ASSIGNMENT

COMP 486

AAYUSH POKHAREL

BSc COMPUTER SCIENCE

ROLL NO 43

1> With diagram, explain recovery block for fault toleran-

Ans:- Fault tolerance is a property of a system that enables it to continue operating at the same or degraded efficiency in the event of one or more fault within the system.

Recovery block is a segment of code or a subsystem designed to handle failures and restore the system to a stable state after a fault occurs. They acheive this by encapsulating critical operations within a structured framework.

That structured framework includes:-

(i) **Primary:-** The main computation or function that performs the intended task.

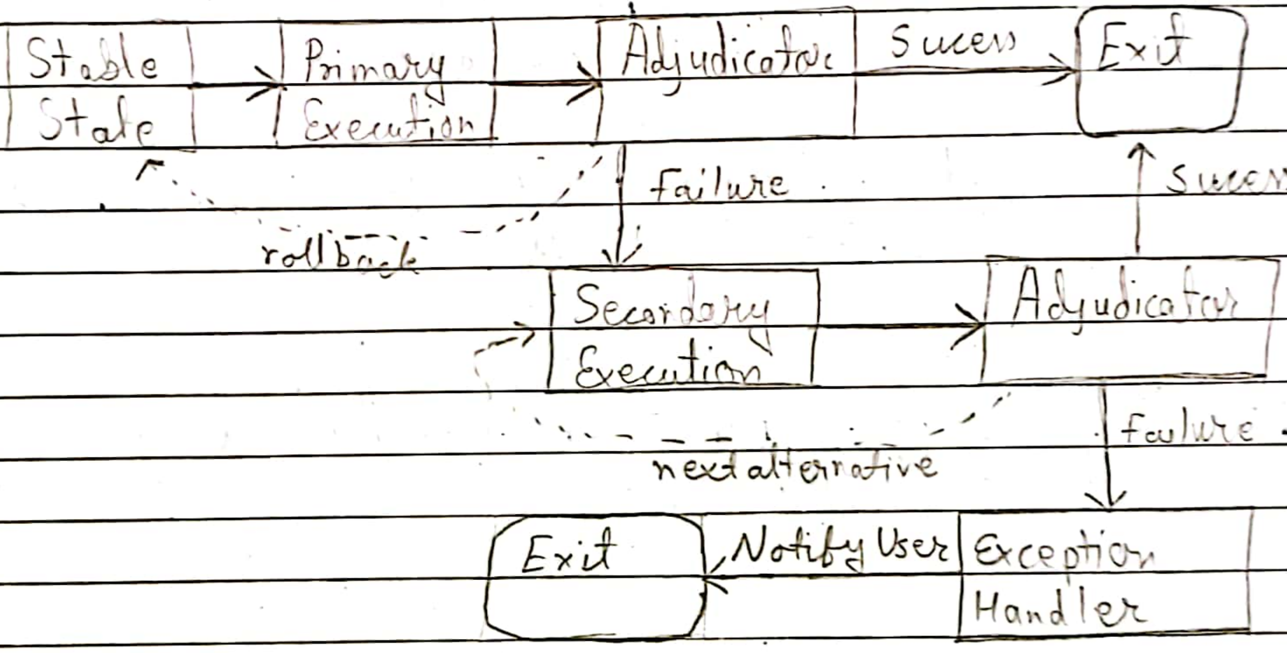
(2) **Secondary(Alternative):** Alternative implementation of the primary, potentially using different algorithms or approaches.

(3) **Adjudicator:-** A decision-making logic that determines whether the primary or secondary's (alternatives) ouptut is valid and reliable.

Rollback:

A mechanism to revert the system (data) to a safepoint before the fault occured (if possible).

The following diagram can be used to explain the workings of recovery block.



(i) The system is a running on a stable state and needs to execute some code. This is inside a recovery block.

(ii) The primary execution inside the recovery block is run and its result will be evaluated by an Adjudicator.

(iii) If the Adjudicator passes it, the execution is deemed sucessed and the block is exited.

<iv> If it fails, then the state is reverted to steady state (if possible)

<v> An alternative execution is started and its result is evaluated by Adjudicator which if passed, exits the block.

<vi> Otherwise on failure, the next alternative is evaluated and the cycle goes on.

vii> When all alternatives are evaluated, the Adjudicator fails the results and the Exception Handler is called.

viii> The Exception Handler notifies the User and gracefully exits the system.

Therefore, in this manner, recovery block contributes to fault tolerance of a system.

2) Define Markov Modeling. With example, explain Markov chain by using Transition Matrix for next and next to next states.

Ans:- Markov Chain Modeling is a probabilistic model used to describe a system's state transition over time in a stochastic process.

In layman terms, it is a type of modeling used to understand ~~the~~ what might happen next, without needing to know the entire history through the use of probability.

It's key concept is that: probability of moving to a new state depends only on the current state, not on how you got there.

Let us take an example of the following to explain Markov Chain.

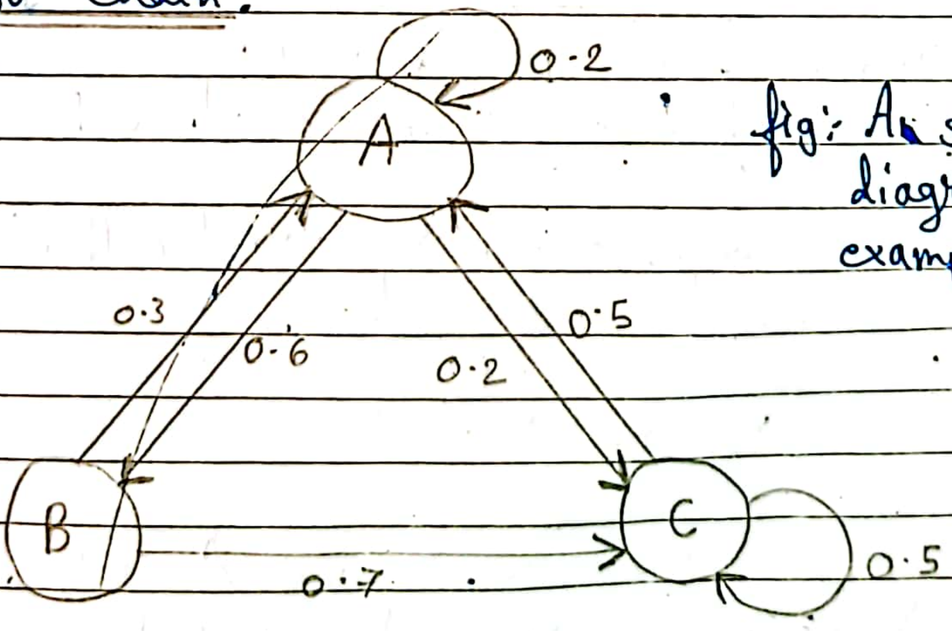


fig:- A state transition diagram of an example problem.

Here,

- A, B, C represent various states
- arrows represent transition function from one set to next
- numbers next to arrow represent probability of that transition taking place.

Now let us represent that state transition diagram as transition matrix.

	A	B	C
A	0·2	0·6	0·2
B	0·3	0	0·7
C	0·5	0	0·5

$$\begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix}$$

Here, each cell/element represent the probability of transfering from one state to another.

For example Matrix$_{12}$ value i.e 0·3 represent the probability of transition from state B to A.

Now let us take a row vector that denotes the probability of the state

$$\pi = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$$

lets suppose we have state A initially, thus the vector becomes, $\pi_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$

now,

$$\pi_0 A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = \pi_1$$

= ~~$[0.3 \quad 0.3 \quad 0.7]$~~

$$= \begin{bmatrix} 1 \times 0.2 + 0 \times 0.3 + 0 \times 0.5 & 0 \times 0.6 + 0 \times 0 + 0 \times 0 & 0 \times 0.2 + 0 \times 0.7 + 0.5 \end{bmatrix}$$

$$\pi_1 = \begin{bmatrix} 0.2 & 0.6 & 0.2 \end{bmatrix}$$

Now when we multiply the probablity of a state (represented by π) with transition matrix (A), we get the probablity of the next state.

i.e. when the system is at A, it has 0.2 probablity of transitioning to A, 0.6 probablity of transitioning to B & 0.2 probablity of transitioning to C.

Now to calulate the next of the next state, we do

$$\pi_1 A = \begin{bmatrix} 0.2 & 0.6 & 0.2 \end{bmatrix} \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.3 & 0 & 0.7 \\ 0.5 & 0 & 0.5 \end{bmatrix} = \pi_2$$

$$\pi_2 = \begin{bmatrix} 0.32 & 0.12 & 0.56 \end{bmatrix}$$

Now, there are probability of the system being in next of next state(π_2) when the next state(π_1) is taken into consideration

3> Define software rejuvenation. Give 10 examples of it.

Ans:- Software rejuvenation is a proactive fault management technique that improves software reliability by avoiding/postponing unanticipated software failures/crashes by alowing proactive repairs at the descretion of the user.

It involves intentionally terminating and restarting software components periodically to prevent the accumulation of internel errors or resource leaks that can lead to system degradation or failure over time.

Some examples of software rejuvenation are:-

i> Periodic Restart:-
Scheduled restarts of system or software components at regular interval to clear memory leaks and reset internal states.

ii> Rolling Update:-
Incrementally updating software components while maintaining system availability by gradually replacing outdated components with newer version.

iii> Cache Refresh:-
Refreshing cache contents periodically to prevent data staleness.

iv) **Connection Pool Reset:-** Resetting database connection pools or network connection periodically to release resources and prevent connection leaks.

v) **Session Timeout:-** Automatically expiring user sessions after a predefined period to release resources and prevent session-related cleanup.

vi) **Resource cleanup:-** Performing routine cleanup tasks to reclaim unused resources such as memory, file handles, and database connections.

vii) **Database Vacumming:-** Running database maintainance tasks like vacuuming or defragmentation to optimize performance and reclaim storage space.

viii) **Log Rotation:-** Rotating and archiving log periodically to prevent them from consuming excessive disk space and to facilitate easier analysis.

ix) **Load Balancer Cycling:-** Cycling through different instances of a service or application within a load balancer to distribute load evenly and ~~prevent~~, prevent any single instance from becoming overloaded.

10> Statemachine:-

Resetting the state of complex state machines or workflows periodically to prevent issues caused by accumulated state corruption or inconsistencies.