操作系统

Operating system

孔维强

大连理工大学

内容纲要

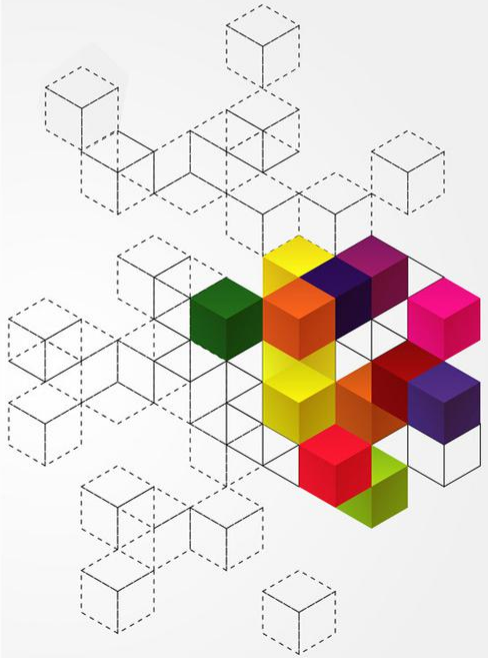
7.6 死锁检测

一、 单资源实例条件下的死锁检测

二、 多资源实例条件下的死锁检测

三、 死锁检测算法的代价分析

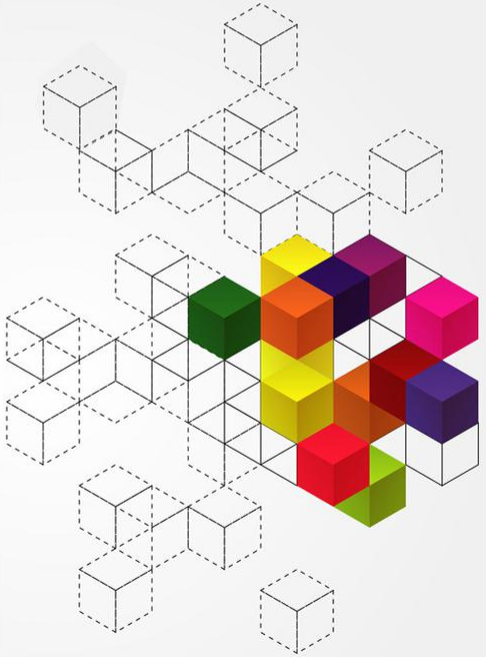
四、 死锁检测示例



一、单资源实例条件下的死锁检测

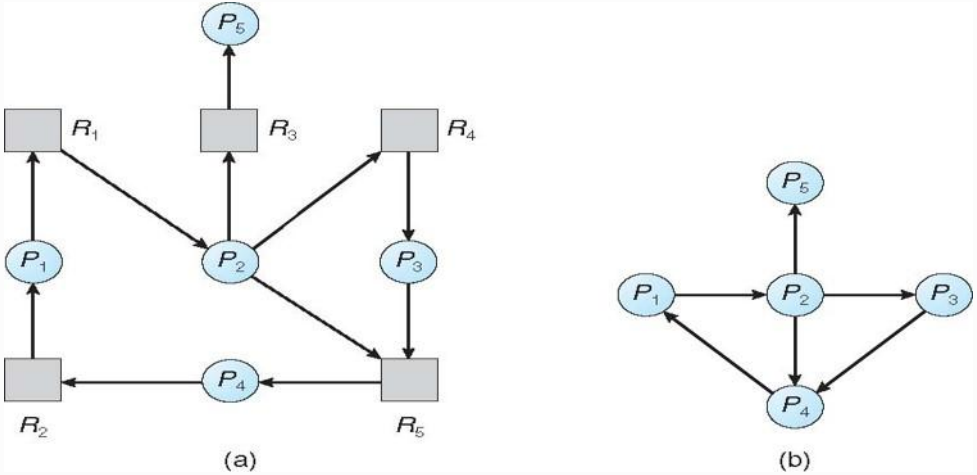
- **死锁检测**
  - 在系统运转过程中，定期进行死锁检测，判断是否有死锁发生
  - 需要设计专门算法来完成死锁检测任务

  - 从资源分配的角度开始考虑，死锁检测算法也要考虑单资源实例与多资源实例这两种不同情况

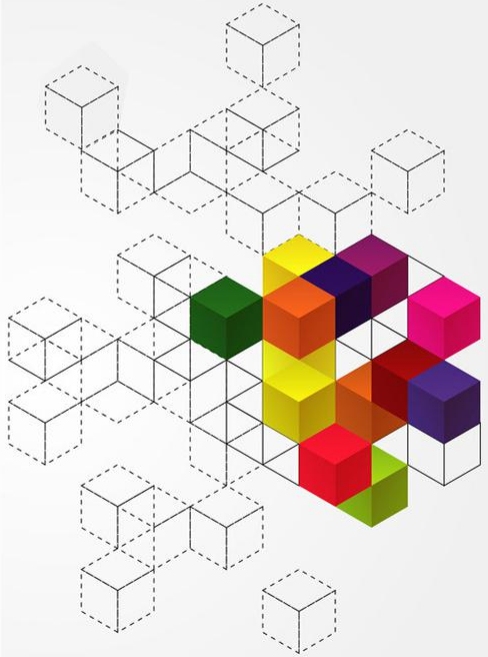


一、单资源实例条件下的死锁检测

- **基本方法**
  - 以资源分配图为基础，构建进程等待图

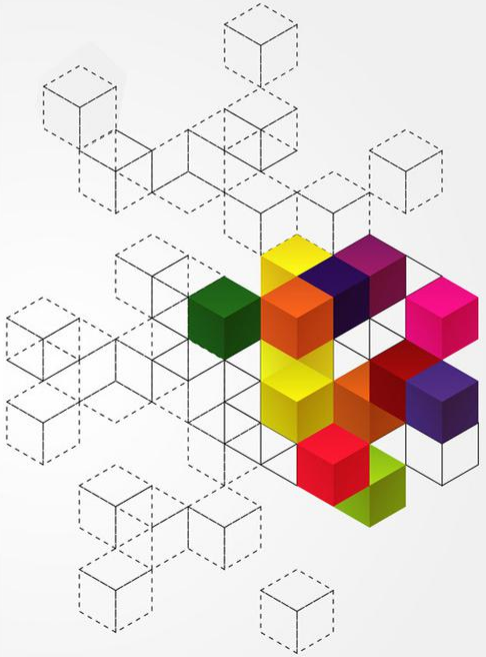


(a)资源分配图　　**(b)进程等待图**



一、单资源实例条件下的死锁检测

- **基本方法**
  - 以资源分配图为基础，构建进程等待图
  - 实现等待图中的环路检测
  - 定期启动对进程等待图的环路检测，如果发现图中有环，那么报告死锁发生

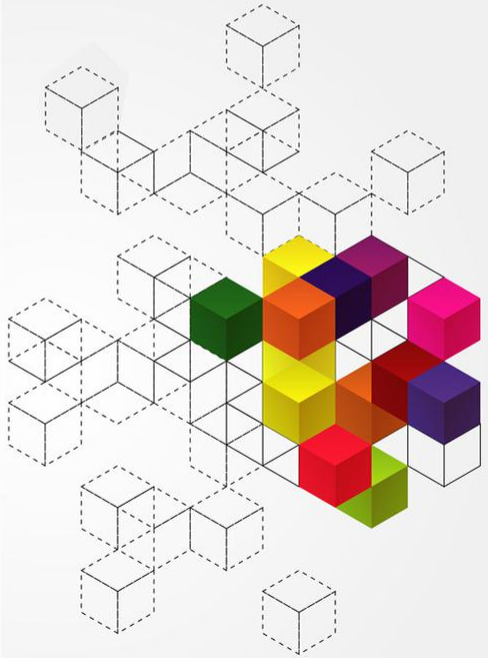


二、多资源实例条件下的死锁检测

基本数据结构（与银行家算法类似）

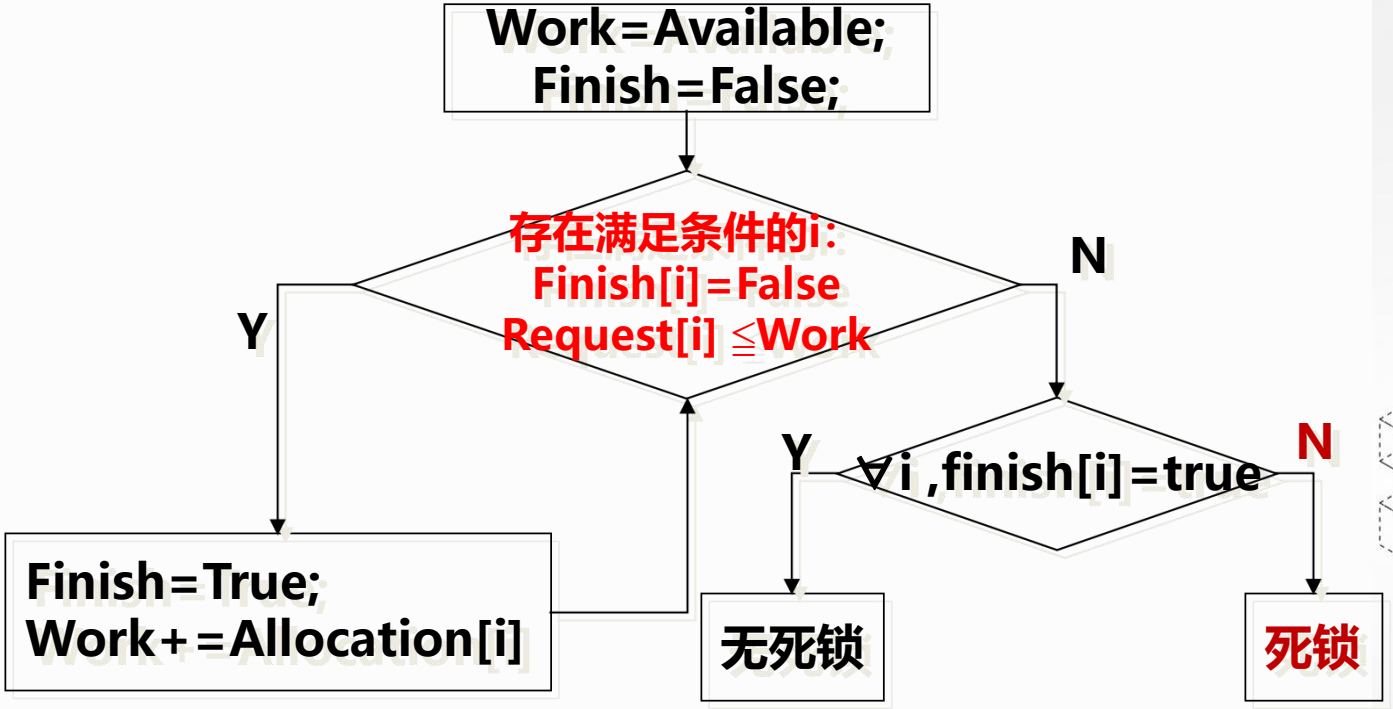
- Available:每类资源当前可用数量。是一个长为m的数组
- Allocation:每个进程当前已分配资源数量.
- Request: n x m 的进程资源请求矩阵

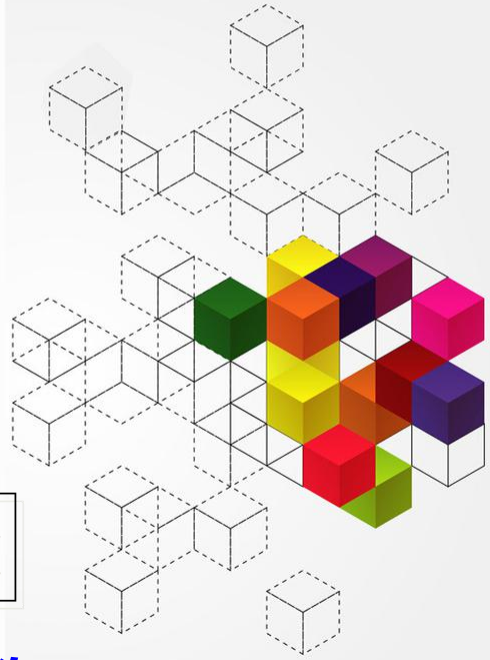
If Request [i][j] = k, then process Pi is requesting k more instances of resource type Rj.



二、多资源实例条件下的死锁检测

死锁检测算法流程





复杂度： O(m*n^2)，其中m-资源类数，n:进程数

四、死锁检测示例

多实例死锁检测

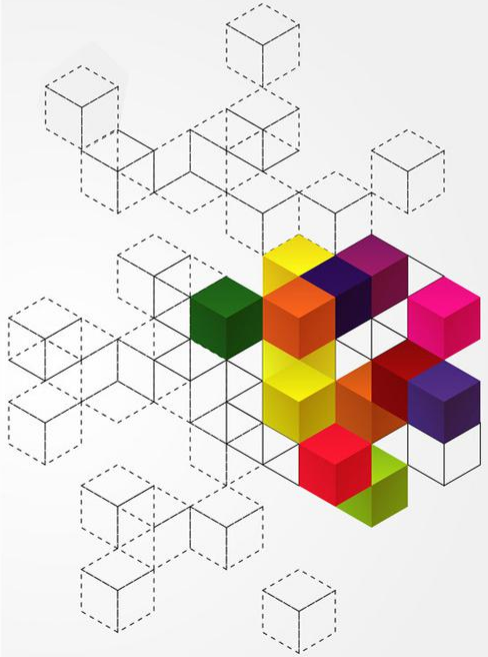
R={A(7),B(2),C(6)};

P={p0,p1,p2,p3,p4}

	Alloc	Request	Available	Work	Finish
	A B C	A B C	A B C	A B C	
P0:	0 1 0	0 0 0	0 0 0		
p1:	2 0 0	2 0 2			
p2:	3 0 3	0 0 0			
p3:	2 1 1	1 0 0			
p4:	0 0 2	0 0 2			

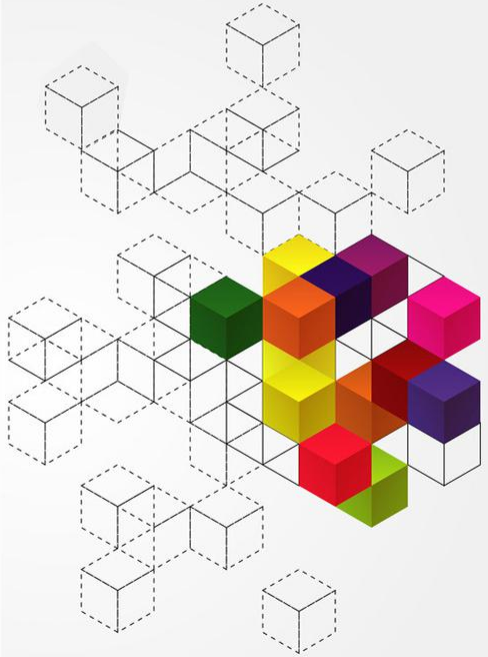
可以找到序列，使得进程能够按序完成

→ **结论：未死锁**



本讲小结

- 死锁检测算法
- 死锁检测算法代价分析
- 死锁检测方法示例



练习

Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix *Need*?

b. Is the system in a safe state?

c. If a request from process P_1 arrives for (0,4,2,0), can the request be granted immediately?