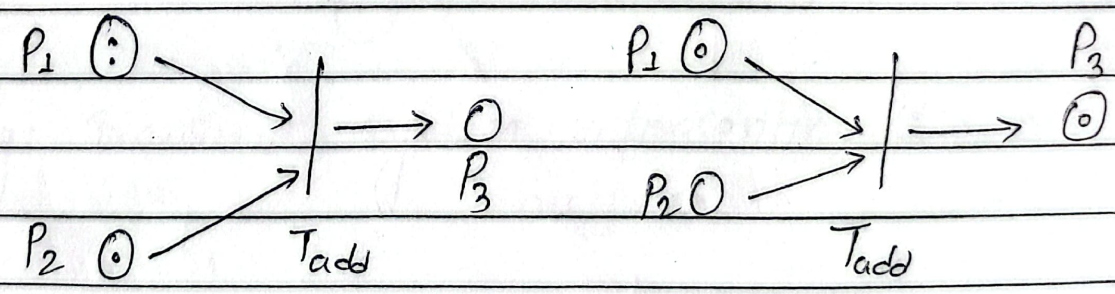
- developed by Carl Adom Petri in 1962
- mathematical modeling language for the description of the system.
- represented through directed graph (bipartite graph)
- Consist of two types of
  1. Place - buffer that holds sth.
     - represented by circle.
  2. Transition - event.
     - represented by straight ling or rectangle.

eg:



before firing of T after firing of T

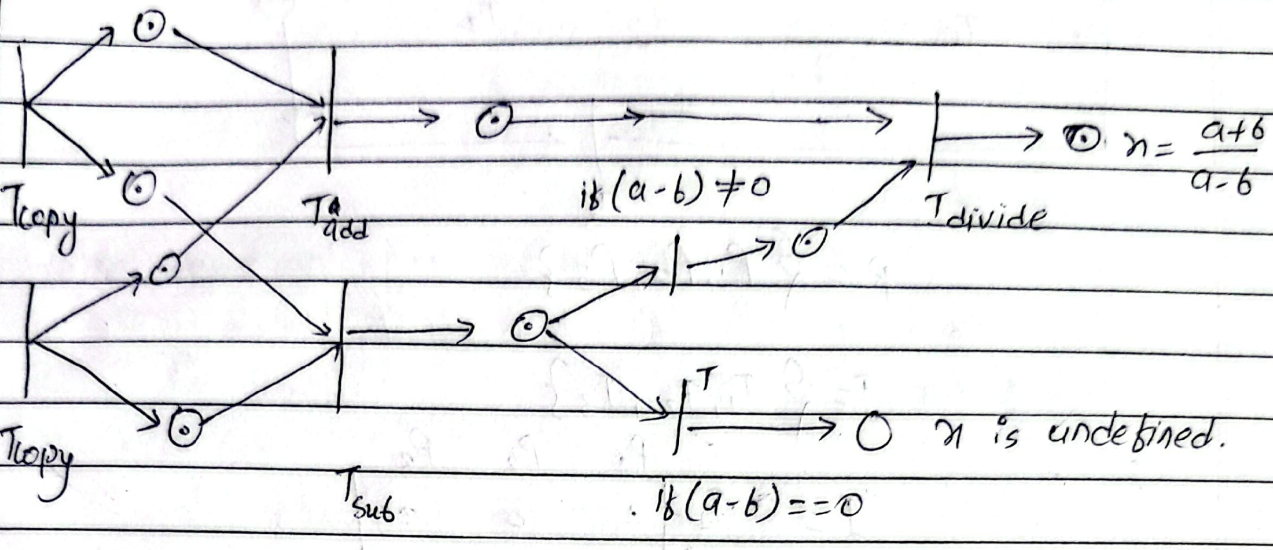
eg:



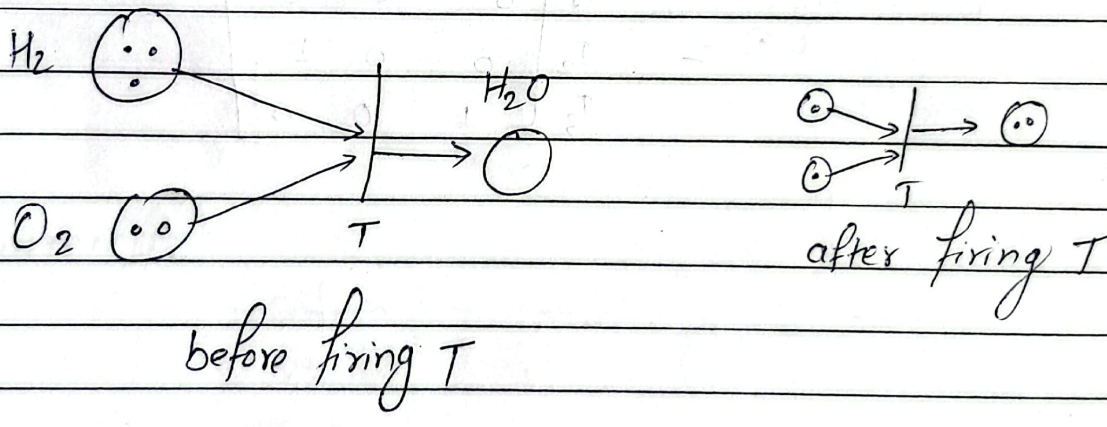
before firing of T_{add} After firing of T_{add}.

Q. Draw petri-Net for $x = \frac{a+b}{a-b}$

eg:



petrinet showing data flow computation ($x = \frac{a+b}{a-b}$)



Q. Formal definition of Petrinets:

formally it can be defined as $PN = (P, T, I, O, M_{to})$

Where

P → set of places = $\{P_1, P_2, \ldots P_m\}$

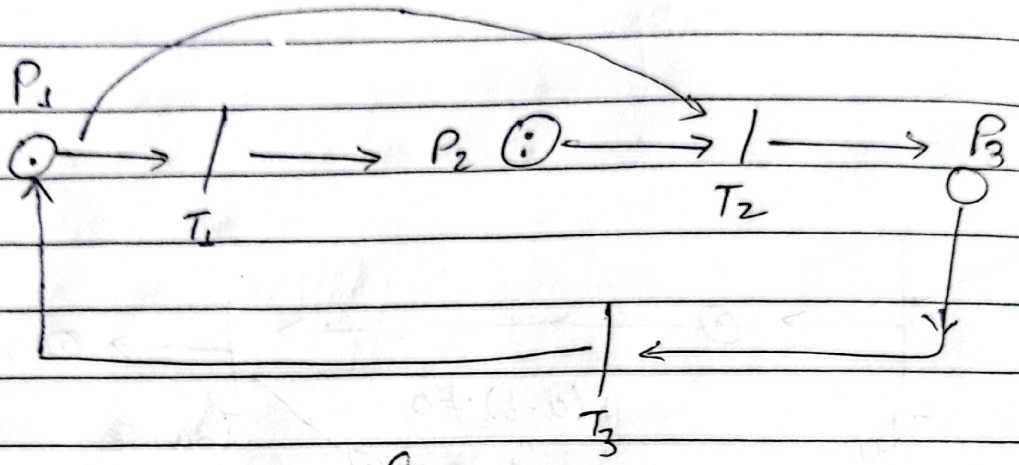
T → set of transition = $\{T_1, T_2 \ldots T_n\}$

I → an nxm input matrix.

O → an nxm output matrix.

M_{to} → represents initial marking of machine.

eg:



$P = \{P_1, P_2, P_3\}$

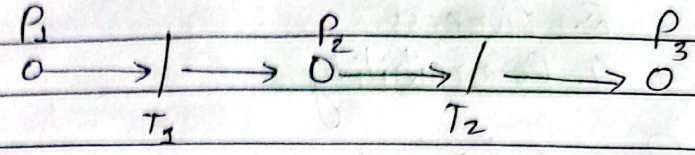
$T = \{T_1, T_2, T_3\}$

$$I = \begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

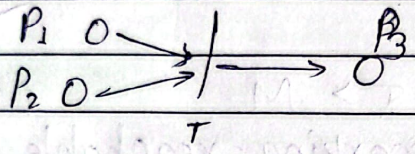
$$O = \begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

Variations of Petri Net:

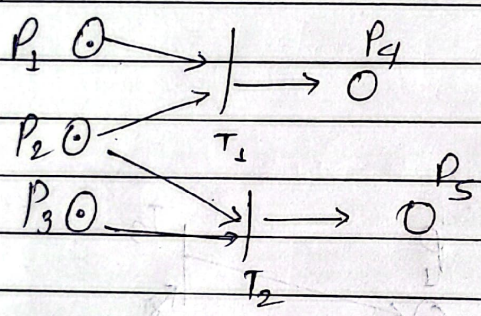
① Sequential Action:



② dependency:

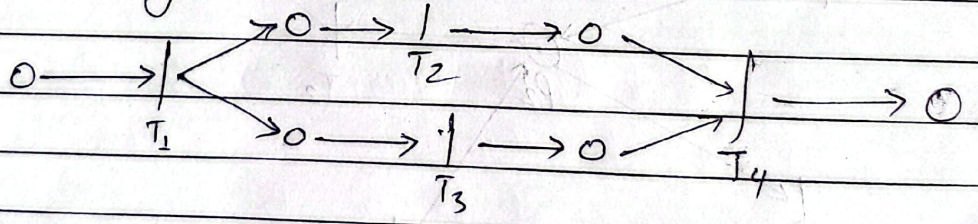


③ Conflict

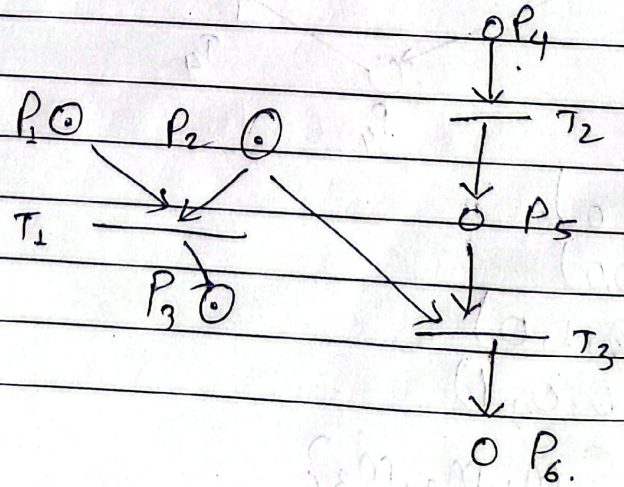


P1 and P2 token → T1 fire.
P2 and P3 token → T2 fire.
all have token → conflict.
Q1ⁿ. Priority.

④ Concurrency:



⑤ Confusion:



Q. Properties of Petri Nets?

Properties of Petri-Nets:

1. Reachability
2. Boundness
3. Liveness
4. Reversibility

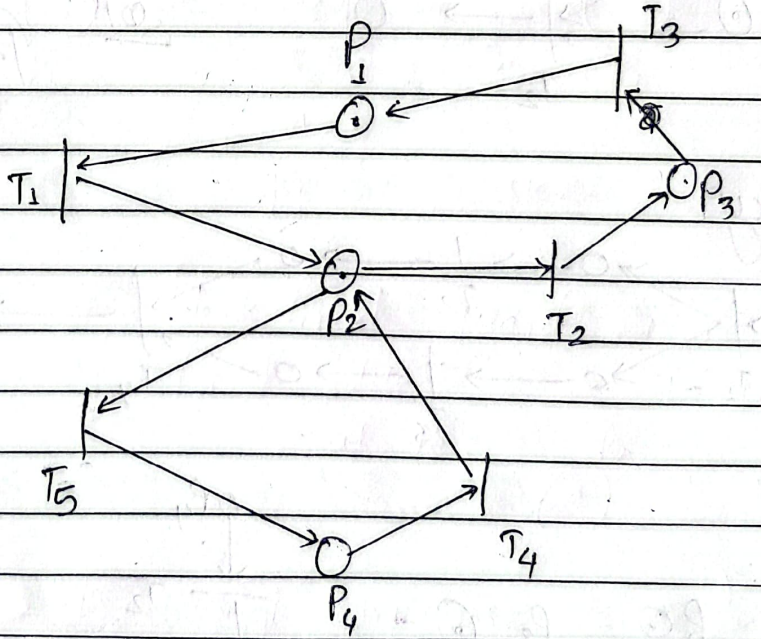
1. Reachability:

- A marking M is said to be reachable from another marking M' if there exist a sequence of transition (σ) such that

$$M' \langle \sigma \rangle M$$

- $R(M_0)$ - set of marking reachable from initial marking M_0

eg.



$M_0 = (1,0,0,0)$

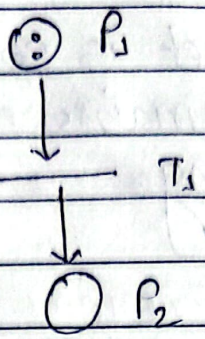
$M_1 = (0,1,0,0)$

$M_2 = (0,0,1,0)$

$M_3 = (0,0,0,1)$

$R(M_0) = \{M_1, M_2, M_3\}$

eg:



$M_0 = (2,0)$

$M_1 = (1,1)$

$M_2 = (0,2)$

$R(M_0) = \{M_1, M_2\}$

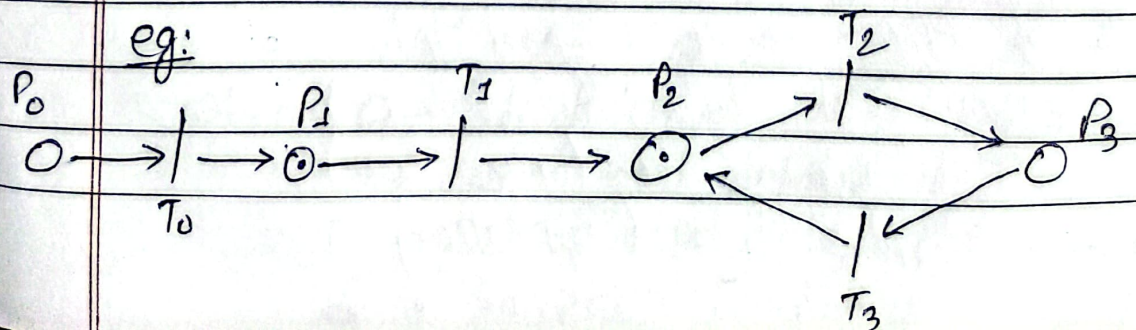
2. Boundness:

- No. of tokens in a place is bounded
- a place P is said to be k-bounded, if the number of tokens in P never exceed k, ie. $M(P) \leq k$
- A petri net is said to be k-bounded if all placess are k-bounded.

3. Liveness:

- A transition is said to be live if it can always be made from any reachable marking.
- a transition is deadlocked if it can never fire
- a transition is said live if it can never deadlocked.
- a transition is said quasi live if it can be fired at any once.

eg:



quasi live.

$T_1 \rightarrow$ ~~dead~~

$T_2 \rightarrow$ live

$T_3 \rightarrow$ live

$T_0 \rightarrow$ dead.

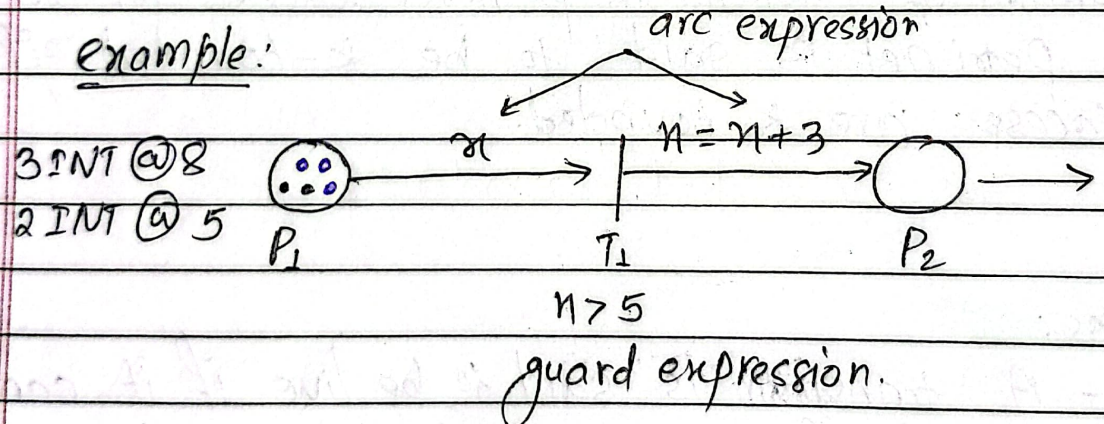
4. **Reversibility:-**

- A petrinet is said reversible, if the initial marking remains reachable from any reachable marking.

High-level Petri-Net:

- consist of following features:
  * Petri-Nets.
  * Marking scheme (like color petrinets)
  * Transition Guard (condition)
  * Arc expression (eqn to be satisfied)

example:



if black fired guard will cover
if fired blue ⟶ x = x+3.
at P2.

1 INT @ 11.

8. **Purpose of coloured Petri nets.**

Q. Basic terms of Complexity theory.

Date
Page

Unit 1.4

Complexity Theory:

- estimates the amount of computational resources needed to solve a problem. (Time and storage)

Basic Terms:

1. Introducing a notion to specify complexity.
2. Choice of machine model to standardize the measure.
3. Refinement of the measure of parallel computation.

Q. Turing machine as a basis of consequences - very good basic machi

Turing Machine as a basis of Consequences:

* Simplicity of Abstraction.
* Dual nature of Turing Machine (TM)
* Inclusion of non-determinism.
* Realization of unbound parallelism.
* Provision of an easy abstract measure.

1. Simplicity of abstraction:

- TM as simple model of an abstract machine that consist of three component:
  a. Finite state machine.
  b. A read/write head.
  c. Infinite tape memory.

Q. formal defⁿ of T.M.

- Mathematically TM can be defined as:

$$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, f)$$

Where, $Q \rightarrow$ Set of state
$\Sigma \rightarrow$ set of input alphabet.
$\Gamma \rightarrow$ Set of tape state.
$\delta \rightarrow$ transition function
$q_0 \rightarrow$ start state.
$B \rightarrow$ Blank symbol.
$f \rightarrow$ set of finite state

Q. Dual nature of TM.

2. Dual Nature of Turing Machine:
- universal Turing machine.
- input consist of program and a dataset for the program to process.

3. Inclusion of Non-Determinism:
NDTM - alternative moves.
- branching tree called OR tree
- a computational halt if there is a sequence of that leads to an accepting state.
- rejects an input if and only if there is no possible sequence of memory.

4. Realization of unbound Parallelism:
- NDTM can be simulated using DTMs by allowing inbounded Parallelism in computation.

5. Provision of an easy abstract measure for Resources:
DTM → by counting each move as one unit time and each cell scanned as one space time
NDTM → Time Same, memory different.

Q. Alternating T.M.

ATM (Alternating Turing Machine)

- State of ATM are partitioned into
  - Universal State (AND tree)
  - Existance State (OR tree)
  - Normal State (Unique)

Parallel Complexity Model:

- Synchronous parallel computation can be used to solve problems like matrix Multiplication.

VLSI Computation Complexity:

- Practical use of algorithm depends upon the design of VLSI
- two fundamentals parameters:
    1. Area of Circuit (space complexity)
    2. Time taken to provide an output for a given input (Time complexity)

Q. Why & when we need leader election.
Q. Ring vs Bully algo

Unit 1.5

Distributed Computing:

- major obstacle in distributed computing is uncertainty due to different Processor speed and occassional failure of components.

Leader election:

- leader can simplify co-ordination among the Processor.
- leader election is the Process of designating a single processor as the leader

Approach:

- Bully algorithm
- ~~Rig~~ Ring Algorithm.

Bully Algorithm:

- biggest guy in town always wins.

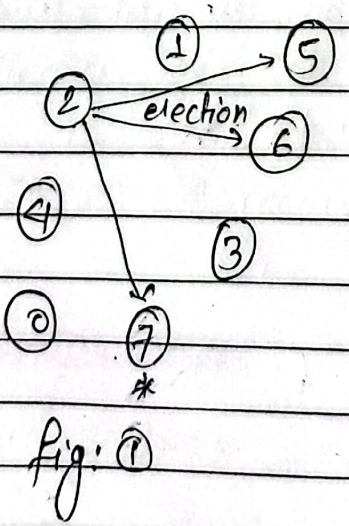
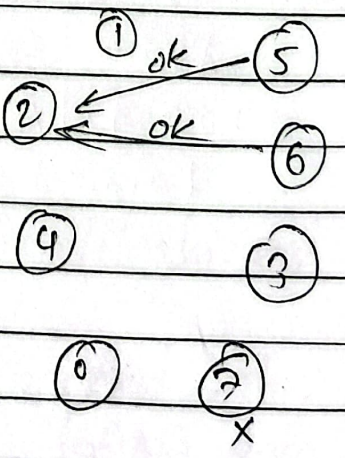


Fig: ①

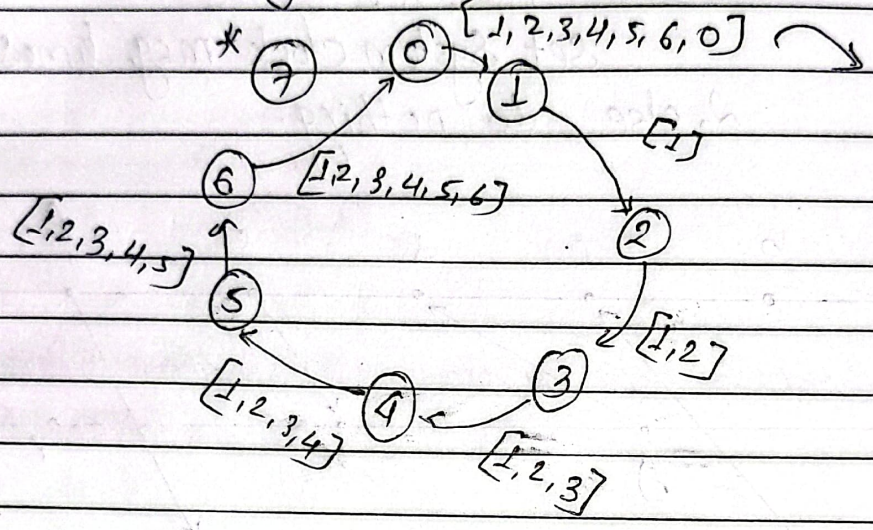
Since 7 is already crashed it doesn't receive the message



Contd.

Ring Algorithm:

- if any process notice that current leader has failed, it start an election by sending message to the first neighbour of the ring.



5. uses of logical clock and vector clock in event ordering.

Ordering of event:

- ordering between the events in the system.

logical Clocking:

- design logical timeStamps to events based on ordering among them.
- if event 'a' happened before event 'b', then it is represented as:

$$a \rightarrow b$$

- if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$
- if $a < b$ then $T(a) < T(b)$

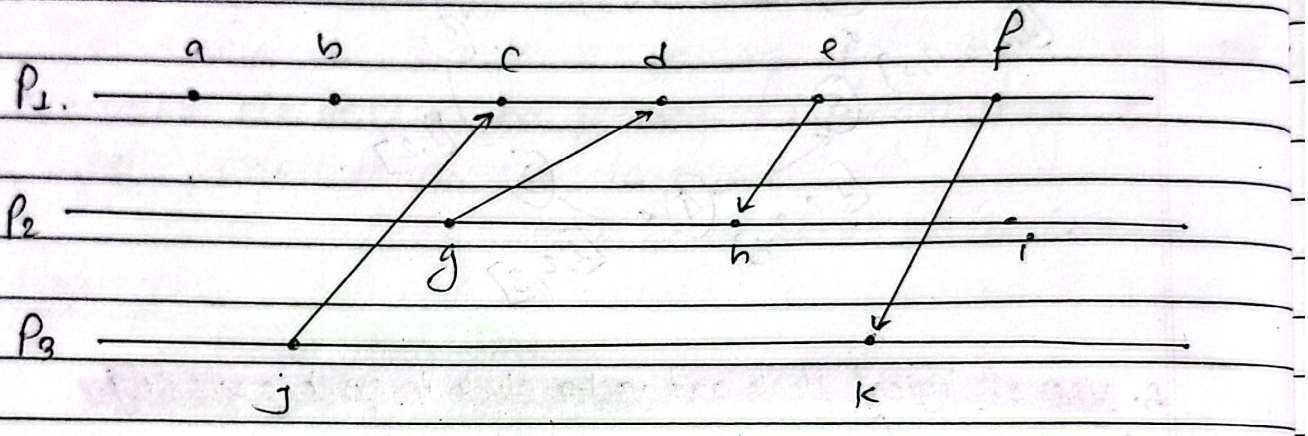
$T \Rightarrow$ Time Stamp.

- if $T(a) < T(b)$ then we guarantee $a \leq b$.

Q. logical clock vs vector clock
Q. Rules of vector clock.

Lamport's logical Clock:

- each message carries a timeStamp of the sender's clock.
- When a message arrives
  1. If receiver's clock is < msg timestamp set system clock msg. timestamp + 1.
  2. else do nothing.

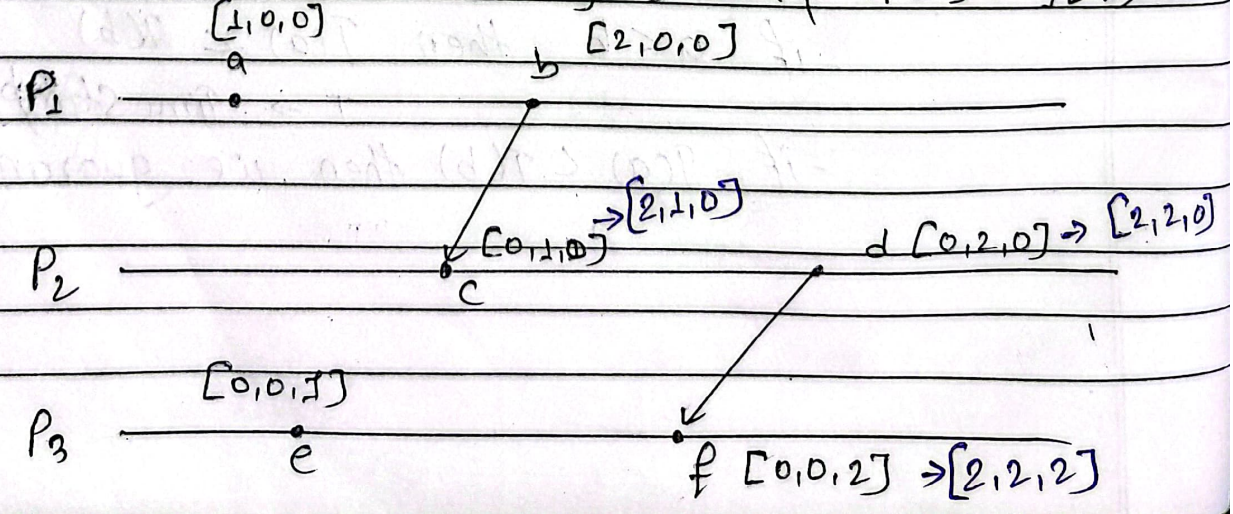


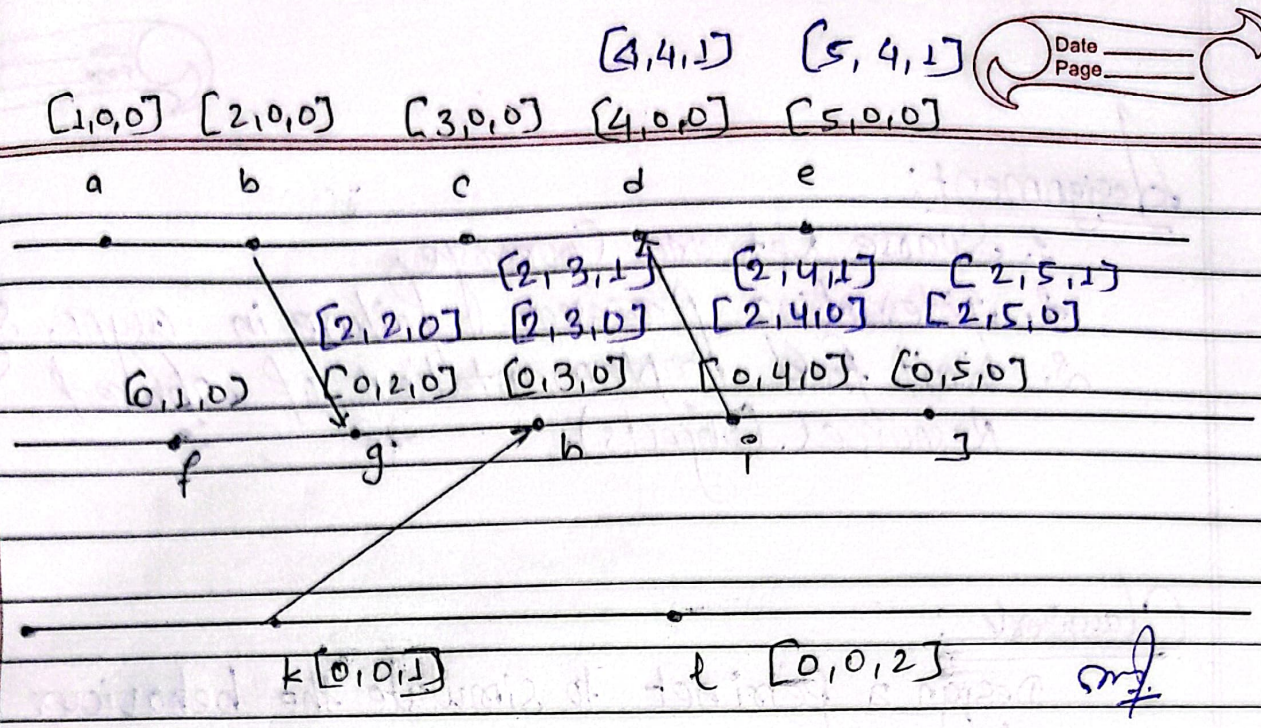
Vector Clock:

Rules:

1. Vector initialization to 0 at each process i.e $V_i[j] = 0$
2. Process increments its element of the local vector as $V_i[i] = V_i[i] + 1$.
3. When P_i receives message, set local vector to higher of two values.
   ie. $V_j[i] = \max(V_i[i], V_j[i])$

eg:





Q. Problem arise in dist. system when processor need to share resources.

* Resource Allocation:

- Problem Arise in distributed computing system when processor need to share resources.
- In such problem the program is partitioned into 4 region.

1. Trying :- where the processor tries to acquire the resources.
2. Critical :- Where the resources are being used.
3. Exit :- Where some cleanup is performed.
4. Remainder :- the rest of code.

Q. list of failure that may occur.

Tolerating Processor failure in Synchronous System.

- It is assumed that communication links are reliable but processor are not reliable.
- Two types of failure:
  1. Crash failure
  2. Byzantine failure

Unit 2-1

PRAM Model:

Parallel Random Access machine Model.
Technique for increasing the computation speed of a task.

Eg: finding the smallest value in the set of N values.

Algorithm:

- Do in parallel
1. P_i derives the pair (i_1, i_2) of indeces
2. P_i reads value $L(i_1)$ and $L(i_2)$
3. If $L(i_1) \geq L(i_2)$ then P_i sends negative outcome to P_{i_1} else P_i sends negative outcome to P_{i_2}.
4. Only one processor P_j, $1 \leq j \leq n$, which didnot receive a negative outcomes.
5. P_j reads the value of $L(j)$ and Write to the output cell

Eg:

$L = \{2, 6, 4, 8\}$

No. of processor $\downarrow$ $\frac{n(n-1)}{2} = 6$

$P_1 = (1,2) \rightarrow P_1$ sends -ve outcome to P_2.
$P_2 \rightarrow (1,3) \rightarrow P_2$ sends -ve outcome to P_3
$P_3 \rightarrow (1,4) \rightarrow P_3$ sends -ve outcome to P_4
~~$P_4 \rightarrow (2,1) \rightarrow P_4$ sends -ve outcome to P_2~~
$P_4 \rightarrow (2,3) \rightarrow P_4$ sends -ve outcome to P_6
$P_5 \rightarrow (2,4) \rightarrow P_5$ sends -ve outcome to P_4
$P_6 \rightarrow (3,4) \rightarrow P_6$ sends -ve outcome to P_4

- (i_1, i_2) is assigned on P_i when $i = \frac{(i_2-1)(i_2-2)}{2} + i_1$

eg: $(2,4) = \frac{(4-1)(4-2)}{2} + 2 = \frac{3 \times 2}{2} + 2 = 5.$

2. $L = \{3, 2, 5, 4\}$

Soln:

$$L = \{3, 2, 5, 4\}$$
(1 2 3 4)

$$N = \frac{4(4-1)}{2} = 6$$

$P_1 = (1,2)$ = P_1 sends -ve outcome to P_2.

$P_2 = (1,3)$ = P_2 sends -ve outcome to P_3

$P_3 = (1,4)$ = P_3 sends -ve outcome to P_4

$P_4 = (2,3)$ = P_4 sends -ve outcome to P_3

$P_5 = (2,4)$ = P_5 sends -ve outcome to P_4

$P_6 = (3,4)$ = P_6 sends -ve outcome to P_4

Q. Types of PRAM

PRAM Model:

① EREW (Exclusive Read Exclusive Write)
- Every memory cell can be read or written to by only one processor at a time

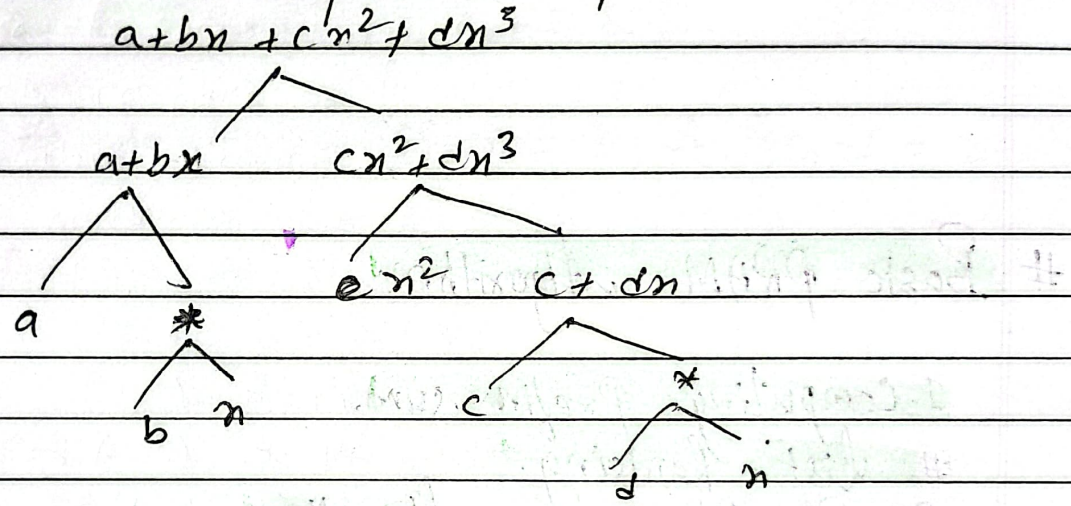
② CREW (Concurrent Read Exclusive Write)

③ ERCW (Exclusive Read Concurrent Write)

④ CRCW (Concurrent Read Concurrent Write)

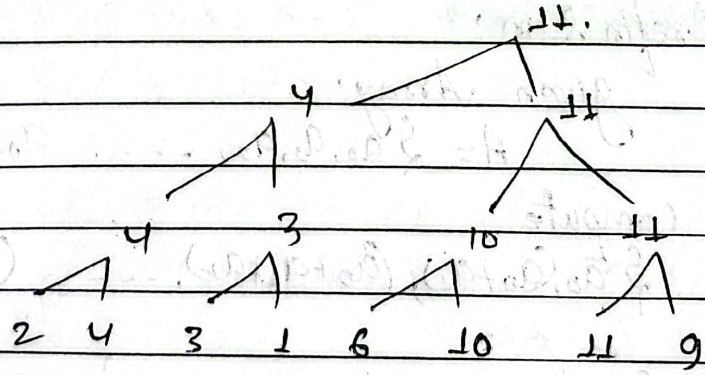
Technique for the design of parallel algorithm:

① Divide and conquer technique.



② Balanced Binary tree method

max (2, 4, 3, 1, 6, 10, 11, 9) = ?



Optimality and Efficiency of Parallel algorithm.

- let A be a problem of size N;
- Assume that problem A can be solved by ~~prat~~ parallel algorithm P_A in Time T(N) by employing P(N) processor.

i.e: W(N) = T(N) * P(N)

Two most important Criteria for evaluation of parallel algorithms are speed up and efficiency of parallel algorithm.

where,

$$\text{Speedup} = \frac{t(N)}{T(N)}$$

$t(N)$ = sequential time

$T(N)$ = parallel time

$$\text{Efficiency} = \frac{\text{Eff Speed up}}{P(N)}$$

Basic PRAM Algorithm:

I. Computing Prefix sum.
II. List Ranking.
III. Parallel Sorting Algorithm.

1. Computing Prefix Sum:

given Array:

$$A = \{a_0, a_1, a_2, \ldots a_{n-1}\}.$$

we have to compute

$$\{a_0, (a_0+a_1), (a_0+a_1+a_2), \ldots (a_0+a_1+\ldots+a_{n-1})\}.$$

Example:

$$A = \{5, 3, -6, 2, 7, 10, -2, 8\}$$

output = $\{5, 8, 2, 4, 11, 21, 19, 27\}$

Algorithm: Parallel prefix sum:

- Input an array. $[a_1, a_2, \ldots a_n]$
- If $n == 1$; return $a[1]$ $S_1 \leftarrow a[1]$

else 1. for $j = 0$ to $\log n - 1$ do

1.1. for $i = 2^j + 1$ to n do in parallel

P_{i-2^j} reads $a[i - 2^j]$

$a[i] = a[i - 2^j] + a[i]$

Example:

1	2	3	4	5	6	7	8
5	3	-6	2	7	10	-2	8

$A = [5, 3, -6, 2, 7, 10, -2, 8]$

Here, $n = 8$ so $\log n = \log_2 8 = 3$ & $\log n - 1 = 3 - 1 = 2$

So,

for $j = 0$ to 2 we have $j = \{0, 1, 2\}$

case ① $j = 0$

for $i = 2^j + 1$ to n $= 2^0 + 1$ to 8 $= 2$ to 8

$a[i] = a[i - 2^j] + a[i]$ where $i = 2$ to 8 so,

① $a[2] = a[2 - 2^0] + a[2] = a[1] + a[2] = 8$

② $a[3] = a[3 - 2^0] + a[3] = a[2] + a[3] = -3$

③ $a[4] = a[4 - 2^0] + a[4] = a[3] + a[4] = -4$

④ $a[5] = a[5 - 2^0] + a[5] = a[4] + a[5] = 9$.

⑤ $a[6] = 17$

⑥ $a[7] = 8$

⑦ $a[8] = 6$.

new array after iteration 1 $= [5, 8, -3, -4, 9, 17, 8, 6]$

case ⑪ for $j = 1$. $i = 2^j + 1$ to $n = 3$ to 8.

$a[3] = a[3 - 2^1] + a[3] = a[1] + a[3] = 2$

$a[4] = a[4 - 2^1] + a[4] = a[2] + a[4] = 4$

$a[5] = 6$

$a[6] = 13$

$a[7] = 17$

$a[8] = 23$

New array =

$[5, 8, 2, 4, 6, 13, 17, 23]$

Case III

[5, 8, 2, 4, 6, 13, 17, 23]

J = 2

i = $2^J + 1$ to n = 5 to 8.

$a[i] = a[i-2^J] + a[i]$

① $a[5] = a[5-2^2] + a[5] = a[1] + a[5] = 11.$

$a[6] = 21.$

$a[7] = 19$

$a[8] = 27.$

So after third iteration new array is.

A = [5, 8, 2, 4, 11, 21, 19, 27]

Which is our final Answer.

* List Ranking:

- The problem is that given a singly linked list with L, with N objects, for each node, compute the distance to the end of the list.
- If d denotes the distance,

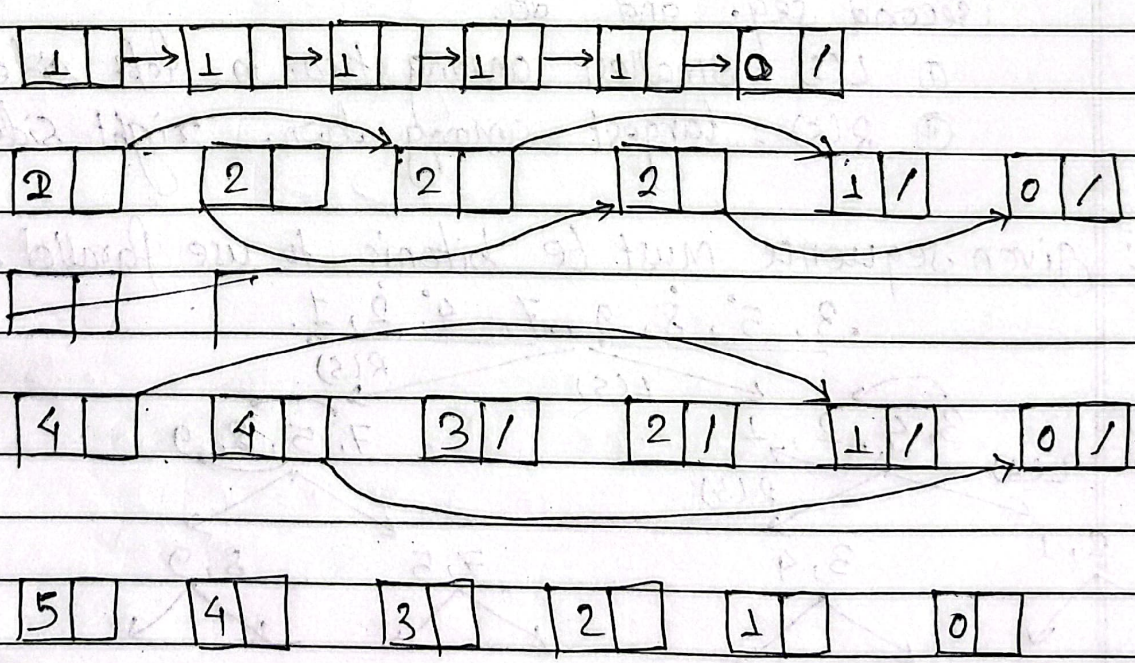
$$node.d = \begin{cases} 0 & \text{if } node.next = null \\ 1 & \text{other wise.} \\ (node.next.d + 1) & \end{cases}$$

Parallel Algorithm

- assign one processor for each node.
- for each node i, do in parallel
  1. i.d = i.d + i.next.d
  2. i.next = i.next.next

  where i = index of n
  d = distance.

eg. let no. of processor = 6.



Q. Bitonic Sort Numerical

Parallel Sorting Algorithm:

Bitonic Sort

- consist of two sequences, one increasing and one decreasing.

eg: 5, 6, 7, 8, 4, 3, 2, 1.
(1st seq. 2nd seq.)

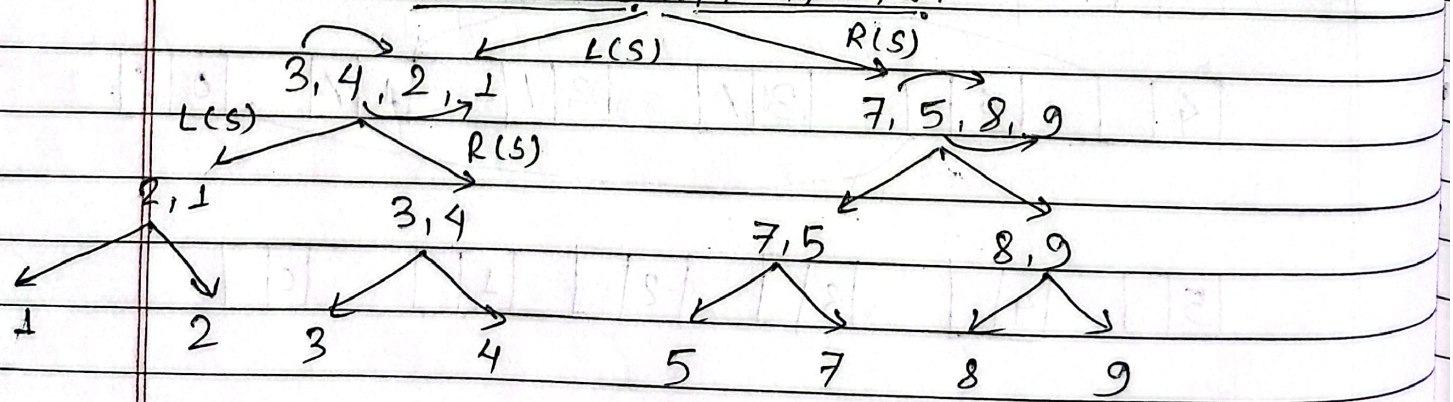
- It is a sequence of elements $\langle a_0, a_1 \ldots a_{n-1} \rangle$ there exist $\langle a_0, a_1 \ldots a_i \rangle$ as increasing sequence and $\langle a_{i+1}, a_{i+2}, \ldots a_{n-1} \rangle$ as decreasing.
- If $n/2$ is the beginning of decreasing sequence S, then,

$$L(S) = \{ \min(a_0, a_{n/2}), \min(a_1, a_{n/2+1}) \ldots, \min(a_{n/2-1}, a_{n-1}) \}$$

$$R(S) = \{ \max(a_0, a_{n/2}), \max(a_1, a_{n/2+1}) \ldots \max(a_{n/2-1}, a_{n-1}) \}$$

Compare first element of 1st seq. with first element of second seq. and do.
① L(S) = Smallest among two. in left side.
② R(S) = largest among two. right side.

Note: given sequence must be bitonic to use parallel sorting algo.



So the Sorted seq. = 1, 2, 3, 4, 5, 7, 8, 9

Q. Imp. of randomized algo.

Date
Page

Randomized algorithm:

- is the algorithm execution is controlled at one or more points by randomly made choices.

eg:

Primality testing.

fermat's little theorem

i.e. $b^p = b \pmod p$

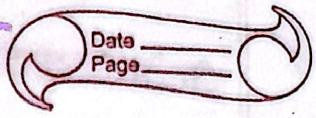
where $b \in Z_p$

Deficiencies of PRAM algorithm:

- No mechanism for representing communications between the processors.
- storage management issues are hidden from algo designer

Q. Define BSR

2.2. Broadcasting with selective Reduction.



BSR

- It is a model of parallel computation in which N processors shares M memory locations.

Broadcast instruction consist of three phases:

* Broadcast phase,
* Selection phase,
* Reduction phase,

Broadcast Phase:

- allow all of the N processor to write concurrently to all the M memory locations.
- each processor P_i, $1 \leq i \leq N$, Produced a record containing 2 fields a tag t_i and datum d_i to be stored.

Selection Phase:

- ~~allow all~~ after the data are received at each memory locations U_j, $1 \leq j \leq m$, a Switch S_j will select the receiving data d_i by comparing tag value t_i by using a selection rule.

$$\nabla \; [<, \leq, =, \geq, >, \neq]$$

Reduction Phase:

- Selected data are reduced to a single value using reduction rule.

$$R\{\Sigma, \Pi, \wedge, \vee, \oplus, \underset{\max}{\min}, \cap, \cup\}$$

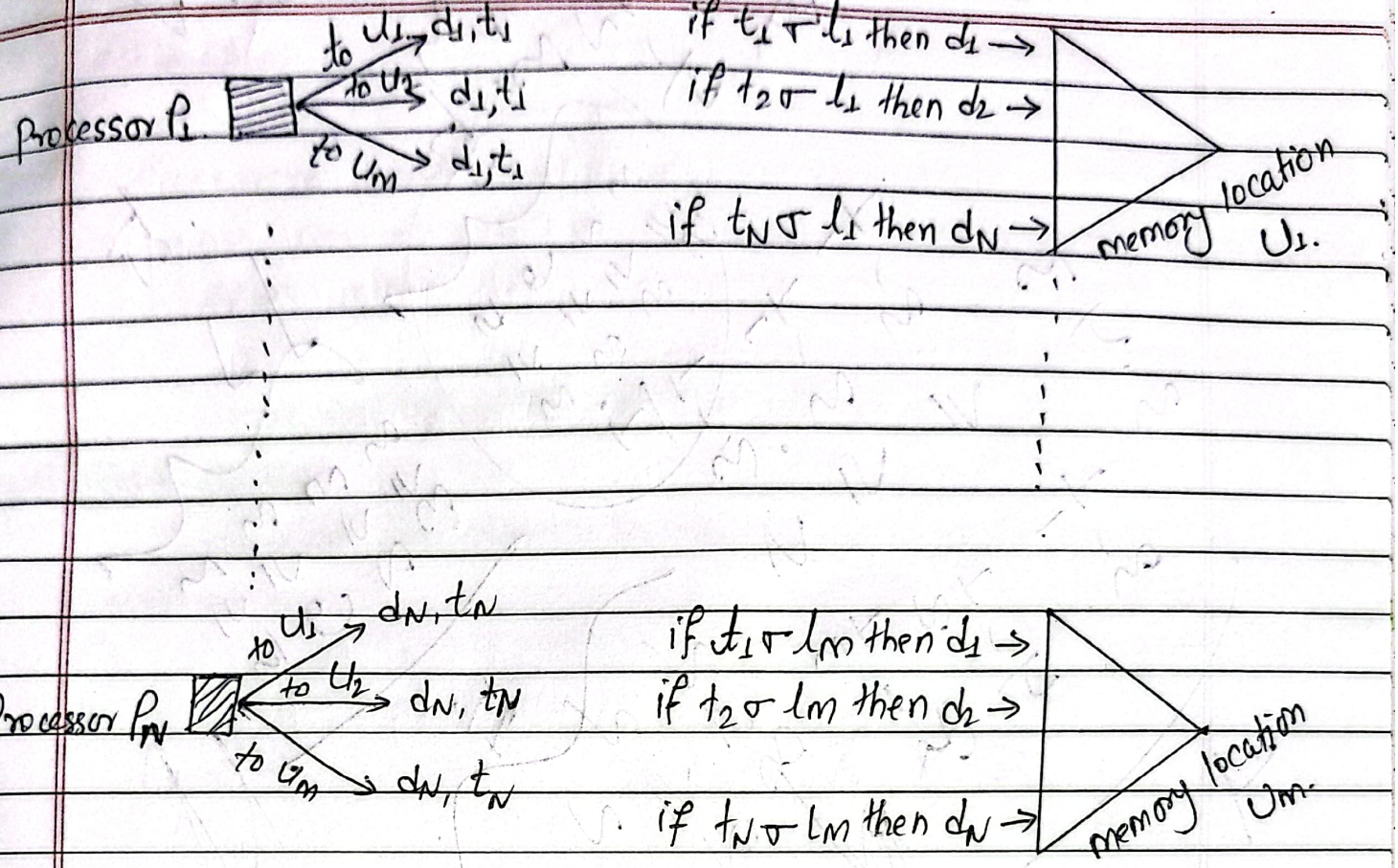


fig: Broadcasting with selective Reduction.

formal definition of Generalised BSR model:

$$U_i \leftarrow R d_i \mid \wedge t(i,h)\, \sigma_h\, l(j,h)$$

$$i \le j \le M, \quad 1 \le i \le N \quad 1 \le h \le k \quad 1 \le j \le M$$

$N \rightarrow$ no. of Processors.
$d_i \rightarrow$ datum broad cast by processor P_i
$\sigma_h \rightarrow$ Selection operation $1 \le h \le k$ (k = no. Criteria)
$t(i,h) \rightarrow$ tag broadcast by P_i for Criteria h.
$l(j,h) \rightarrow$ limit value j for Criteria σ_h.
$R \rightarrow$ reduction operation $R = \{\Sigma, \Pi, \wedge, \vee, \oplus, \cap, \cup\}$.
$U_i \rightarrow$ Memory location to Store Single reduced value.

Q. Sort given Array.

Sorting:

- There are three broadcast instructions.
- first gives rank r_j of element x_j in array $x_1, x_2, \ldots x_n$.
- Number of processor needed = no. of elements in the array.

rank $r_j = \Sigma_1 \{ x_i \leq x_j \}$

Case I

When all elements in the array is distinct.

example:

$\{3, 2, 6, 5\}$
P_1, P_2, P_3, P_4

$i \leq j$

i = value in list
j = value choosen

$P_1 \Rightarrow 3 = 3 \leq 3, \cancel{2 \leq 2}, \cancel{6 \leq 6}, \cancel{5 \leq 5}$
$1 \quad 1 \quad 0 \quad 0 \Rightarrow 2$

$P_2 \Rightarrow 2 = 2 \leq 2, \cancel{3 \leq 2}, 5 \leq 2, 6 \leq 2$
$1 \quad 0 \quad 0 \quad 0 \Rightarrow 1.$

$P_3 \Rightarrow 6 = 2 \leq 6, 3 \leq 6, 5 \leq 6, 6 \leq 6$
$1 \quad 1 \quad 1 \quad 1 \Rightarrow 4$

$P_4 \Rightarrow 5 = 2 \leq 5, 3 \leq 5, 5 \leq 5, 6 \leq 5$
$1 \quad 1 \quad 1 \quad 0 \Rightarrow 3$

Sorted data $\Rightarrow \{2, 3, 5, 6\}$

Case II,

When there are duplicate element.

$\cancel{\{4, 4, 5\}}$ $\{3, 4, 3\}$

$S_j = \Sigma_1 \, t_i \leq t_j$ where $t_i = x_i - \frac{1}{i}$, $t_j = x_j - \frac{1}{j}$

i) $A[1] = 3$;

$t_1 = 3 - \frac{1}{1} \leq t_1 = 3 - \frac{1}{1} \rightarrow$ True.
$t_2 = 4 - \frac{1}{2} \leq t_1 = 3 - \frac{1}{1} \rightarrow$ false
$t_3 = 3 - \frac{1}{3} \leq t_1 = 3 - \frac{1}{1} \rightarrow$ false

} rank 1.

Q. Parenthesis matching algo. Numerical

Parenthesis Matching Algorithm:

→ One Criterion BSR algorithm
→ The problem is to find the pairs of matching Parenthesis in a given legal sequences $t_1 \ldots t_n$ of parenthesis.
→ legal means every parenthesis must have its matching parenthesis in the sequence

- for each Processor j do in Parallel

1. If t_j = '(' then $b_j = 1$ else $b_j = -1$
2. $P_j = \Sigma\, b_j \mid i \leq j$
3. If $b_j == -1$ then $P_j = P_j + 1$.
4. $P'_j = P_j - \frac{1}{j}$
5. $q_j = -1$, $t_j = 0$, $r_j = 0$
6. $q_j = \cap\, P'_i \mid P'_i < P'_j$
7. $t_j = \cap\, i \mid P'_i = q_j$
8. $r_j = \cup\, i \mid t_i = j$
9. If $t_i == $ ')' then $M_j = t_j$ Else, $M_j = r_j$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
Sequence	(	)	(	(	(	)	)	)	(	(	)	)
Index j	1	2	3	4	5	6	7	8	9	10	11	12
bj	1	-1	1	1	1	-1	-1	-1	1	1	-1	-1
$P_j = \Sigma b_j$	1	0	1	2	3	2	1	0	1	2	1	0
Pj	1	0+1=1	1	2	3	2+1=3	1+1=2	1	1	2	2	1
P'j	0	0.5	0.66	1.75	2.8	2.83	1.857	0.87	0.89	1.9	1.91	0.91
qj	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
qj	-1	0	0.5	0.91	1.91	2.8	1.75	0.66	0.87	1.857	1.9	0.89
tj	0	1	2	12	11	5	4	3	8	7	10	9
rj	2	3	8	7	6	0	10	9	12	11	5	4
mj	2	1	8	7	6	5	4	3	12	11	10	9

10. if opening bracket $m_j = r_j$.
if closing bracket $m_j = t_j$

Process:

1. assign inden as increasing from 1 to n. (no. of processer)
2. for opening bracket assign 1 and closing bracket assign -1 bj value.
3. Sum bj upto that point Pj value eg: $P_3[P_3] = b_1 + b_2 + b_3 = 1 - 1 + 1 = 1$.
4. Again Pj = ~~Pj of prev if ≥0~~
bj of prev if bj = +1.
Pj+1 if bj = -1.
5. $P'_j = P_j - \frac{1}{j}$ for $P'_1 = P_1 - \frac{1}{1} = 1 - 1 = 0$ for $P'_{10} = P_{10} - \frac{1}{10} = 2 - \frac{1}{10} = 1.9$
6. assign $q_j = -1$ for all
7. q_j = look for p' of that cell
find all the p' lower than that p'.
choose max of the set.
eg. for $q_7 = p'_7 = 1.857 = \{0.87, 0.89, 0.91, 1.75, 0.66, 0.5\}$ max = 1.75.
8. $t_j = I$ max inden of P'_j where $q_j = p'_i$
eg. t_5 = location of 1.91 = P'_{11} ∴ $t_5 = 11$.
9. for r_5 = in t_j look for 5 and find its inden ∴ $r_5 = 6$.

Q. max. sum subsegment. Numerical

Maximal Sum Sub Segment:

- given an array of numbers $d_1, d_2, \ldots d_m$, it is required to find a contiguous sub array of maximal sum.

eg. $A = \{-2, -3, 4, -1, -2, 1, 5, -3\}$.

Algorithm:

1. $S_j = \Sigma d_i \mid i \leq j$
2. $M_j = n\, S_i \mid i \geq j$
3. $e_j = n\, i \mid S_i = m_j$
4. $m_j = m_j - S_j + d_j$
5. $t = n\, m_i$
6. $x = n\, i \mid m_i = t$ → Start point
7. $y = e_n$ → end Point.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
Sequence(d_j)	-2	-3	4	-1	-2	1	5	-3
inden (j)	1	2	3	4	5	6	7	8
S_j	-2	-5	-1	-2	-4	-3	2	-1
M_j	2	2	2	2	2	2	2	-1
e_j	7	7	7	7	7	7	7	8
m_j	2	4	7	3	4	6	5	-3

t = max m_i = 7 inden = 3

n = max i such that $m_i = t = 3$ start ind

$y = e_n = e_3 = 7$ → end inden.

1. assign each element to processor
2. assign inden.
3. $S_j = \Sigma d_i$ → sum each element upto that inden.
   for $S_5 = -2-3+4-1-2 = -4$
4. m_j = from that inden to last inden
   { S_j list } max
   eg: for m_7 = 7 to 8 max = $P_7 = 2$
   for m_4 = 4 to 8 max = $P_7 = 2$
5. e_j = inden (max) for that value of m_j on S_i.
   eg. m_j on $P_4 = 2$, that value on $S_j = P_7$.
   ∴ 7.
6. $M_j = M_j - S_j + d_j$ = $M_5(P_5) = M_5 - S_5 + d_5$
   $= 2 - (-4) + (-2)$
   $= 4$
7.

So Subset = inden 3 to inden 7.

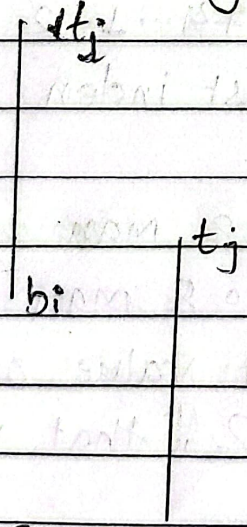
$= \{4, -1, -2, 1, 5\}$

Q. types of BSR:

Date
Page

* Two Criterion BSR:
  - tested for two criteria before being allowed to take part in reduction.
  - eg.
    generate even numbers from 1 to 100
    1st criteria = $1 \leq n \leq 100$
    2nd criteria = $n \% 2 == 0$

* Three Criterion BSR:
  - have to pass three criteria to go to reduction process.
  - eg:
    vertical segment visibility:



$(x_i < x_j) \wedge (t_i \geq t_j) \wedge (b_i \leq t_j)$.

or

Unit 2.3 Data flow Models:

- flow of Information between data processing entities.
- following are variants of DFM:
  * Control flow:
  * Data flow:

Control flow assumes that a Program in a Series of instruction, each of which Specifies:
1. either an operation with memory location.
2. or Specifies transfer of control to another instruction

eg:

```
if (n==0)
then
    c = a+b
else
    c = a-b
```

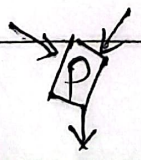
But in data flow model it executes any operation as soon as necessary operands are available.

Q. Primitives of Dataflow.

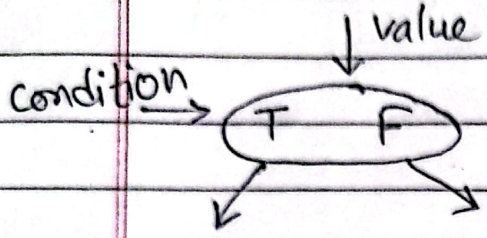
Basic Primitives of Data flow:

1. **Operator** : - a data value is Produced by an operator by performing some operation f.

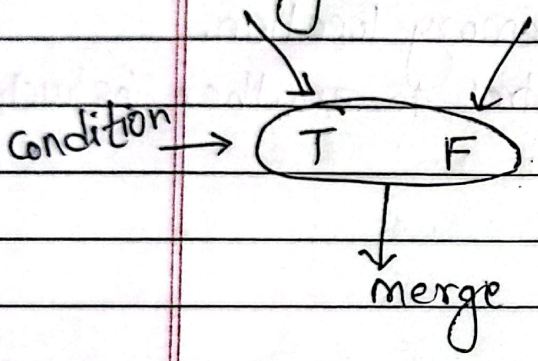
2. **Predicate** : - a TRUE or FALSE control value is generated by a decider depending on its input tokens.



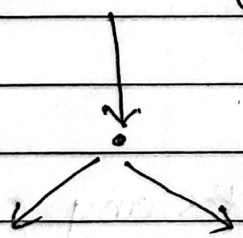
2. Switch :- a switch actor directs an input data token to one of its output depending upon the Condition applie



4. Merge :- a merge actor passes one of its input tokens to the output dependi on the input control token.

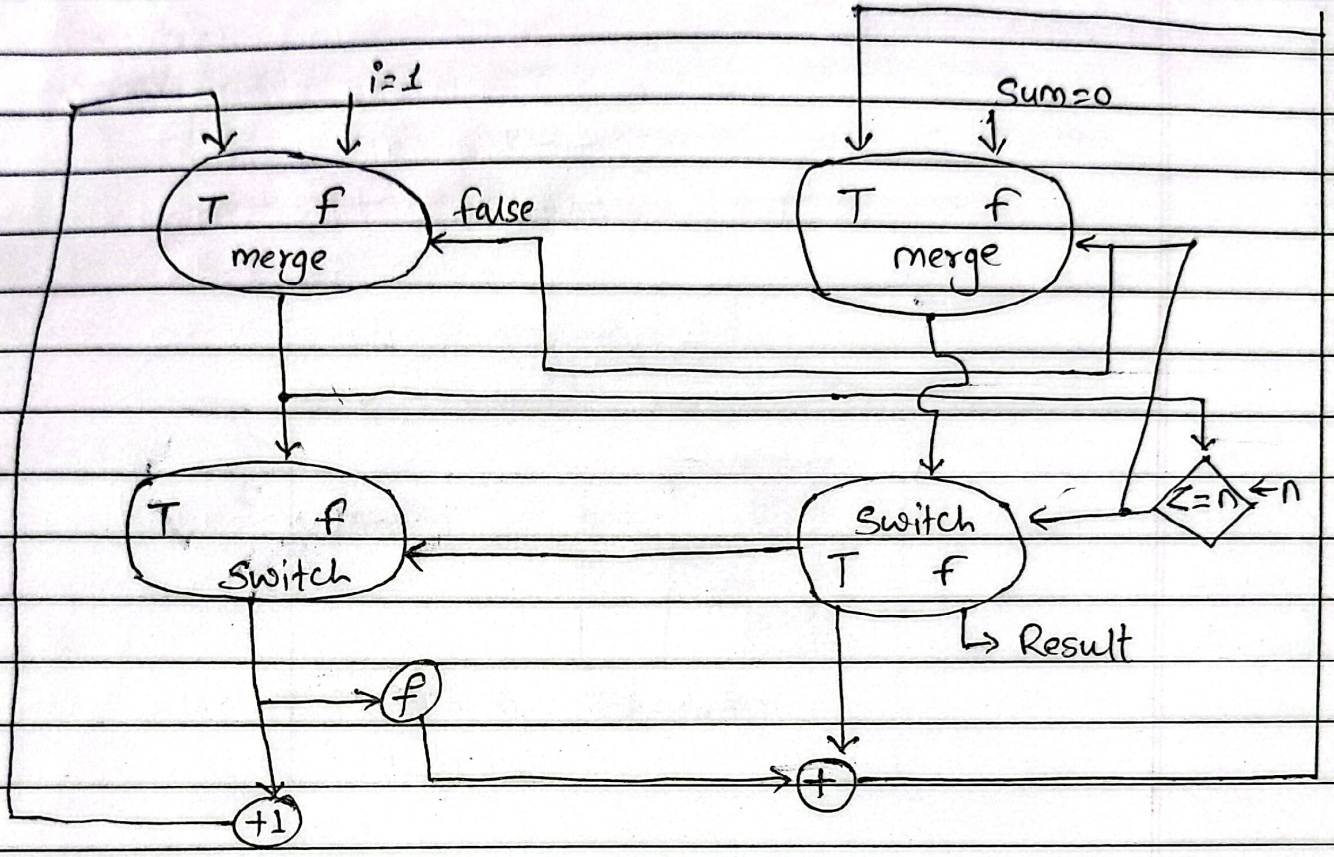


5. Copy : → a copy is an identity operator which duplicates input tokens.



example:

$$Sum = \sum_{i=1}^{n} f(i)$$



Demand Driven Data flow Computing Model:

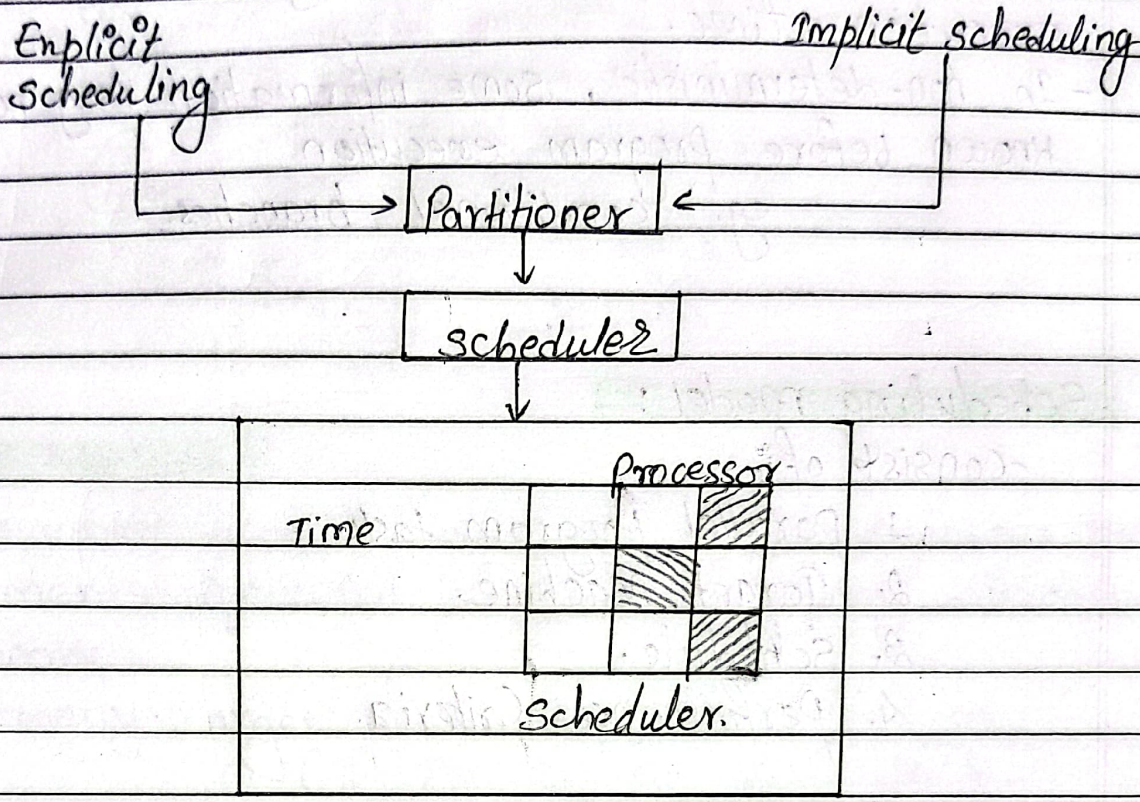
an operation of a node will be performed only if there are tokens on all the input edges and there is a demand for the result of applying the operation.

Assignment:

- Static Dataflow Model.
- Dynamic Data flow Model.

2.4 Partitioning and Scheduling:

- Program partitioning and task scheduling are two distinguishing features of parallel vs sequential programming.
-



Data Partitioning: - Same computation on different set of data concurrently.
eg. Matrix multiplication

Function Partitioning: - different computation on same data concurrently.
eg. flight simulation.

Task Scheduling: - after program partitioning, tasks must be optimally scheduled on the processors such that the program execution time is minimized.
- Scheduling techniques can be classified as deterministic

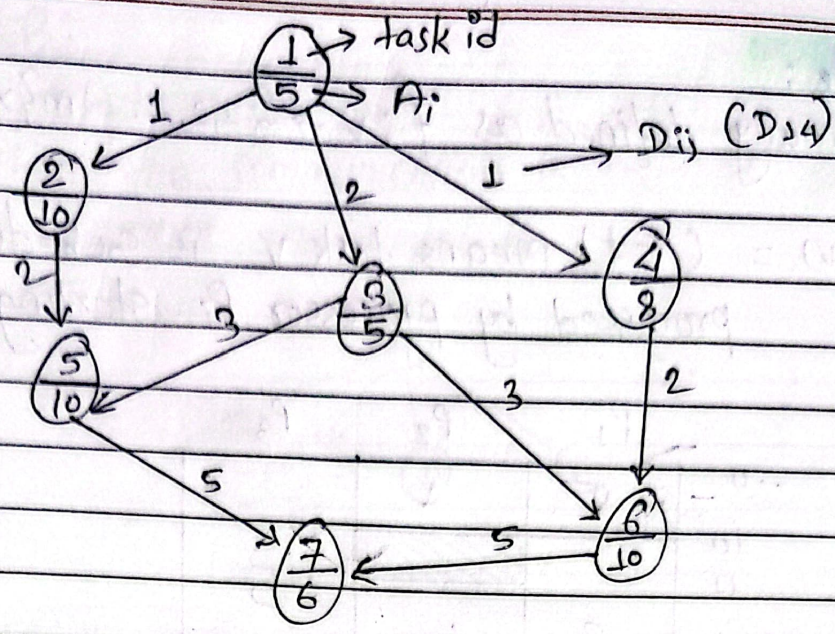
and non-deterministic.
- In deterministic scheduling, all the information about task to be scheduled is entirely known prior to execution time.
- In non-deterministic, some information may not be known before program execution
  eg. conditional branches.

Scheduling model:

- consist of
  1. parallel program task
  2. Target machine
  3. Schedule,
  4. performance Criteria.

1. Parallel Program Task:

- Can be defined as the System $(T, \prec, D_{ij}, A_i)$
  Where,
  $T = \{t_1, t_2, \ldots t_n\}$ is a set of task to be executed.
  $\prec$ - Partial order defined on T.
  - $t_i \prec t_j$ means t_i must be ~~executed~~ completed before t_j can start execution.
  D_{ij} - an nxn matrix of communication data, where $D_{ij} \geq 0$ is the amount of data required to be transmitted from t_i to t_j
  A_i - n vector of the amount of computation i.e. the number of instruction required to execute t_i



2. Target Machine:

- assumed to be made up of "M" heterogeneous processing elements connected using an arbitrary interconnection network.
- formally target machine can be defined as:

$$(P, P_{ij}, S_i, T_i, B_i, R_{ij})$$

where,

P_{ij} → mxm interconnection matrix.

$P \rightarrow \{P_1, P_2, \ldots P_m\}$ is set of m processor.

S_i → Specifies the speed of processor P_i

T_i → Specifies the startup cost of initiating message m on processor p.

B_i → Specifies startup cost of initiating process on P_i

R_{ij} → transmission rate over link connecting P_i & P_j

execution time of task i ~~can~~ on processor j can be computed as:

$$T_{ij} = \frac{A_i}{S_j} + B_j$$

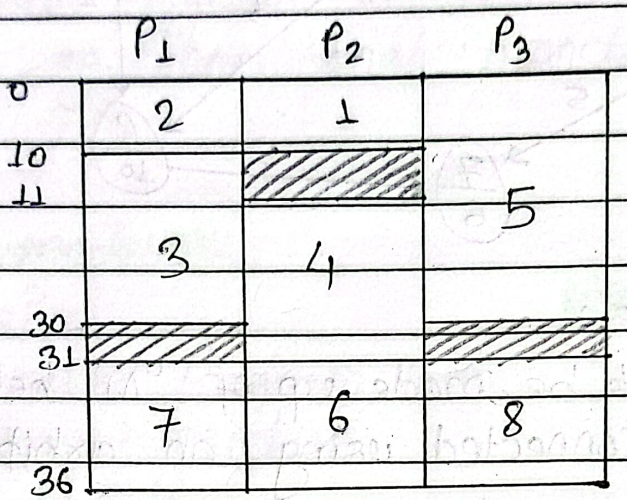
task Id ↙ ↓ Processor Id.

3. Schedule :

- formally defined as $f : V \rightarrow \{1, 2, \ldots m\} \times \{0, 1, \ldots \infty\}$

i.e.

$f(v) = (i, t)$ means task v is scheduled to be processed by processor P_i starting at time t.



Gantt Chart for schedule

4. Performance Criteria :

- scheduling length or maximum finishing time.

i.e.

length $(f) = \text{maximum} \{ t + T_{ij} \}$

Communication Model :

- execution time
- communication delay

Model A :

- total cost = communication cost + execution cost

where,

execution cost = schedule length.

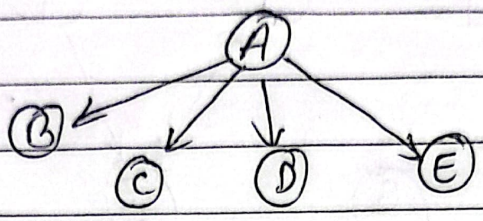
communication cost = no. of msg * cost per msg.

No. of msg = Pairs (u, v) and processor $(u) \neq$ processor (v)

Model B:

- if tasks u, v and $w \in V$ such that $(u,v), (u,w) \in A$ then the communication is counted twice in model A but once in Model B.

eg:



Optimal Scheduling Algorithm:

- deal in three cases:
  * When the task graph is a tree
  * When the task graph is an interval order.
  * When there are only 2 processor available.

Scheduling tree structure task graph

* Scheduling in-forest/out-forest without communication
* Scheduling in-forest/out-forest with communication.

Without Communication:

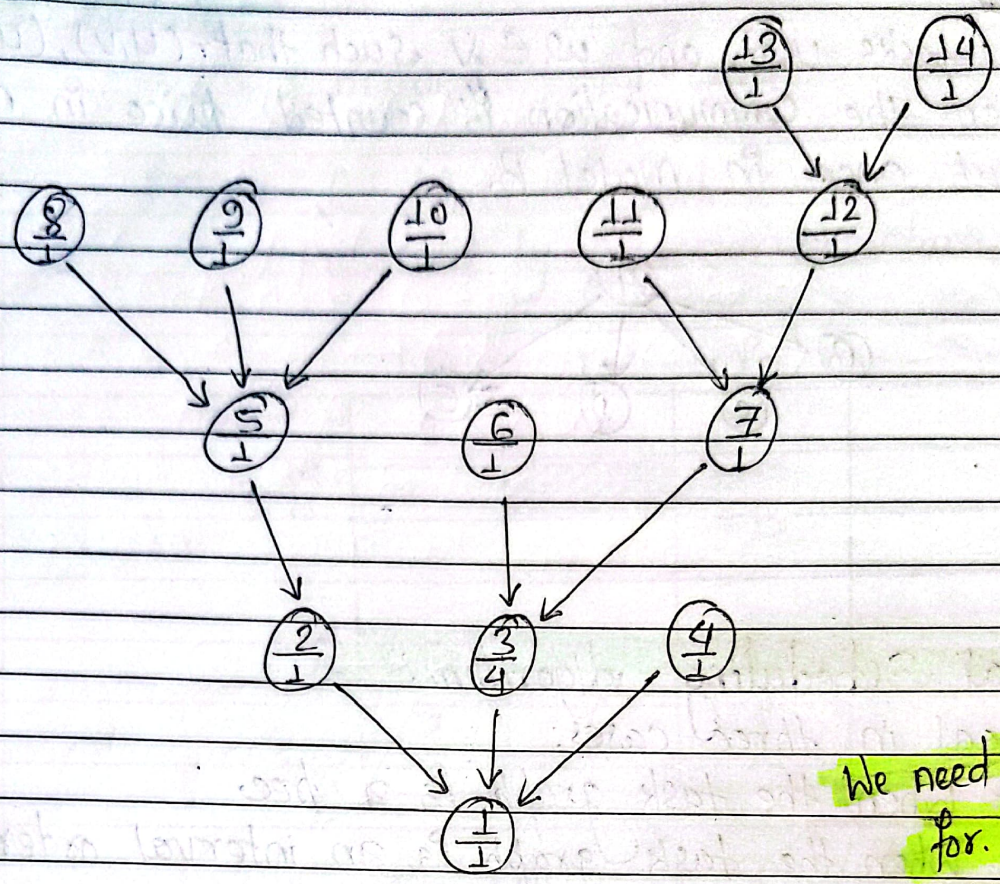
- task graph is either

1. In forest → each node has at most one immediate successor
2. Out forest → each node has at most one immediate predecessor.
   - generally highest level ~~elements~~ node are used first.

Algorithm:

1. The level of each node in the task graph is calculated and used as each node ~~Priop~~ priority.
2. Whenever a processor becomes available, assign the unexecuted ready task with the highest priority.

eg:



Task graph.

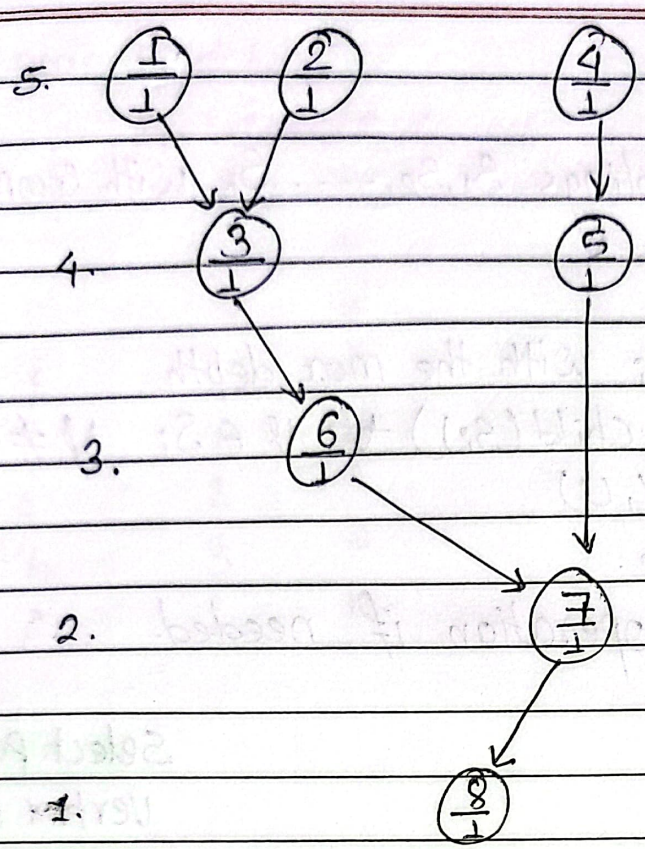
We need to look for task dependency as well

Node	Priority.
14	5
13	5
12	4
11	4
10	4
9	4
8	4
7	3
6	3
5	3
4	2
3	2
2	2
1	1

Task Priority.

	P_1	P_2	P_3
0	14	13	11.
1	12	10	9
2	8	7	6
3	5	4	3
4	2		
5	1.		

Task Schedule.



Task graph

Node	Priority.
1.	5
2.	5
3.	4
4.	5
5.	4
6.	3
7.	2
8.	1.

Task Priority.

	P_1	P_2	P_3
0.	1	2	4
1.	3	5	
2.	6		
3.	7.		
4.	8		

Task schedule

With Communication:

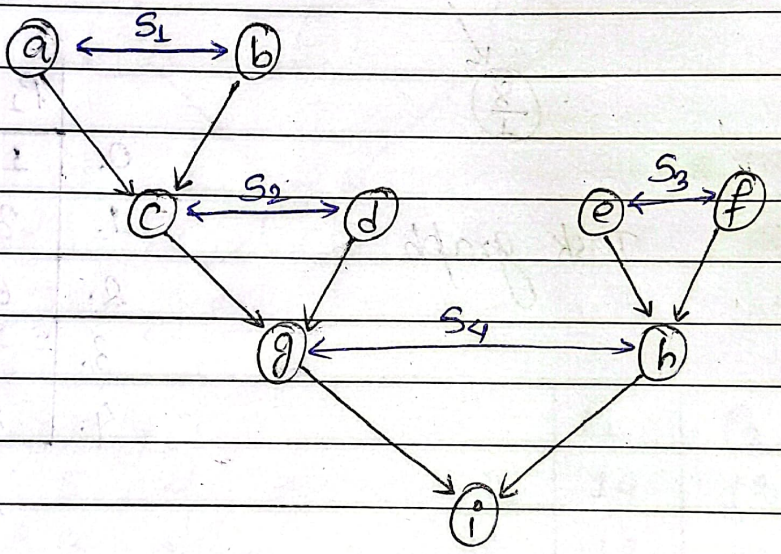
- augmented task graph is created from original task graph.
- scheduling the augmented task graph without communicatio is equivalent to scheduling original task graph with communication.

Node depth: depth of a node is the length of the longest path from any node with depth zero to that node.

Operation Swap all: - Swap all the task pair scheduled on processor i and j is the scheduled at $t_i \geq t$.

Algorithm:

1. Identify the set of siblings $S_1, S_2, \ldots S_K$ with common child.
2. $A_1 \leftarrow A$
3. for every set S_i
   - Pick node $U \in S_i$ with the max depth
   - $A_1 \leftarrow A_1 \leftarrow (V, child(S_i)) \; \forall \; V \in S_i \; V \neq U$
   - $A_1 \leftarrow A_1 \cup (V, U)$
4. obtain the schedule.
5. perform "swap all" operation if needed.



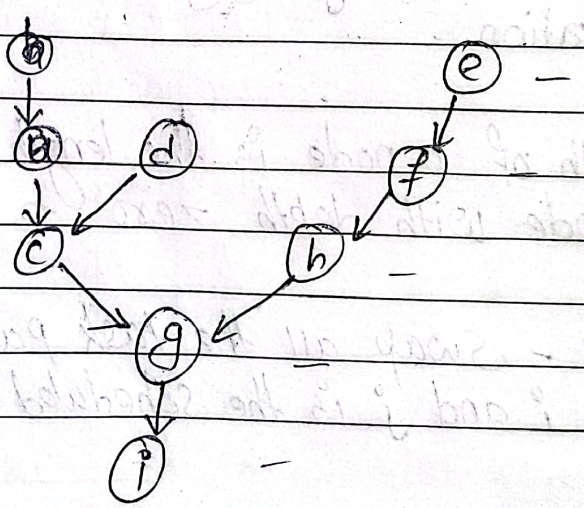
Select pair of vertex with common child.

$S_1 = (a, b) \rightarrow$ both have no. predecessor, choose any $\rightarrow$ (b)

$S_2 = (c, d) \rightarrow$ c has 1 preedecessor & d has 0 $\rightarrow$ (c)

$S_3 = (e, f) \rightarrow$ both have 0, choose any $\rightarrow$ (e)

$S_4 = (g, h) \rightarrow$ g has 2, h has 1 $\rightarrow$ (g)



Augmented Task Graph.

node	depth.	
a	4	↓
b	5	↓
c	3	↓
d	4	↓
e	5	↓
f	4	↓
g	2	↓
h	3	↓
i	1.	↓

	P_1	P_2
0	b	e
1.	a	d
2.	f	c
3.	h	
4.	g	
5.	i	

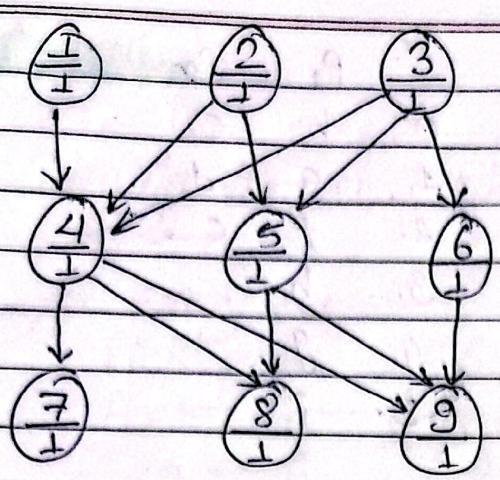
Scheduling Interval Ordered Task :

- used to indicate the task that describe the precedence to location among tasks in an interval order:
    1. Without Communication.
    2. With communication.

Without Communication :

- the number of all successors of each node is used as each nodes Priority.
- whenever a processor becomes available assign it with the exexecuted ready task with the highest priority.

eg.



Task graph

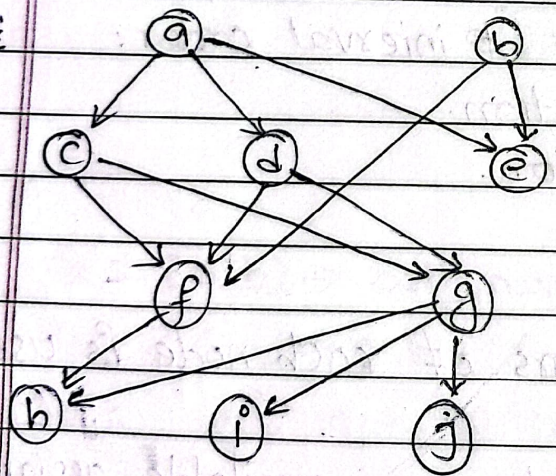
node	No. of successor
1	4
2	5
3	6
4	3
5	2
6	1
7	0
8	0
9	0

Task Priority.

P_1	P_2	P_3
3	2	1
4	5	6
7	8	9

Time Scheduling.

eg:

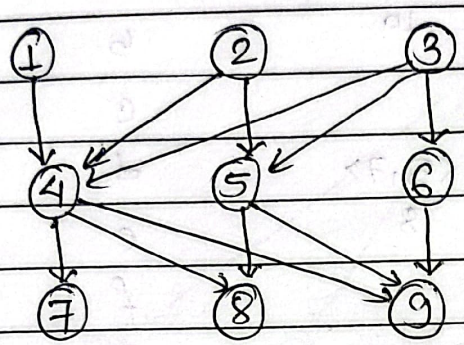


node	No. of successor.
a	8 ✓
b	3 ✓
c	5 ✓
d	5 ✓
e	0 ✓
f	1
g	3 ✓
h	0
i	0
j	0

P_1	P_2	P_3
a	b	
c	d	e
g	f	
h	i	j

With Communication:

- Communication delay is considered as one unit time whenever they are scheduled on different processor.

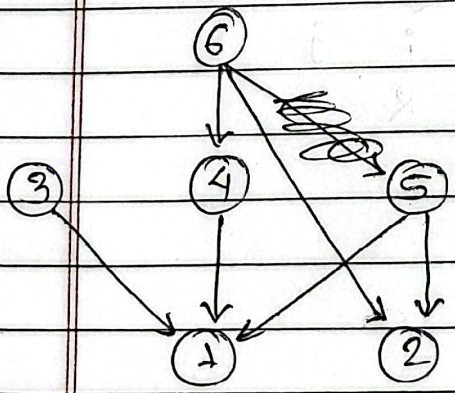


Task graph

node	no. of successor.
1.	4
2.	5
3.	6
4.	3
5.	2
6.	1
7.	0
8.	0
9.	0

Task Priority.

P_1	P_2	P_3
3	2	1
6		
4	5	
7		
8	9	

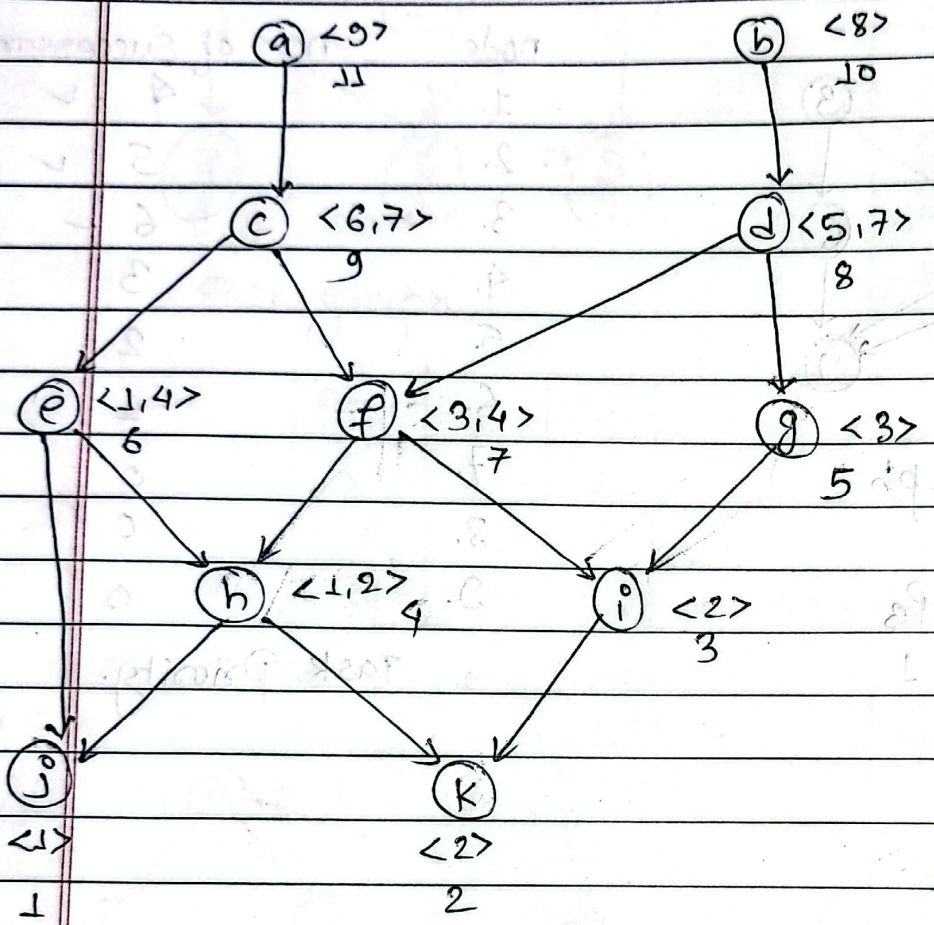


node	no. of successor.
6	3
5	2
4	1
3	1
2	0
1	0

P_1	P_2	P_3
6	5	
4	3	
2		
1.		

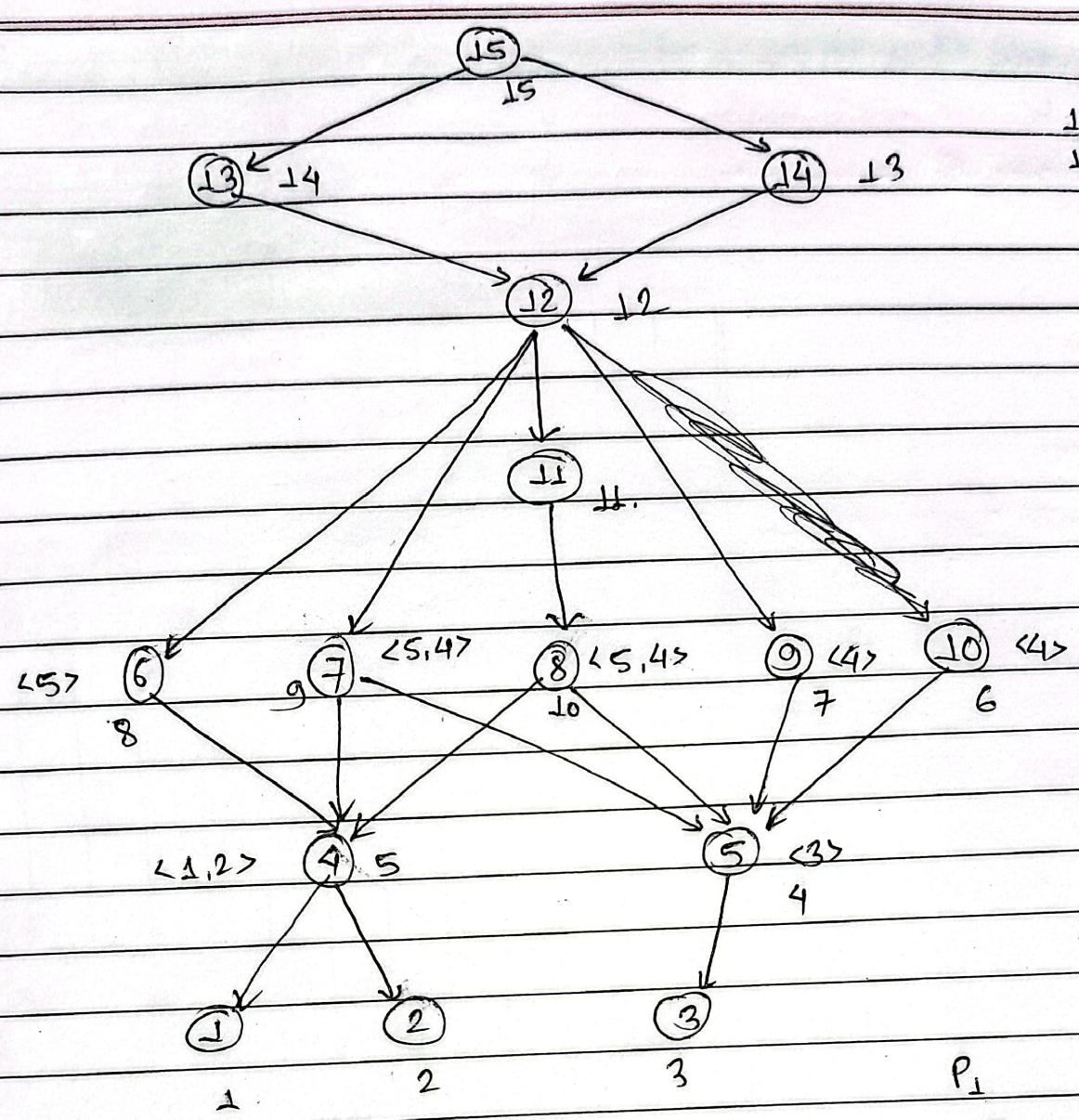
Two Processor Scheduling:

- In this case, the number of processor is limited to only two.



node	label
a	11
b	10
c	9
d	8
e	6
f	7
g	5
h	4
i	3
j	1
K	2

P_1	P_2
a	b
c	d
f	e
g	h
i	j
K	



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

P_1	P_2
15	
13	14
12	10
11	7
8	6
9	
4	5
3	2
1.	

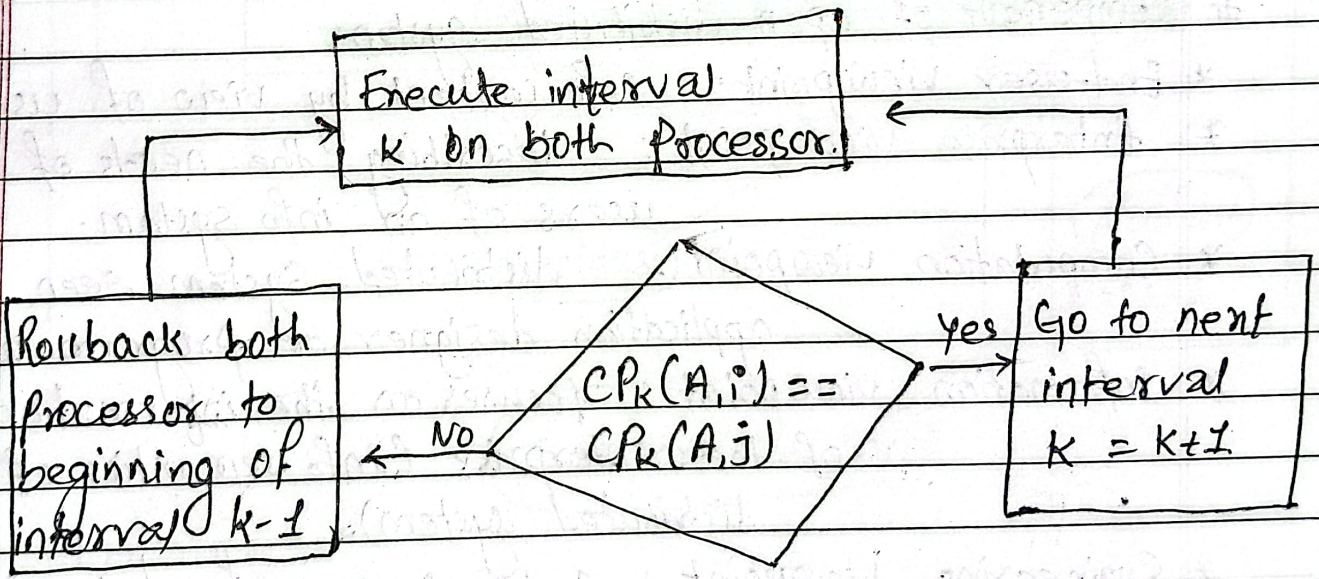
Unit 2.5

Checkpointing in Parallel and Distributed System:

- Fault Detection.
- Fault location.
- Fault Containment.
- Fault Recovery.

Checkpointing using task duplication:

- Compares the state of same task that are executed in parallel on different processor.
- matching state is an indication of a correct execut

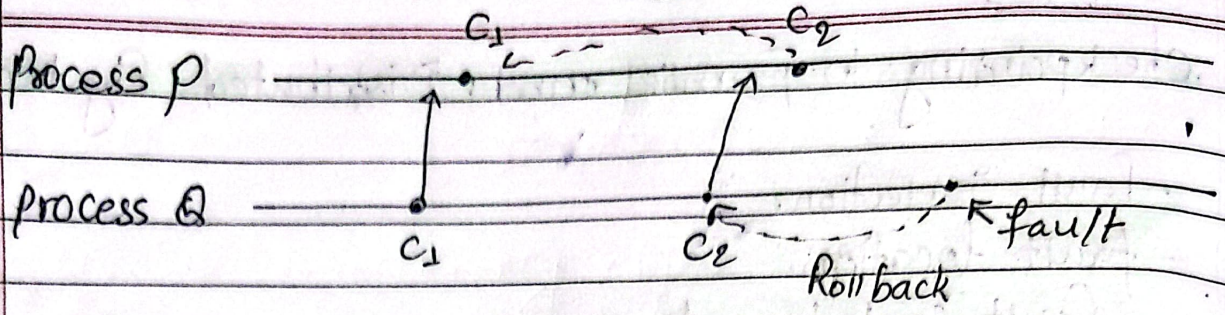


Technique for consisting consistent checkpointing.

- When a fault is detected in one of the processor the process is rolled back to its last saved checkpoint.
- In order to maintain consistent checkpoint, some other task need to be rolled back, called domino effect.

Domino effect

(domino effect) also Rollback



Page: 125

Unit 2.6 Open Distributed System:

- Complex structure composed of many types of hardware and software component.
- eg. telecommunication system.

Component of open-distributed system.

* End-user viewpoint: Info content by view of user.
* Enterprise viewpoint: describing the needs of the users of an info system.
* Computation viewpoint: distributed System seen by application designer or programmer.
* Information viewpoint: focuses on the information content of the enterprise (Info semantics of distributed system).
* Engineering Viewpoint: System support for distributed system
* Technology Viewpoint: Technical component info.

Engineering Model:

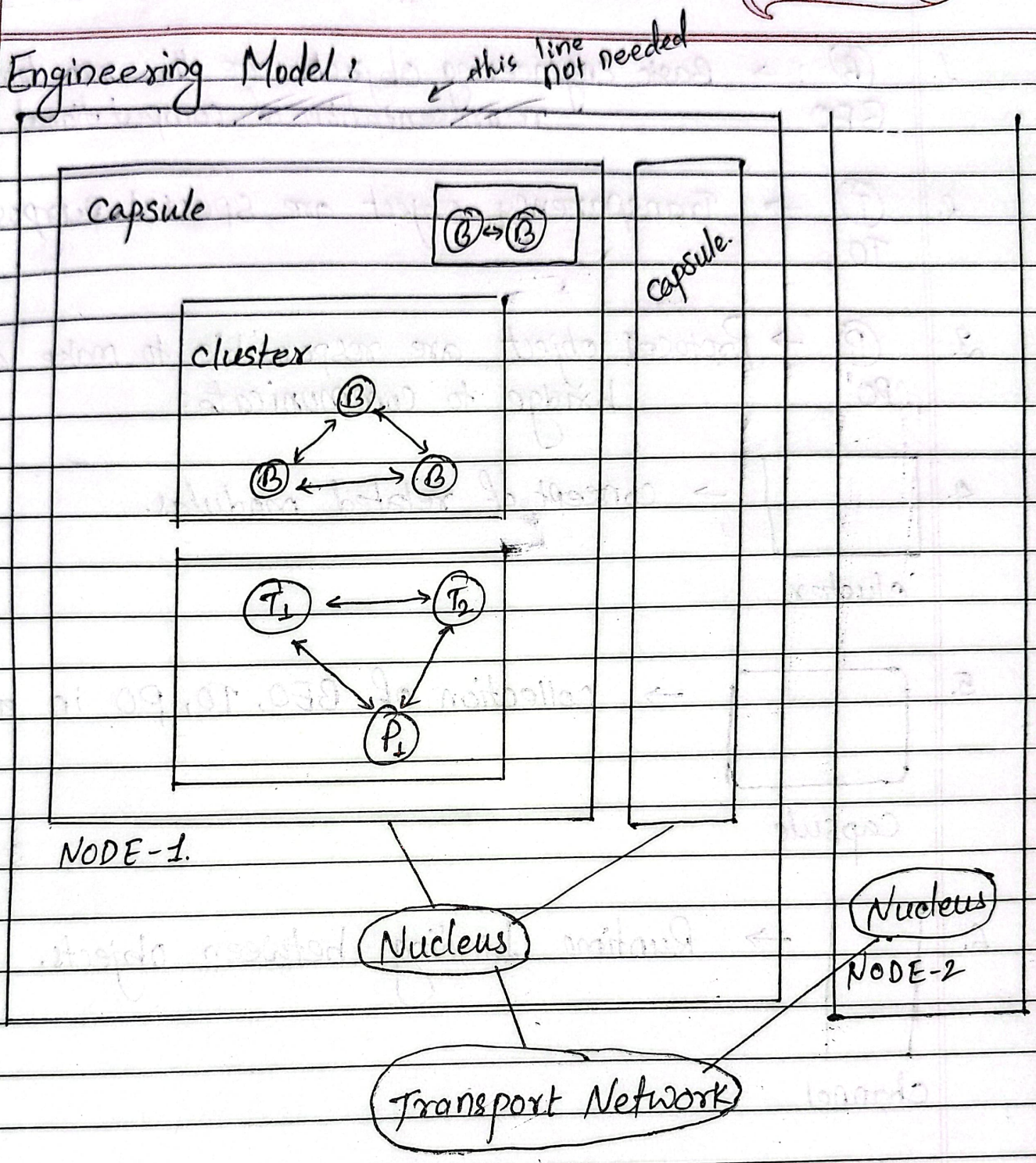
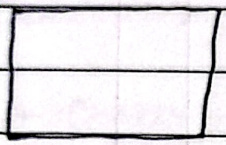


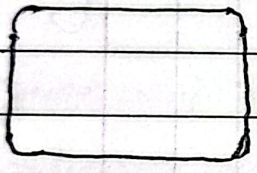
fig: Engineering Model.

1. (B) : → Basic Engineering Objects are the run time representation of computational object.
BEO

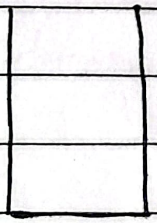
2. (T) → Transparency object are special purpose Model.
TO

3. (P) → Protocol object are responsible to make a bridge to communicate.
PO

4.  → concept of related modules.
cluster

5.  → collection of BEO, TO, PO in a window
Capsule

6.  → Runtime binding between objects.
channel

7.  → Single Computer System.
node

Date :
Page:

Unit 3.1. Fundamental of Parallel Algorithm:

Models of Parallel Computation:

1. PRAM Model

$$T_p(n) = O\left(\frac{T_s(n)}{p}\right)$$

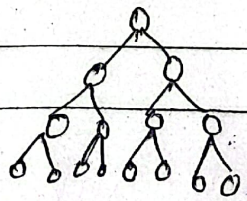
2. Shared Memory Model:
   - executes asynchronously.

3. Network Model:
   - topology of interconnection network

4. LogP model:
   - L → upper bound of the latency in communicating the msg.
   - O → Overhead, length of a time that a processor is engaged in submission of message
   - g → gap, maximum time interval between consecutive node message transmission.
   - P → no. of Processor.

Balanced Tree:
- strategy for designing parallel algorithm.

eg. Computing the sum of element in the array.



Date : .....
Page: .....

Divide and Conquer:

- Partitioning the inputs into several partition.
- Solve each partition in parallel.
- merge the solution.

eg. Quick sort.

Date :
Page :

Unit 3.2 Tree Graph.

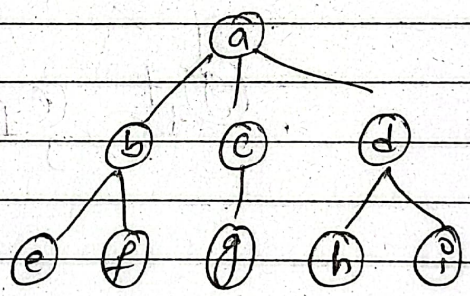
Computing a post order numbering:

→ left, right, root.

→ given a rooted tree, T along with its Euler Tour, the task of computing a post order numbering of the node T can be performed as follows;

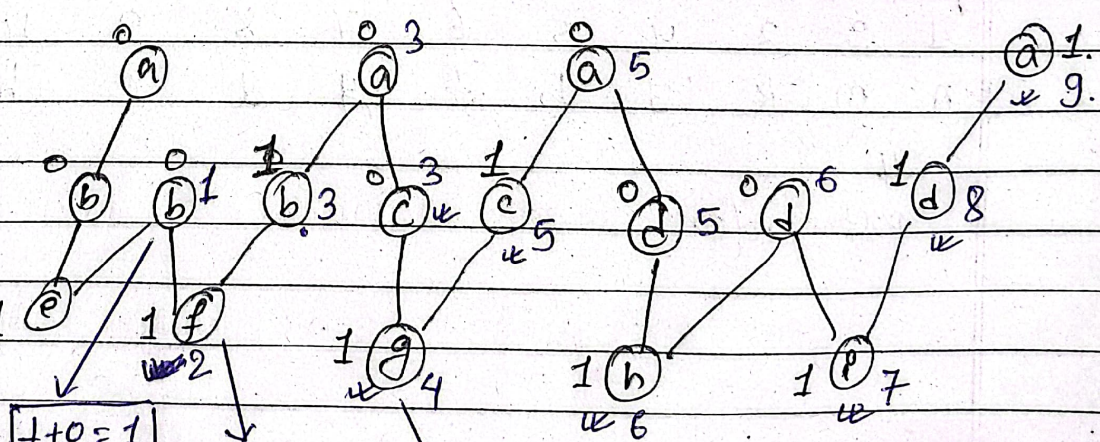
1. for every node V of T, assign a weight of 0 to $v^1, v^2, \ldots v^{d(v)}$ and weight of 1 to $v^{d(v)+1}$.
2. compute the weighted rank of every node in the Euler Tour.
3. for every node $v^{d(v)+1}$, the corresponding weighted rank is the post order of V.

eg.



1. assign 1 to leaf node and 0 to all other node if it is not last and if it is last assign 1.

Euler Tour.



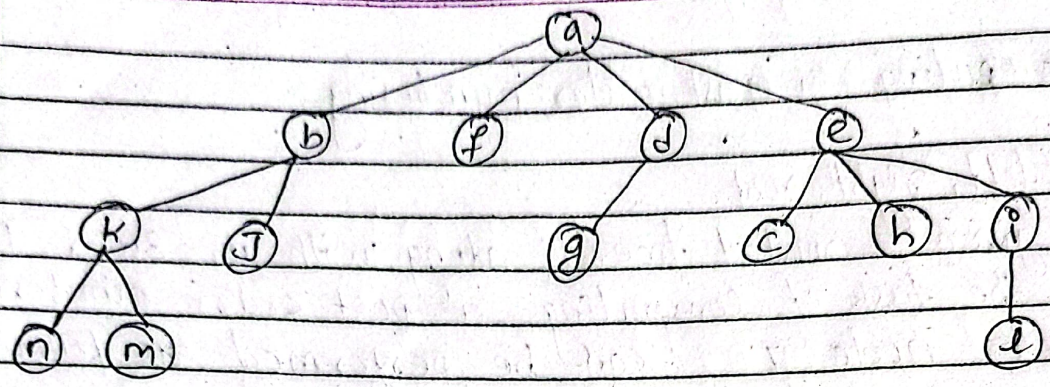
2. Take first node for every value
eg. 1 = e.
3 = b.

2. Start adding and updating weight.

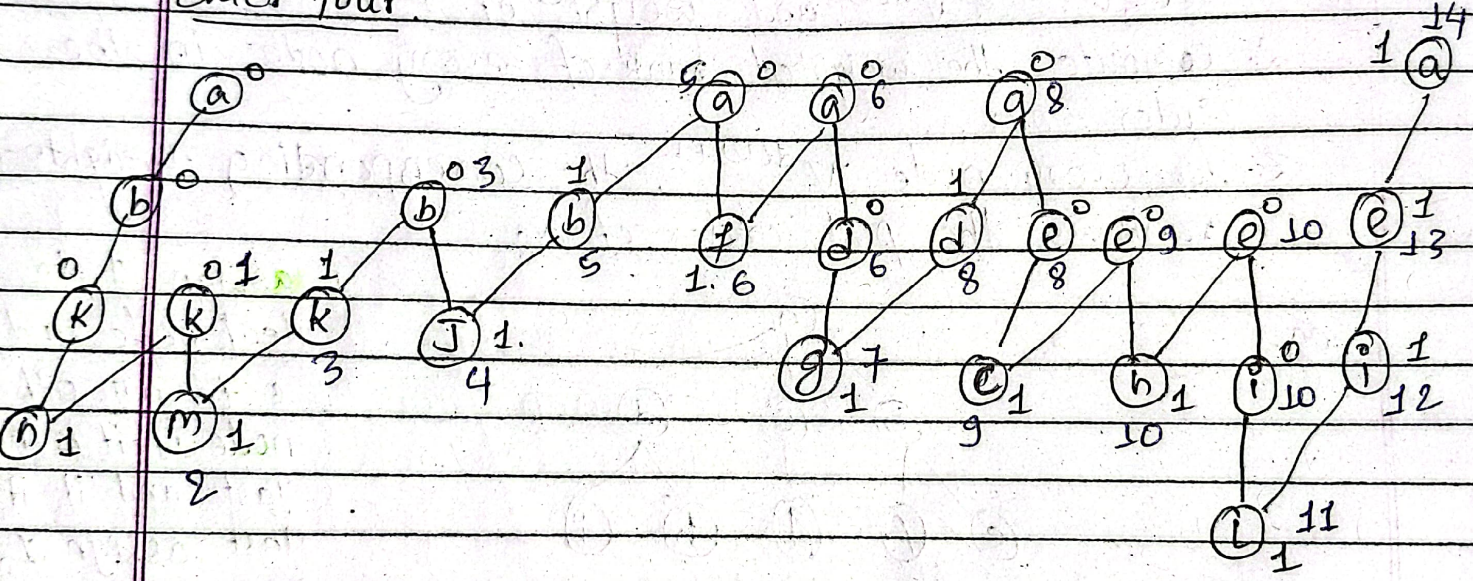
1 2 3 4 5 6 7 8 9
e f b g c h i d a

Date : .....

Page : .....



Euler Tour :



1	2	3	4	5	6	7	8	9	10	11	12	13	14
n	m	K	J	b	f	g	d	c	h	l	i	e	a

↓

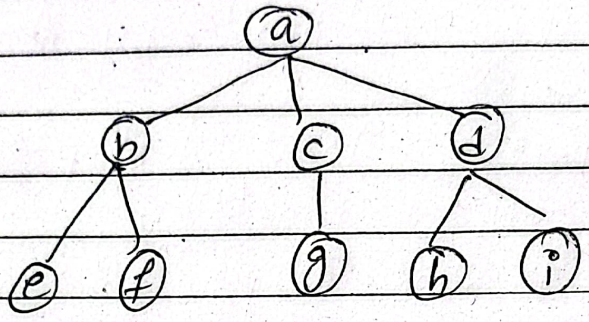
post order.

Date :
Page :

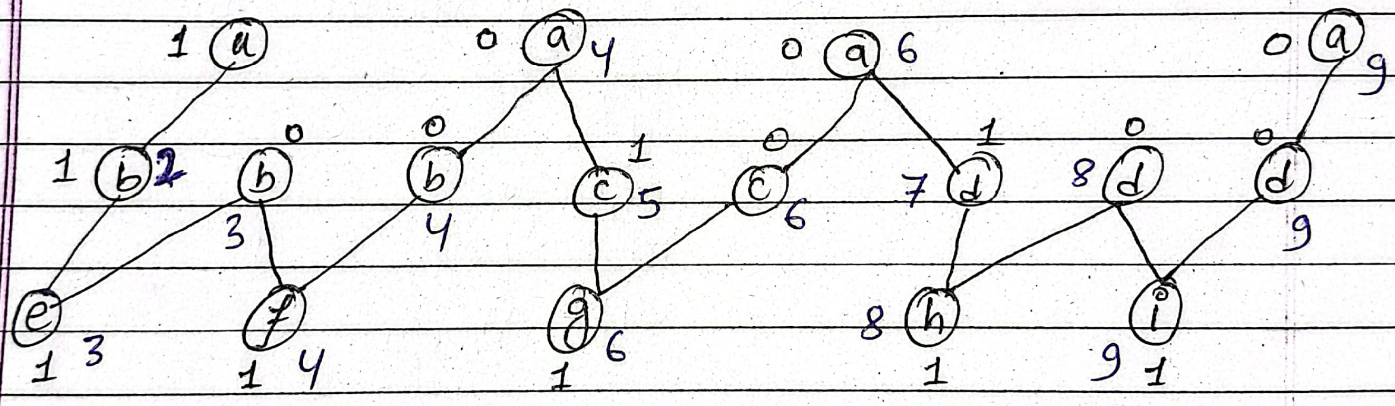
Computing the number of descendents:

- given the routed tree T and ~~et~~ euler TOUR, assign a weighted of 1 to v^1 and 0 to $v^2, v^3 \ldots v^{d(v)+1}$
- no. of descendent $= x(v^{d(v)+1}) - x(v^1) + 1$.

eg:



Euler TOUR:



1. Assign 1 to all leaf node
   1 to node if it is first then 0
2. Sum value of each node with prev and update.

no. of descendent of (a) = heighest of a - lowest of a + 1
$= 9 - 1 + 1 = 9$

no. of descendent of b $= 4 - 2 + 1. = 3$

no. of descendent of d $= 9 - 7 + 1 = 3.$

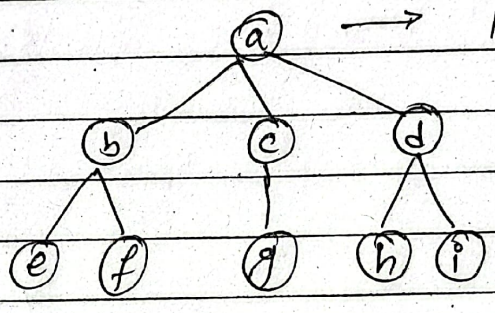
e, f, g, h, i → leaf node ∴ 1.

Date : .....
Page : .....

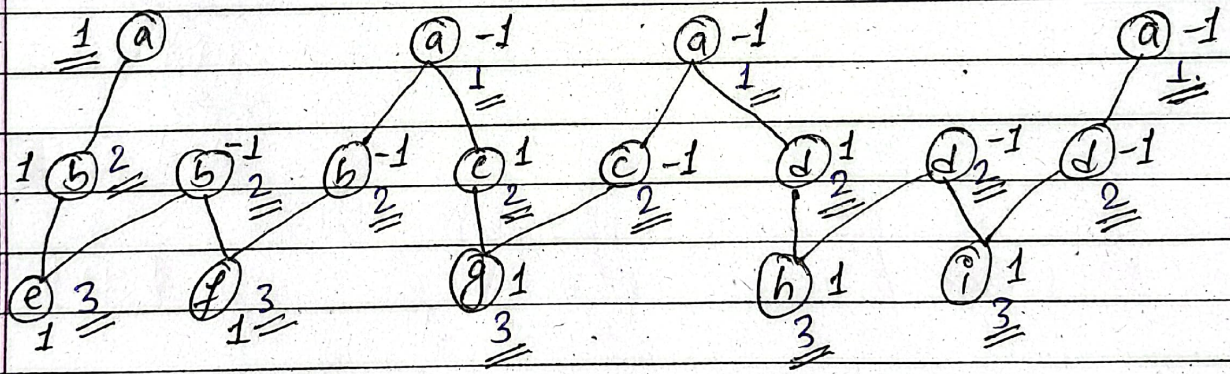
Level Computation:

→ Consider a tree T with Euler Tour.
→ Assign a weight 1 to v^1 and -1 to every v^2 ∴ I ≠ i

Example:



Euler Tour.



Step 1: Euler Tour.
Step 2: Assign 1 to first occur
-1 to other.
Step 3: Compute as like other.

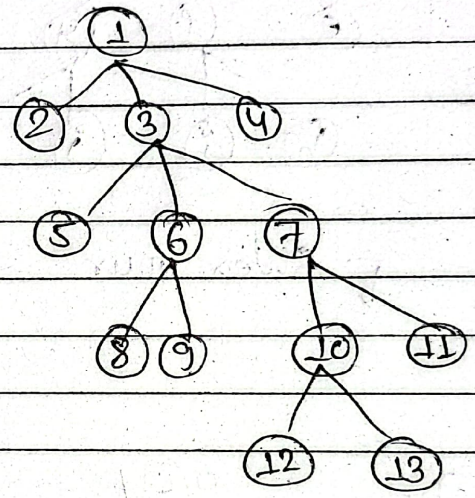
Date: .....
Page: .....

Unit 3.3

Lowest Common Ancestor:

Let T be a rooted tree with n nodes.

- lowest common ancestor between two nodes v & w is defined as the lowest node in T that has both v & w as descendents.



LCA (9, 12) = 3

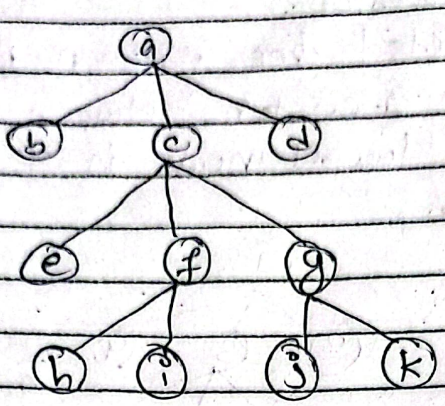
Algorithm:

- Compute pre-ordering numbering of the nodes in T
- for all node u in T do in parallel.
  - compute $l(u), r(u)$
- for all $v \in [l(u), r(u)]$ do in parallel.
  - $A[u, v] = u$
- for all $x \in [l(w_i), r(w_i)]$ and $y \in [l(w_j), r(w_j)]$ $1 \le i \ne j \le d(u)$ do in parallel.
  - $A[x, y] = u$

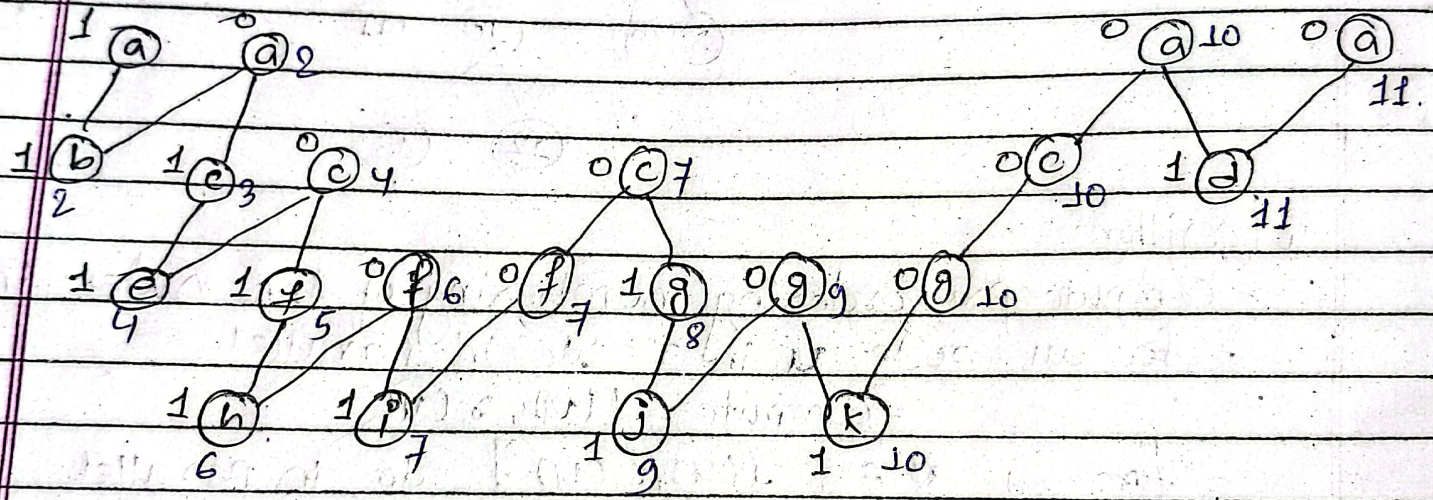
Return A.

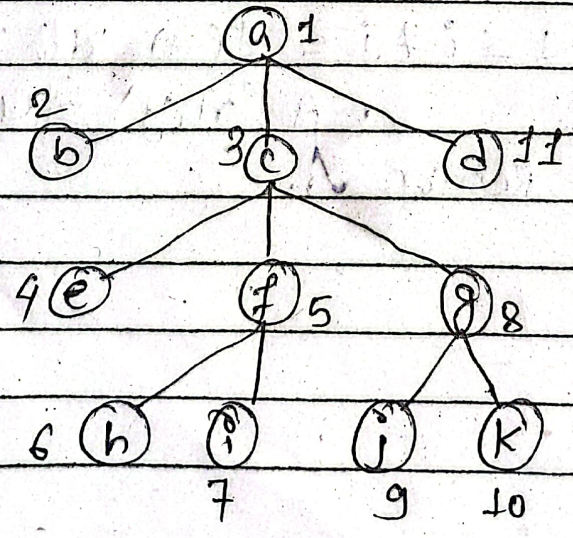
Date : .....
Page : .....

example :



↳ Euler tour





$l(a) = 1$	$r(a) = 11$
$l(b) = 2$	$r(b) = 2$
$l(c) = 3$	$r(c) = 10$
$l(d) = 11$	$r(d) = 11$
$l(e) = 4$	$r(e) = 4$
$l(f) = 5$	$r(f) = 7$
$l(g) = 8$	$r(g) = 10$
$l(h) = 6$	$r(h) = 6$
$l(i) = 7$	$r(i) = 7$
$l(j) = 9$	$r(j) = 9$
$l(k) = 10$	$r(k) = 10$

Date : .....

Page: .....

	1	2	3	4	5	6	7	8	9	10	11
1	A	A	A	A	A	A	A	A	A	A	A
2	A	B	A	A	A	A	A	A	A	A	A
3	A	A	C	C	C	C	C	C	C	C	A
4	A	A	C	E	C	C	C	C	C	C	A
5	A	A	C	C	f	f	f	C	C	C	A
6	A	A	C	C	f	H	f	C	C	C	A
7	A	A	C	C	f	f	I	C	C	C	A
8	A	A	C	C	C	C	C	G	G	G	A
9	A	A	C	C	C	C	C	G	J	G	A
10	A	A	C	C	C	C	C	G	G	K	A
11	A	A	A	A	A	A	A	A	A	A	D

for a,

Unit 3.3:

Data Parallel Algorithm:

- a model of parallel computing in which the same set of instruction is applied to all the elements in a dataset.

machine model:

- Based on SIMD machine model
- Single sequence of instruction is executed simultaneously in an arbitrary set of processor with each processor operating on its own data.

CU/PE overlap: (Page no. 133).

- CU initiates parallel computation by sending instruction
- these instruction are executed by PE while CU can be performing its own computation called CU/PE overlap.

Parallel Reduction Operation:

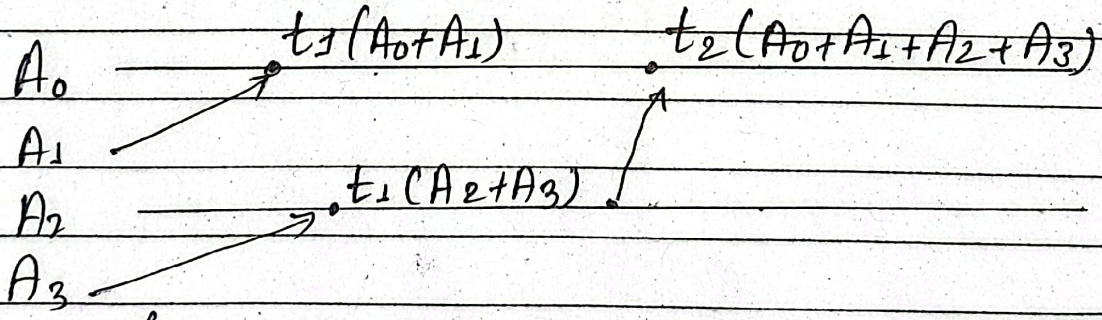


fig: single Reduction.

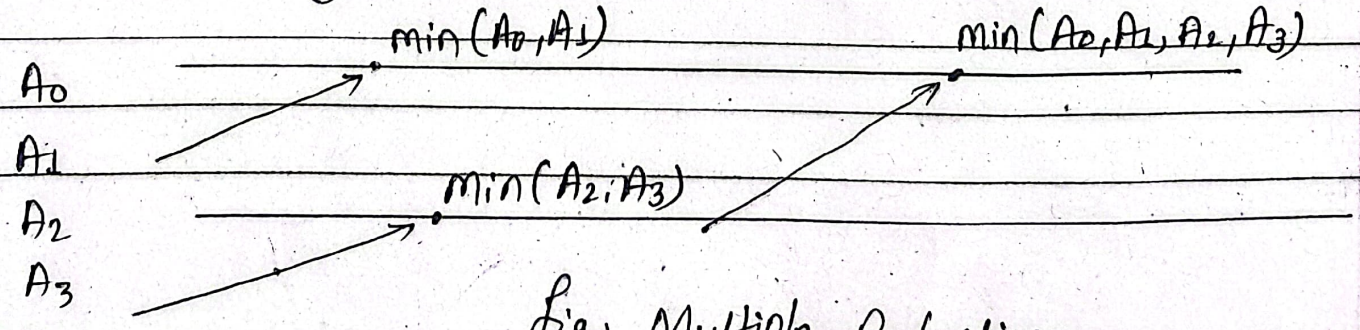


fig: Multiple Reduction.