

-: Details :

Names : Shreyansh Dixit (23101040),
Mansij Kumar Jha (23101044),
Satish Vijay Rathod (23101009),
Priya Amit Kewalramani (23101003)

Subject : CAOS(PPT)

Branch : CSE

Cluster: A

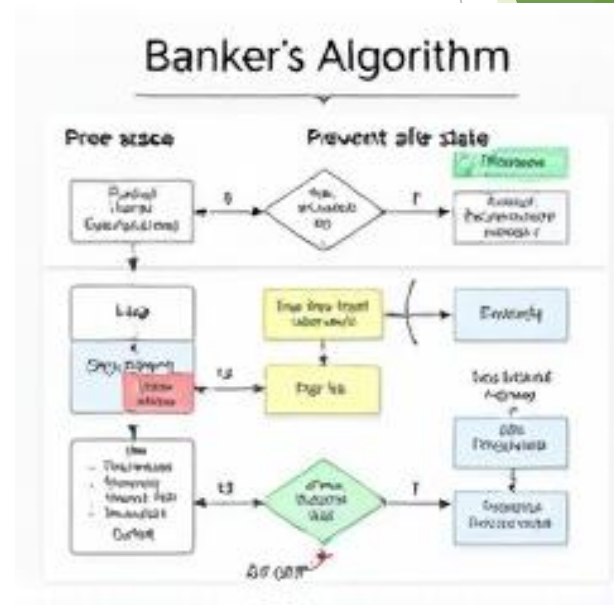
College : G.H.Raisoni College of the Engineering,
Jalgaon

Overview of Banker's Algorithm in Operating Systems

- ▶ Introduction to Banker's Algorithm
- ▶ Key Concepts of Banker's Algorithm
- ▶ State of a System
- ▶ Data Structures Used
- ▶ The Algorithm Steps
- ▶ Example Scenario
- ▶ Advantages of Banker's Algorithm
- ▶ Limitations of Banker's Algorithm
- ▶ Conclusion and Takeaways

Introduction to Banker's Algorithm

- ▶ **Algorithm Overview:** The Banker's Algorithm dynamically allocates resources ensuring safe states, preventing potential deadlocks in systems.
- ▶ **Role in Operating Systems:** It prioritizes safe resource allocation, enhancing system reliability and process efficiency in multitasking environments.
- ▶ **Practical Implications:** Real-world applications illustrate its utility, including scenarios in banking and scheduling systems for process management.



Key Concepts of Banker's Algorithm

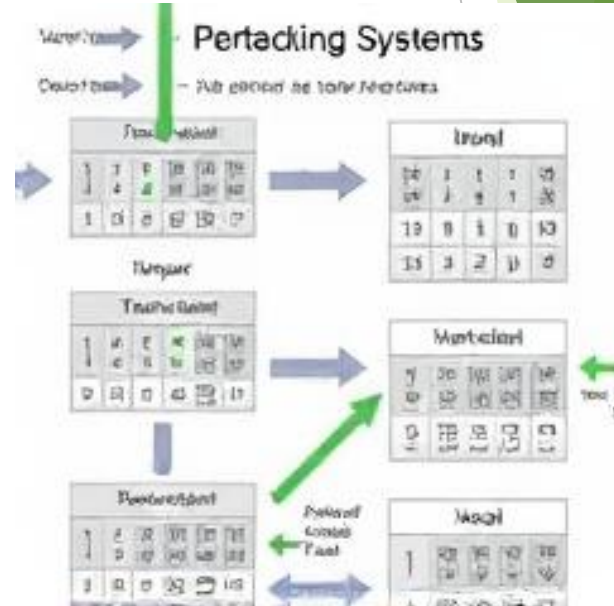
- ▶ **Resource Allocation:** Refers to distributing resources among processes to optimize performance and prevent deadlock situations effectively.
- ▶ **Maximum vs. Need:** Maximum indicates total required resources, while need defines remaining resources for processes to complete execution.
- ▶ **Available Resources:** Denotes the current free resources in the system; crucial for determining process scheduling feasibility.

State of a System

- ▶ **System State Definition:** A system state represents the current allocation of resources among processes, indicating overall system health.
- ▶ **Safe vs. Unsafe States:** Safe states allow all processes to complete without deadlocks; unsafe states risk potential deadlock scenarios.
- ▶ **State Evaluation Mechanism:** The algorithm assesses states by simulating resource allocation and checking process completion sequences accurately.

Data Structures Used

- ▶ **Allocation Matrices:** These matrices track currently allocated resources to each process, ensuring efficient resource management in the system.
- ▶ **Maximum Matrices:** They represent the maximum resources each process may request, essential for maintaining safe operating conditions.
- ▶ **Need Matrices:** Need matrices determine remaining resource requirements for each process, facilitating effective resource allocation decisions.



The Algorithm Steps

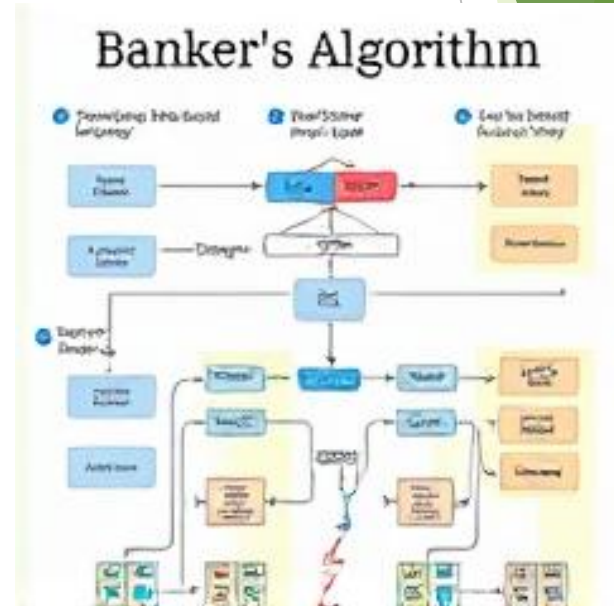
- ▶ **Request Evaluation:** Initial request validation checks process resource demands against current availability to ensure feasibility.
- ▶ **Safety Algorithm Execution:** Simulates potential resource allocations to verify safe states to conclude if processes can complete.
- ▶ **Resource Allocation:** Upon meeting safety conditions, resources are allocated, updating matrices to reflect new system state.

Example Scenario

- ▶ **Example Scenario Overview:** This scenario illustrates the Banker's Algorithm with specific processes, resources, and allocation requests.
- ▶ **Process and Resource Table:** A collaborative table detailing processes, their resource allocations, maximum requirements, and current availability.
- ▶ **Real-time Execution:** Demonstrating the step-by-step resource allocation and evaluation processes based on simulation results.

Advantages of Banker's Algorithm

- **Deadlock Prevention:** Effectively prevents deadlocks by ensuring only safe resource allocations are granted, maintaining system stability.
- **Efficient Resource Management:** Optimizes resource allocation processes, ensuring minimal wastage and enhancing overall system performance and responsiveness.
- **Predictable Behavior:** Allows for a predictable resource allocation strategy, enabling better planning and enhanced system performance predictions.



Limitations of Banker's Algorithm

- ▶ **Knowledge Requirement:** The algorithm necessitates prior knowledge of maximum resource needs from all processes, limiting flexibility.
- ▶ **Complexity Challenges:** Its complexity can lead to significant overhead, hindering performance in systems with numerous processes.
- ▶ **Limited Applicability:** Not suitable for dynamic environments where resource requirements frequently change, risking inefficient management.

Conclusion and Takeaways

- ▶ **Significance of the Algorithm:** The Banker's Algorithm is essential for managing resource allocation and preventing system deadlocks effectively.
- ▶ **Areas for Further Reading:** Explore topics like advanced resource management techniques and algorithms beyond the Banker's approach.
- ▶ **Practical Implementation:** Investigate real-world case studies that exemplify the Banker's Algorithm in various operating systems.