BANKERS ALGORITHM

The Banker's algorithm is primarily used in systems where multiple processes compete for a finite set of resources. Its purpose is to ensure that resources are allocated in a safe and deadlock-free manner. Deadlock refers to a situation where processes are unable to proceed because they are waiting for resources held by other processes, resulting in a system deadlock.

The algorithm works by maintaining information about the available resources, the maximum resource requirements of each process, and the currently allocated resources to each process. Based on this information, it predicts whether granting a request for resources from a process will lead to a safe state or potentially result in deadlock.

The basic steps of the Banker's algorithm:

1. **Initialization**: At the start, the system defines the total number of available resources of each type and the maximum resources required by each process.
2. **Request**: When a process requests resources, the system checks if the requested amount is less than or equal to the available resources. If it is, the system temporarily grants the resources to the process and checks if this allocation will lead to a safe state.
3. **Safety check**: The algorithm simulates resource allocation by assuming all pending requests are granted until a safe sequence is found or all processes have been checked. It uses an algorithm like the resource-allocation graph or a variant of the Banker's algorithm to determine if the system can avoid deadlock and satisfy all pending requests.
4. **Resource allocation**: If the requested resources will result in a safe state, the system allocates the resources permanently to the process. Otherwise, the request is denied, and the process must wait until sufficient resources become available.

The Banker's algorithm ensures that the system can avoid deadlock by carefully considering resource allocation requests. It avoids granting resources that could potentially lead to a state where a process cannot proceed due to resource unavailability caused by circular dependencies. It is widely used in operating systems to manage resource allocation and prevent deadlock in systems with multiple competing processes and limited resources.