

系统模型

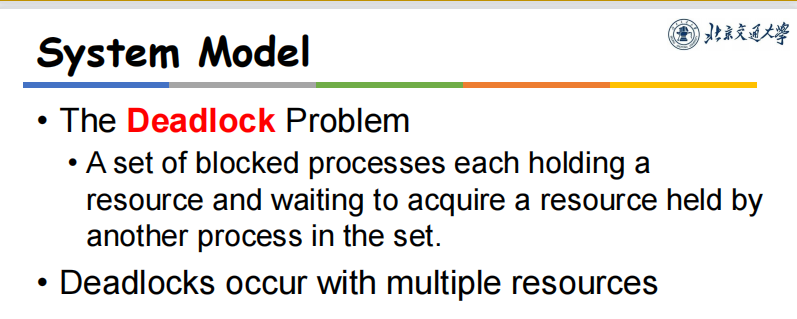
死锁特性

处理死锁的方法

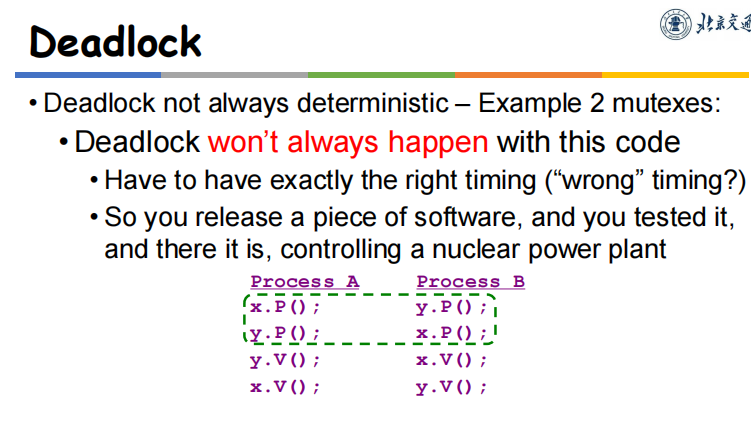
1. 防死锁
2. 死锁避免
3. 死锁检测
4. 从死锁中恢复

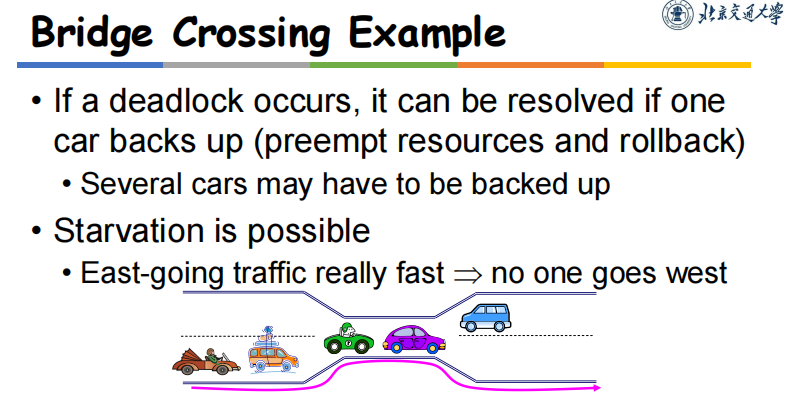
死锁的四个必要条件

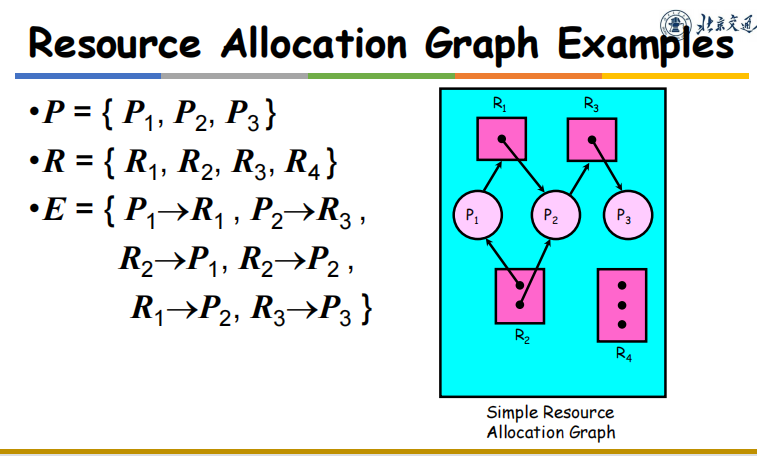
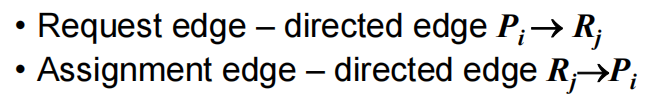
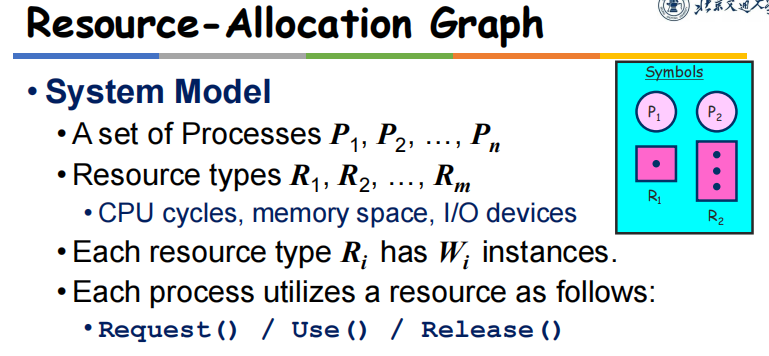
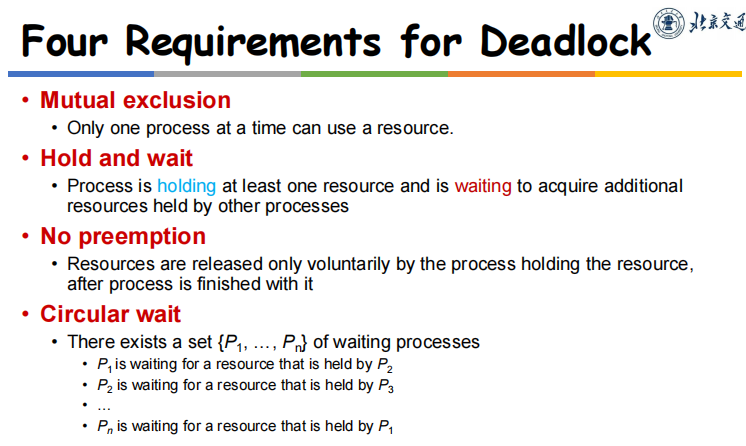
1. 互斥条件：资源是独占的且排他使用，进程互斥使用资源，即任意时刻一个资源只能给一个进程使用，其他进程若申请一个资源，而该资源被另一进程占有时，则申请者等待直到资源被占有者释放。
2. 不可剥夺条件：进程所获得的资源在未使用完毕之前，不被其他进程强行剥夺，而只能由获得该资源的进程资源释放。
3. 请求和保持条件：进程每次申请它所需要的一部分资源，在申请新的资源的同时，继续占用已分配到的资源。
4. 循环等待条件：在发生死锁时必然存在一个进程等待队列{P1,P2,…,Pn},其中P1等待P2占有的资源，P2等待P3占有的资源，…，Pn等待P1占有的资源，形成一个进程等待环路，环路中每一个进程所占有的资源同时被另一个申请，也就是前一个进程占有后一个进程所深情地资源。
5. 以上给出了导致死锁的四个必要条件，只要系统发生死锁则以上四个条件至少有一个成立。事实上循环等待的成立蕴含了前三个条件的成立，似乎没有必要列出然而考虑这些条件对死锁的预防是有利的，因为可以通过破坏四个条件中的任何一个来预防死锁的发生。



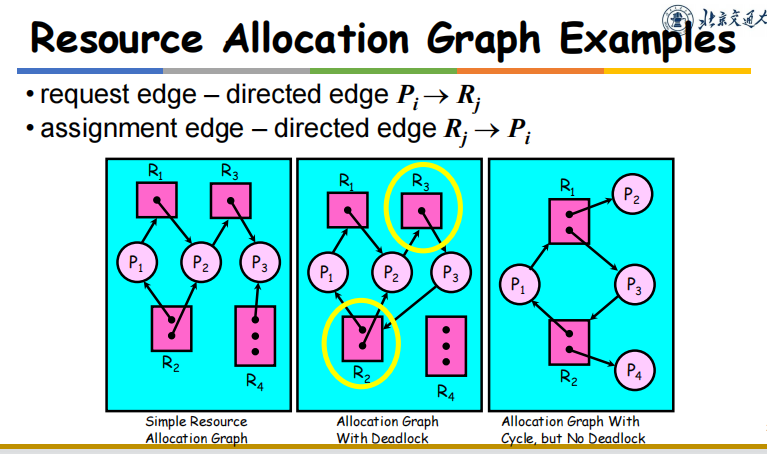
死锁不总是死锁：

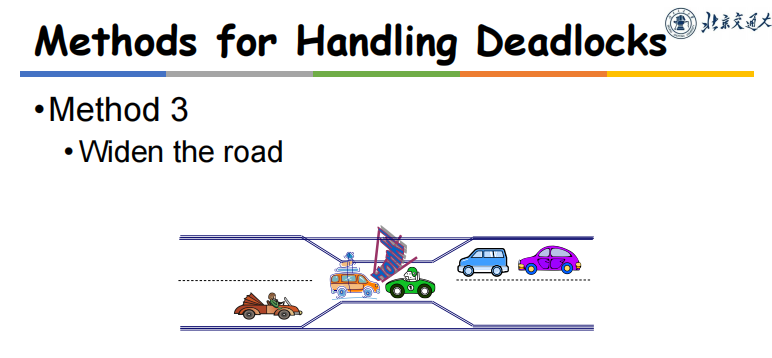
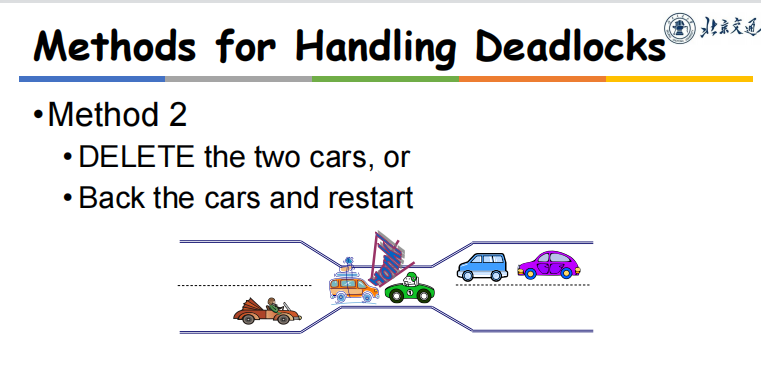
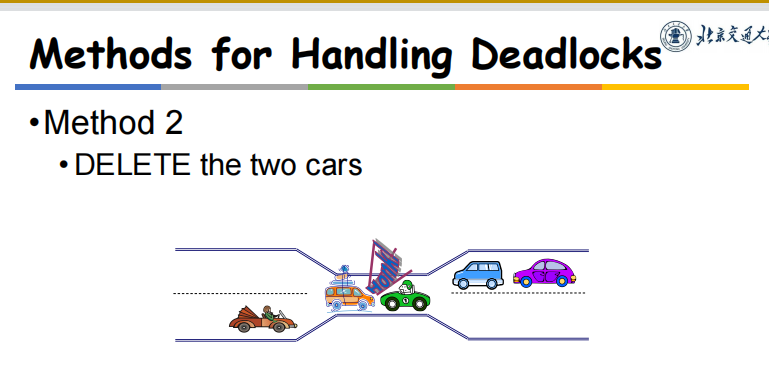
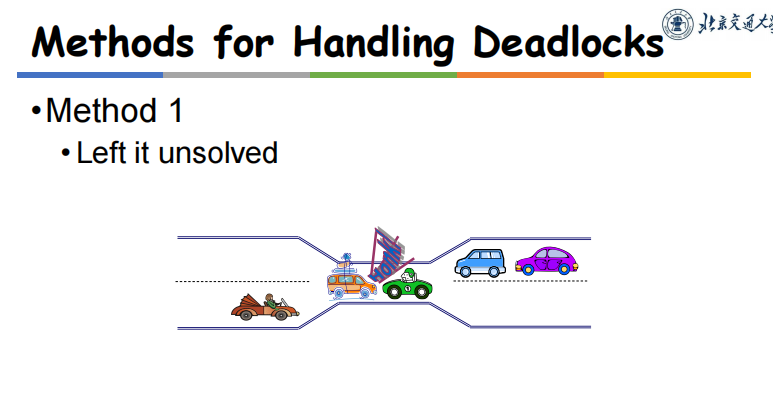


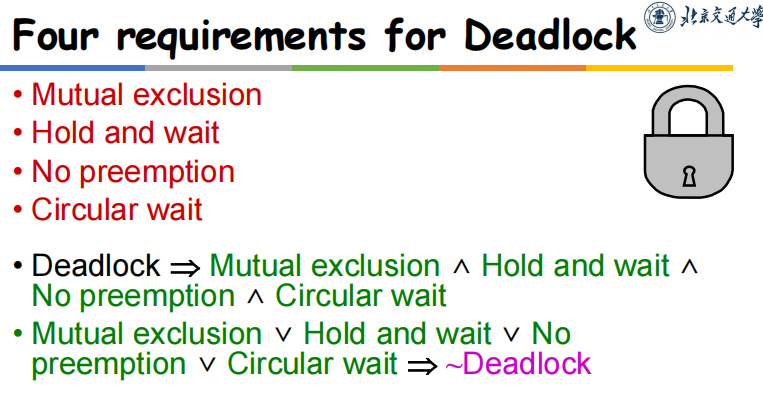
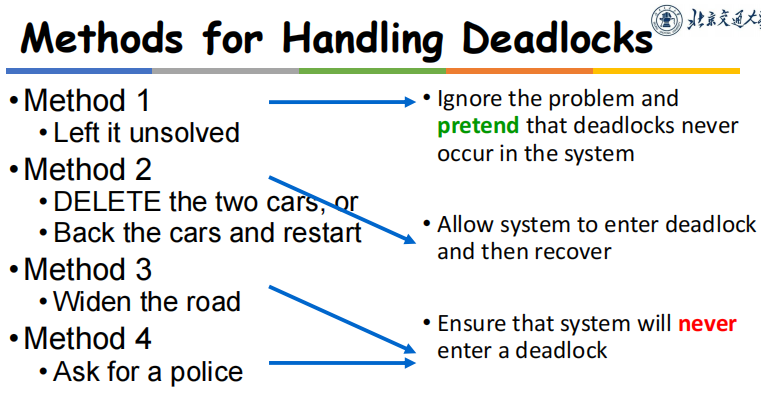
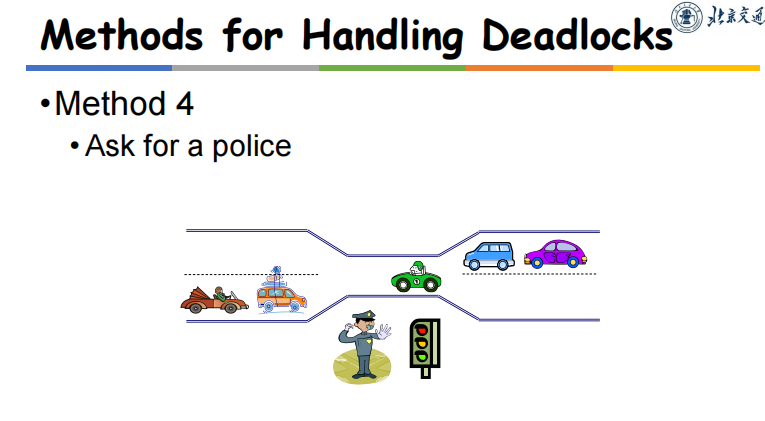
可能会导致饥饿，就是没人去西边

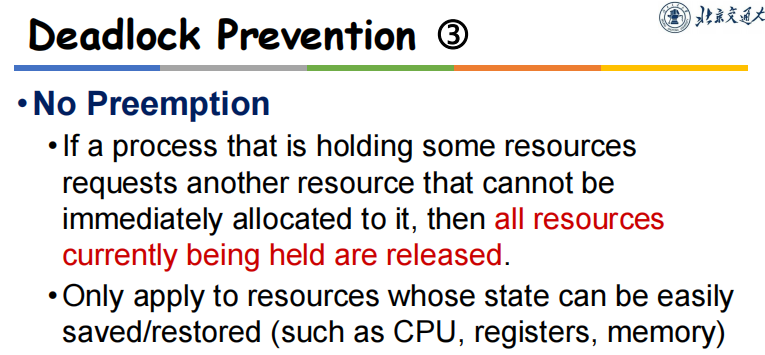
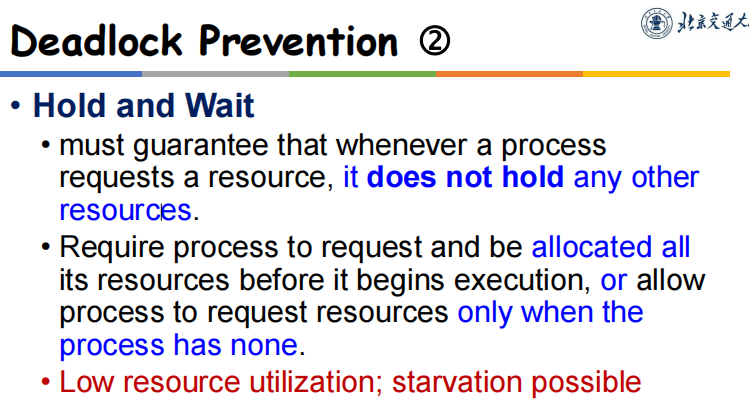
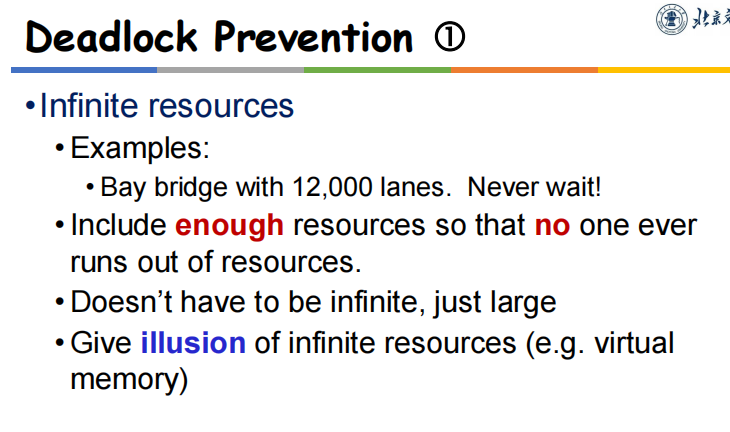
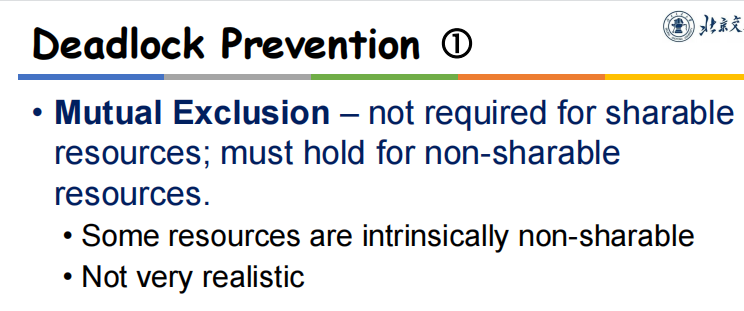


第三种情况比较特殊，此时资源很多，只要分配一个R1就可以让P1 P3进行下去了。







拿的资源不共享无优先购买权

如果一个持有一些资源的进程

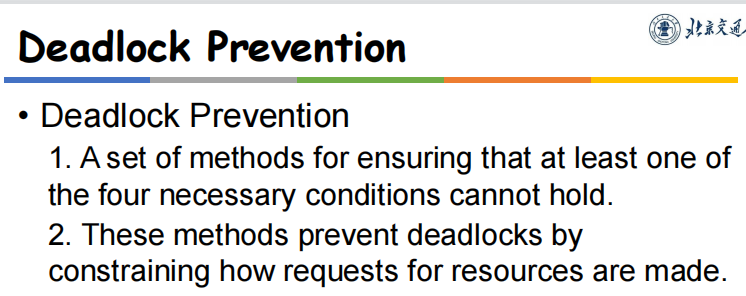
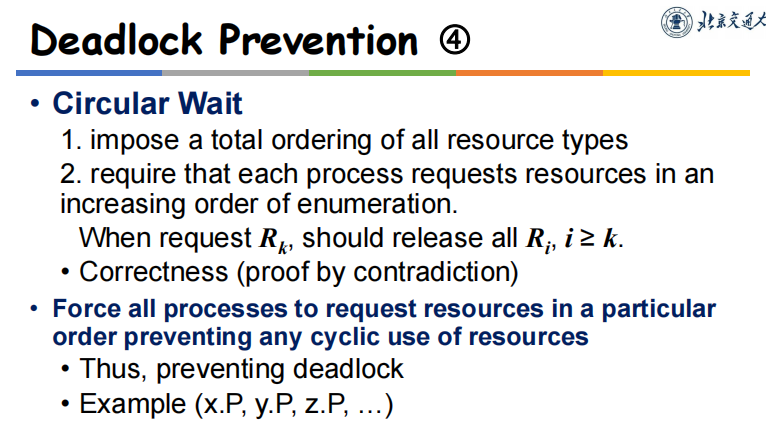
请求另一个不能使用的资源

立即分配给它，然后是所有的资源

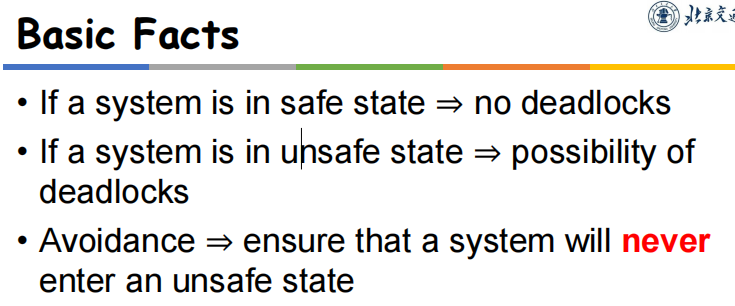
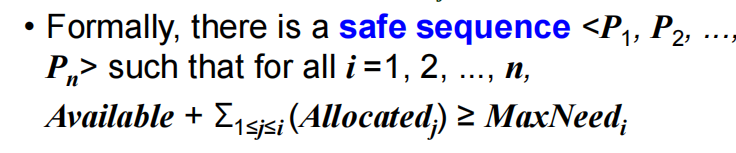
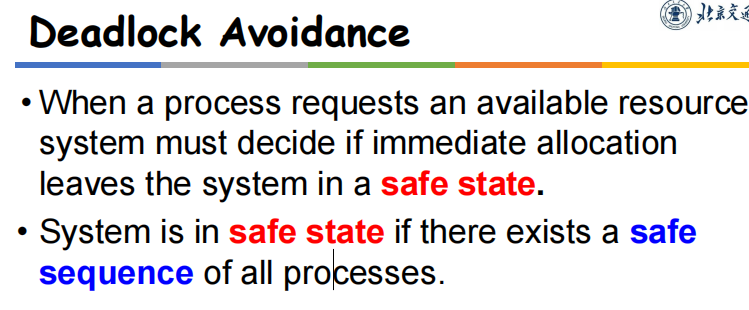
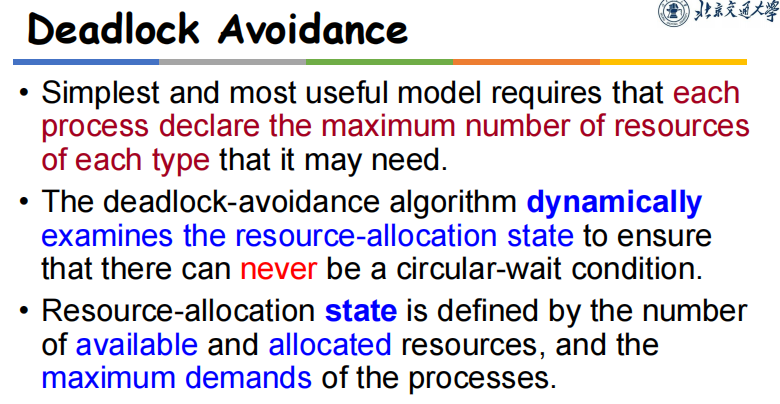
目前正在被关押的人将被释放。

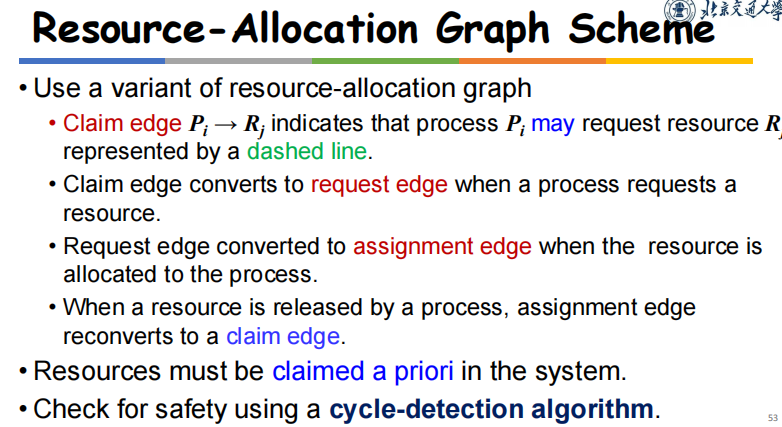
只适用于状态可以容易的资源

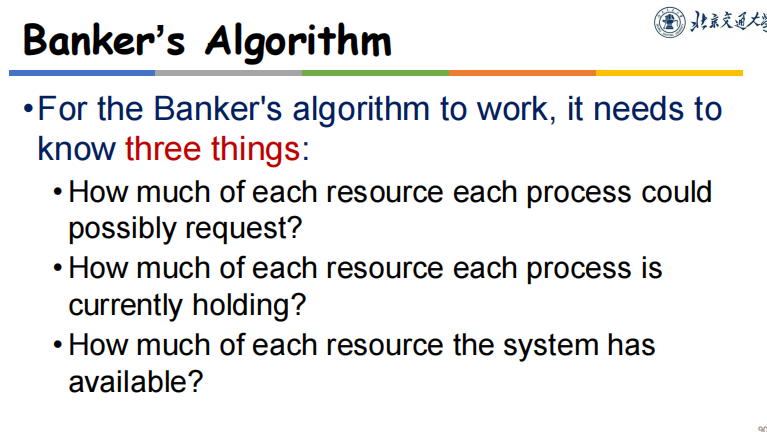
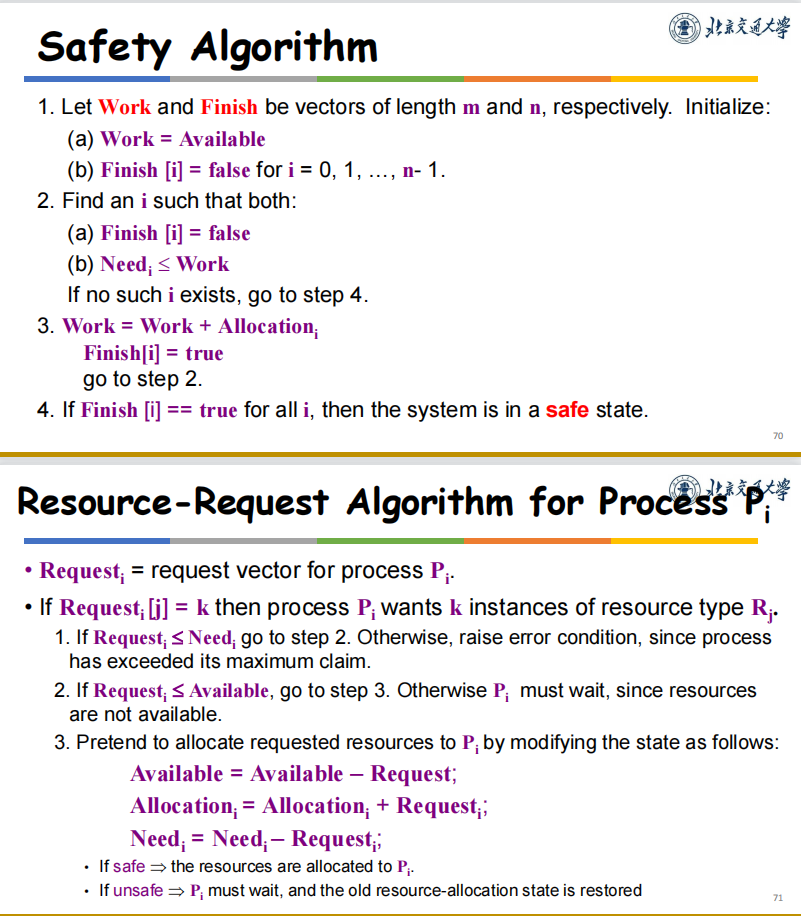
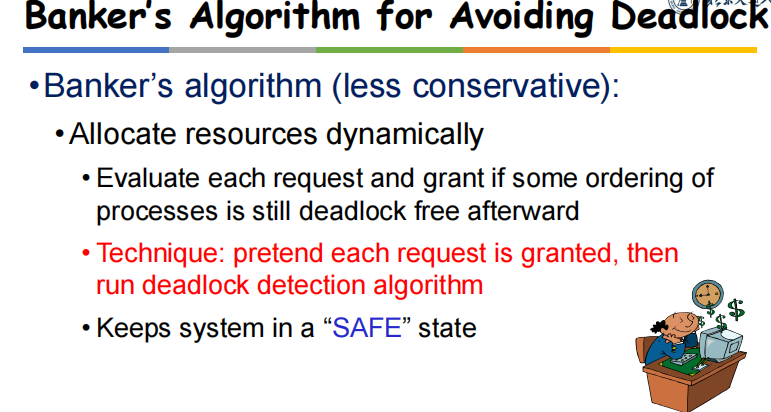
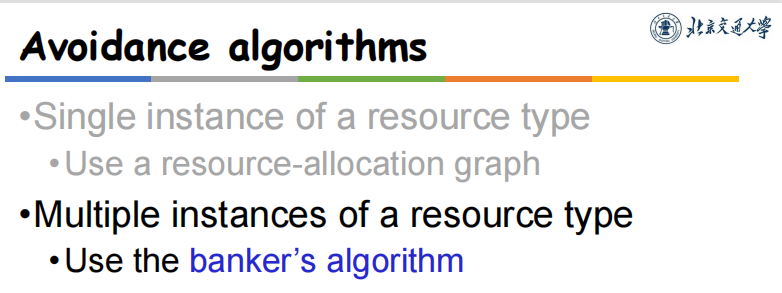
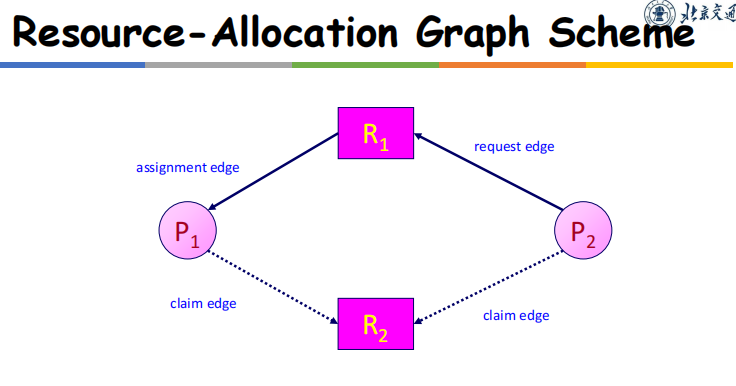
已保存/恢复（如CPU、寄存器、内存）

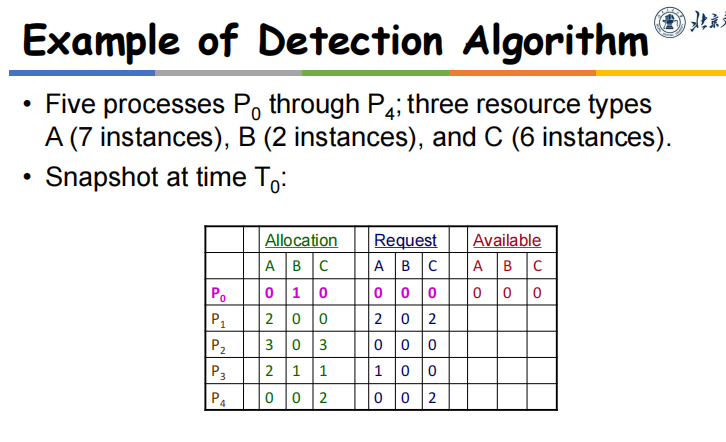
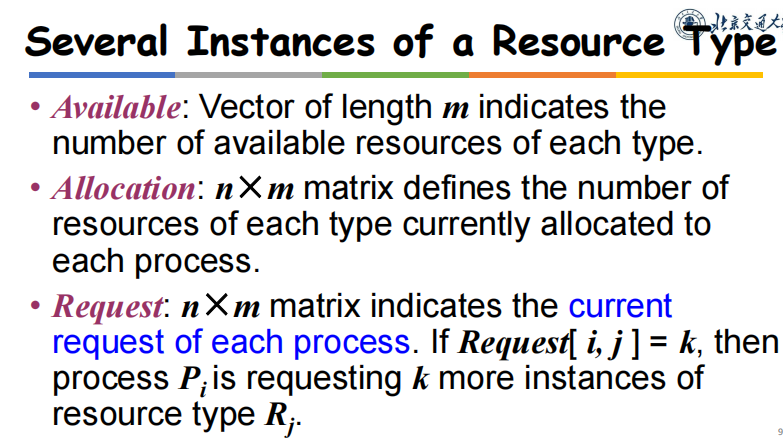
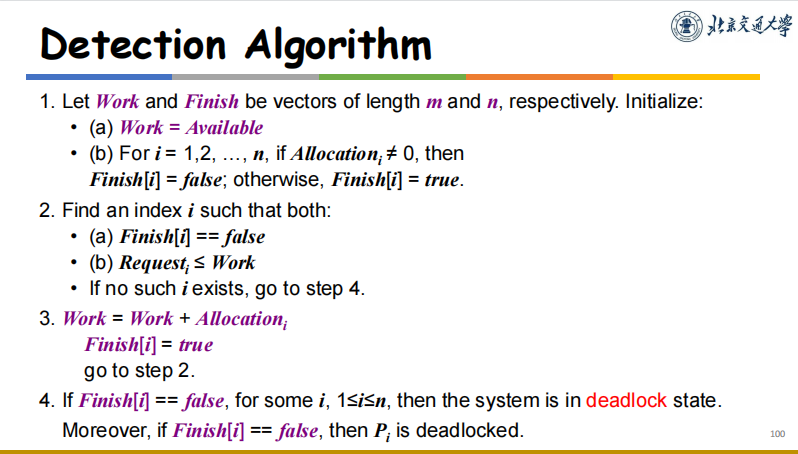
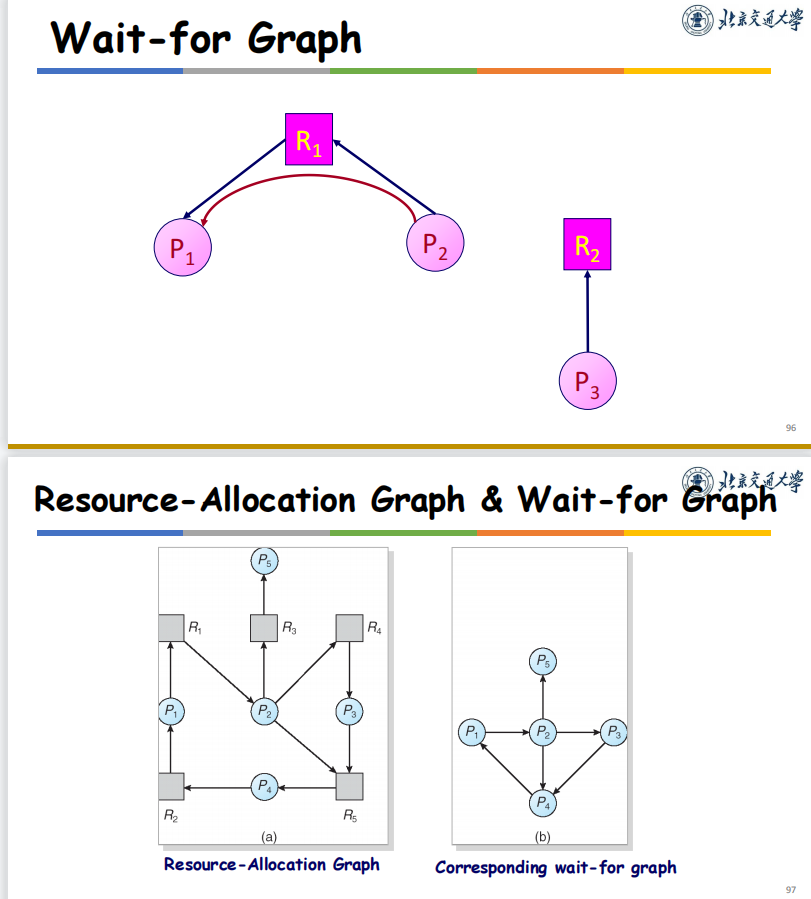
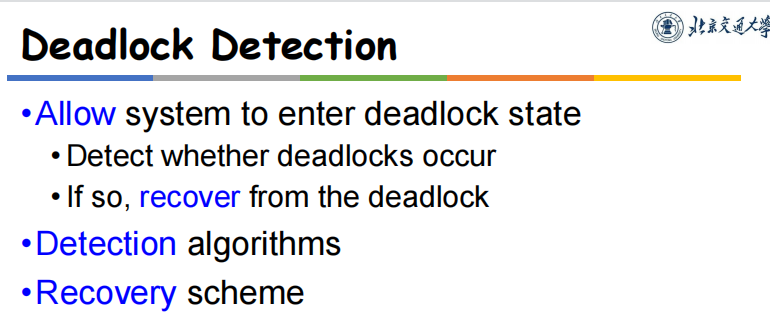


动态检查：





三种状态：申明想要--正在请求--已经分配了。





方法1：终止进程（按照某种顺序）

方法2：回滚

方法3：资源权限

