资源分配图(RAG)

下面我们来探讨资源分配图这个概念

资源分配图(RAG:RESOURE ALLOCATION GRAPH)

用有向图描述系统资源和进程的状态

二元组G=（V，E）

V：结点的集合，分为P(进程)，R(资源)两部分

P = {P1, P2, … , Pn}

R = {R1, R2, … , Rm}

E：有向边的集合，其元素为有序二元组

(Pi, Rj) 或 (Rj, Pi)

但是呢，这样去做呢很抽象，所以我们呢就把它简化成一种图示的方法

资源分配图画法说明

系统由若干类资源构成，一类资源称为一个资源类；
每个资源类中包含若干个同种资源，称为资源实例

资源类：用方框表示

资源实例：用方框中的黑圆点表示

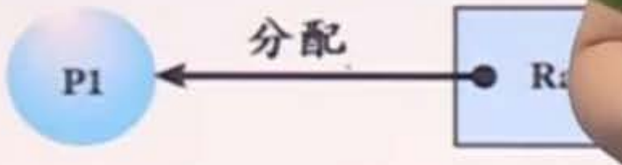
进程：用圆圈中加进程名表示

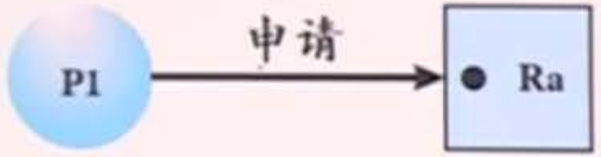
分配边：

资源实例 → 进程

申请边：

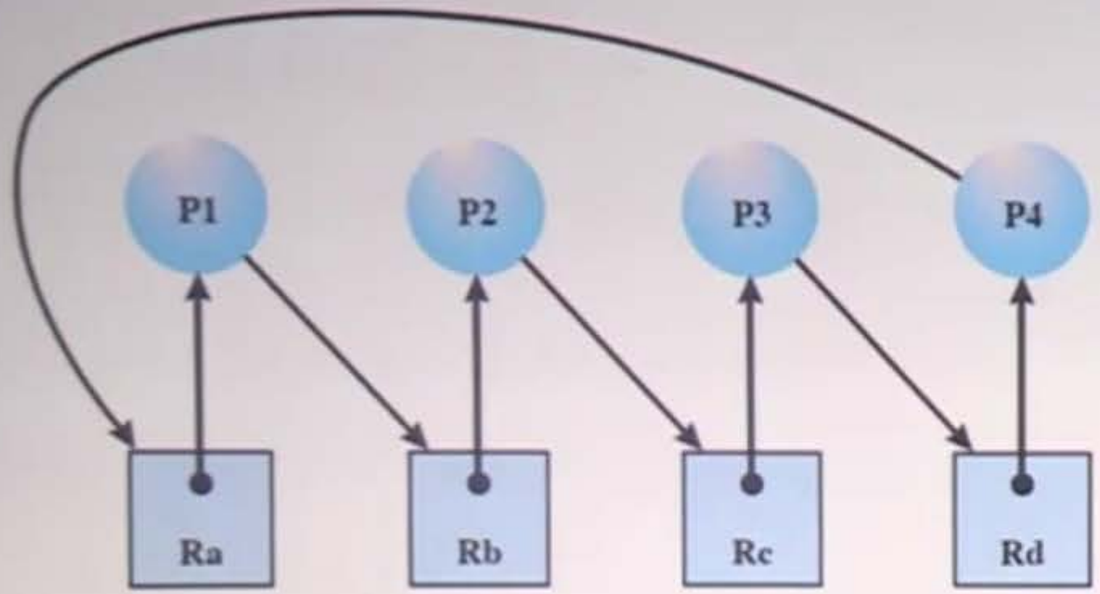
进程 → 资源类





William Stallings

例子：十字路口

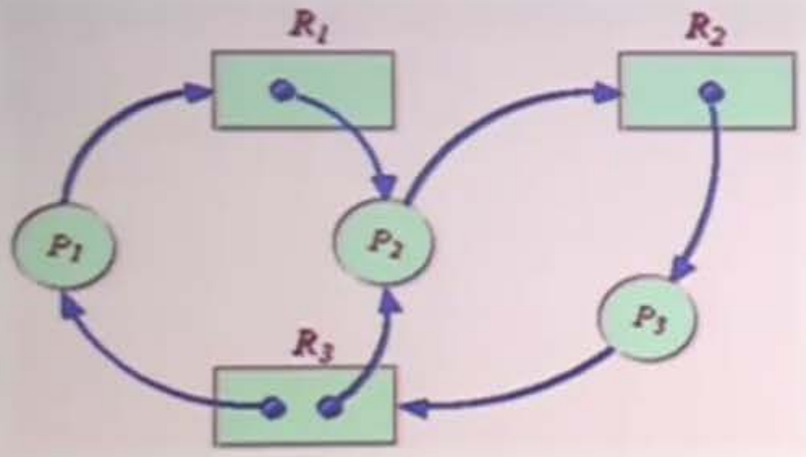


有一个环路 这四个进程，一个互相等待的这么一个循环

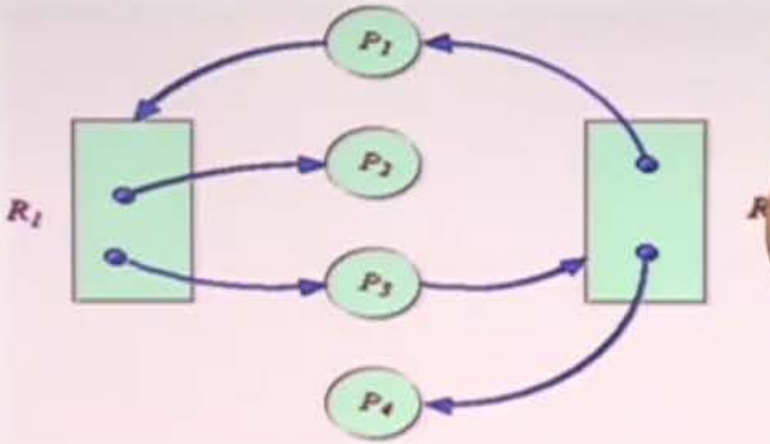
William Stallings

死锁定理

- 如果资源分配图中没有环路，则系统中没有死锁，如果图中存在环路则系统中可能存在死锁
- 如果每个资源类中只包含一个资源实例，则环路是死锁存在的充分必要条件



有环有死锁



有环无死锁

右边这张图呢，是有环路但是没有死锁 那么怎么样能够判断出来

8:08 / 12:10

资源分配图化简

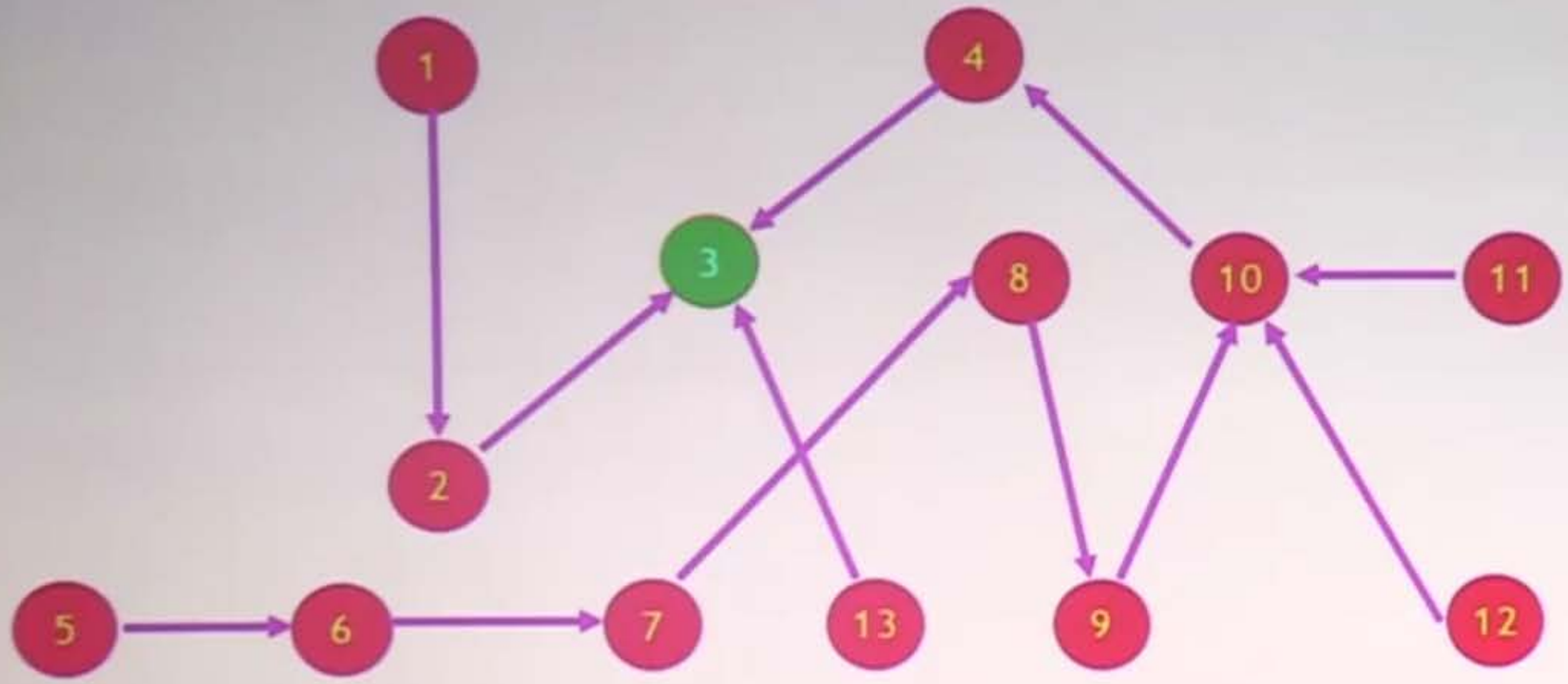
化简步骤：

1）找一个非孤立、且只有分配边的进程结点
去掉分配边，将其变为孤立结点

2）再把相应的资源分配给一个等待该资源的进程
即将该进程的申请边变为分配边

重复1）、2）

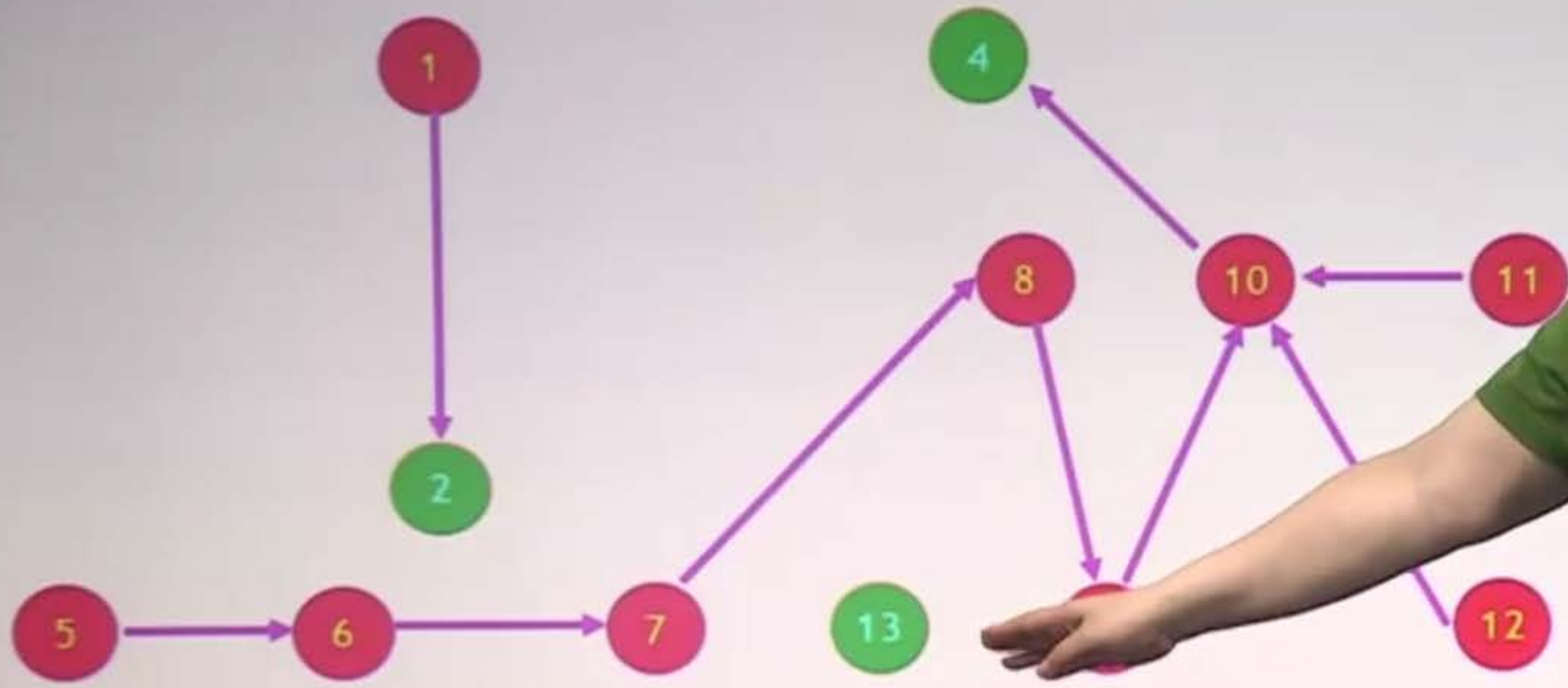
资源分配图化简的例子

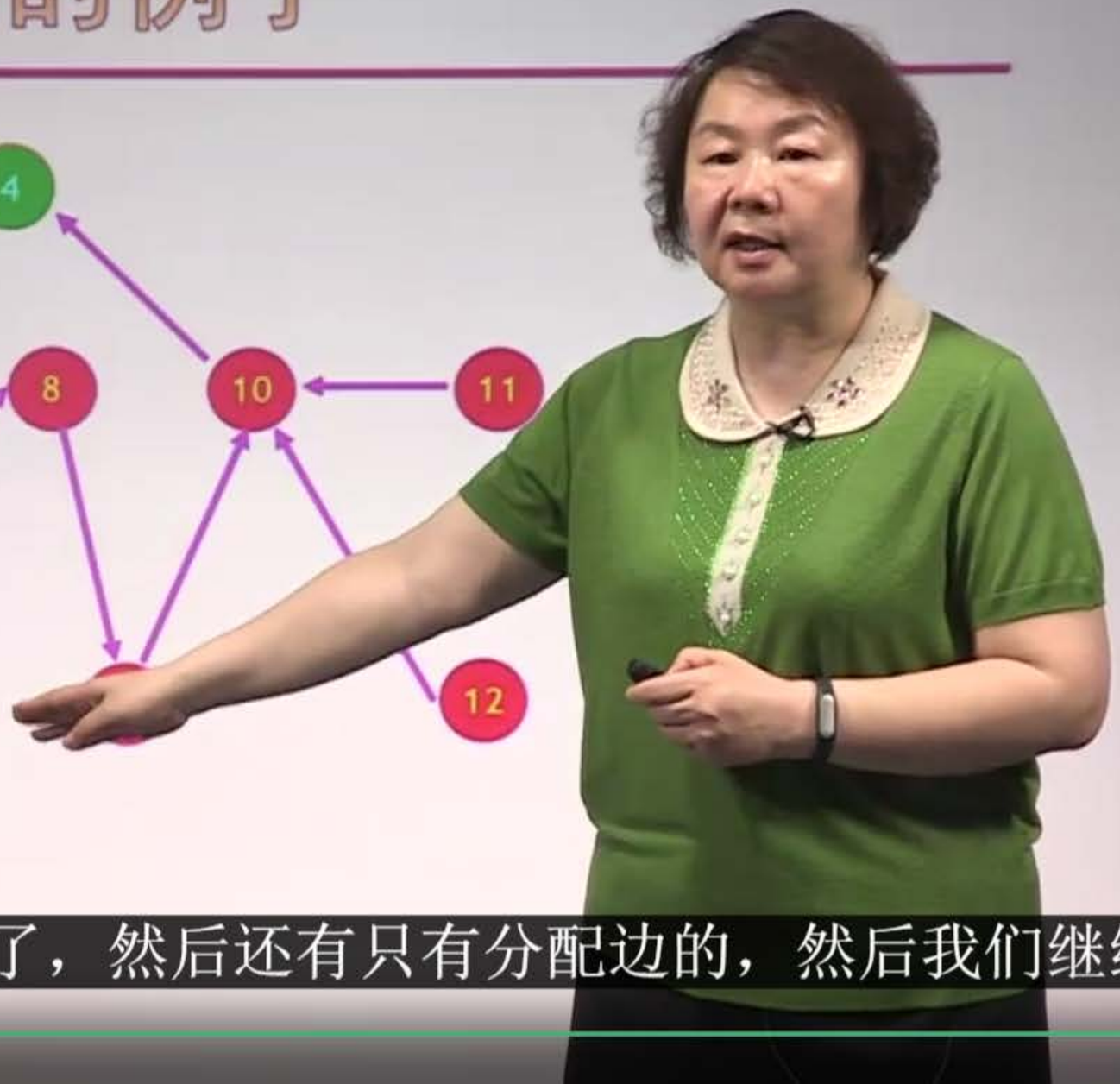


3 号进程的分配边给它去掉

然后把相应的资源分配给其它的进程，那我们可以做到这样一个过程

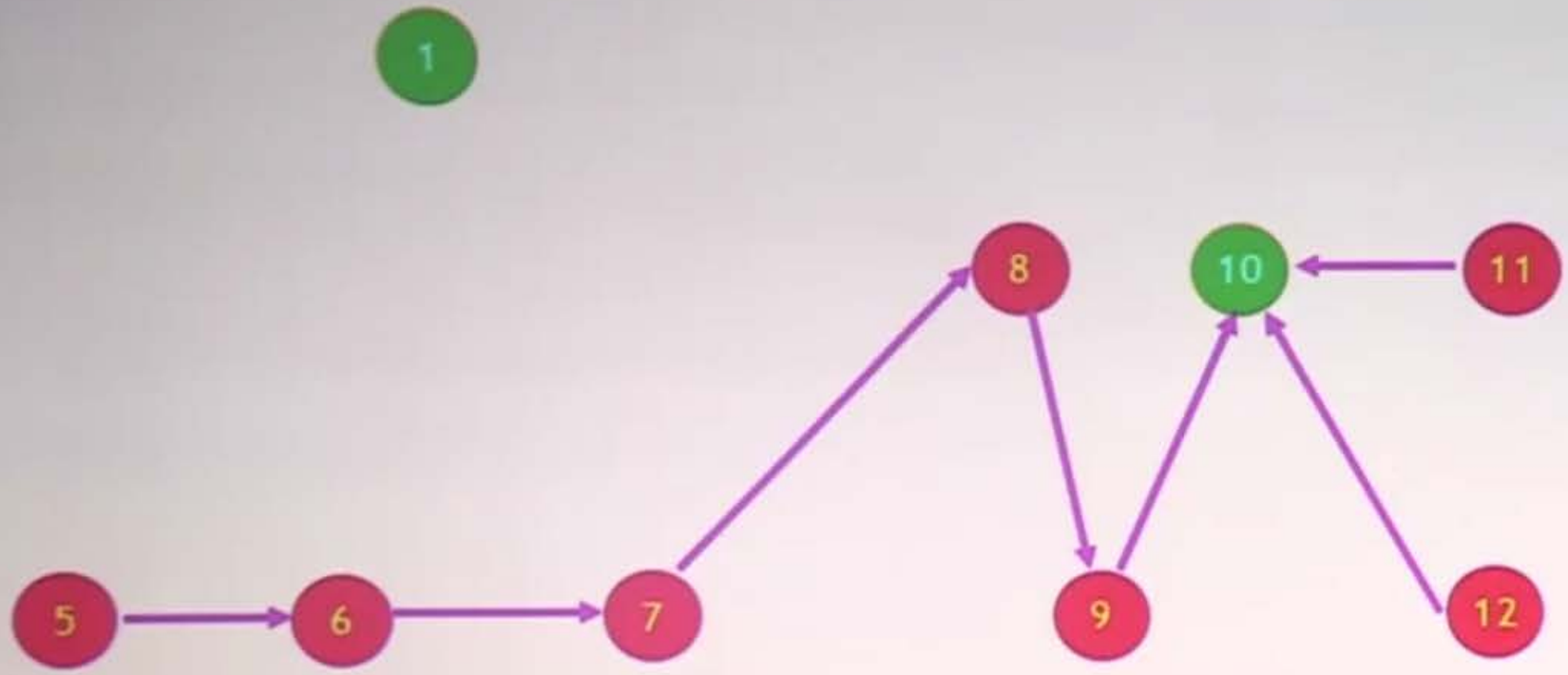
资源分配图化简的例子





系统中剩下当然这是孤立结点了，然后还有只有分配边的，然后我们继续啊

资源分配图化简的例子



把它化简 这又是只有分配边的，继续化简

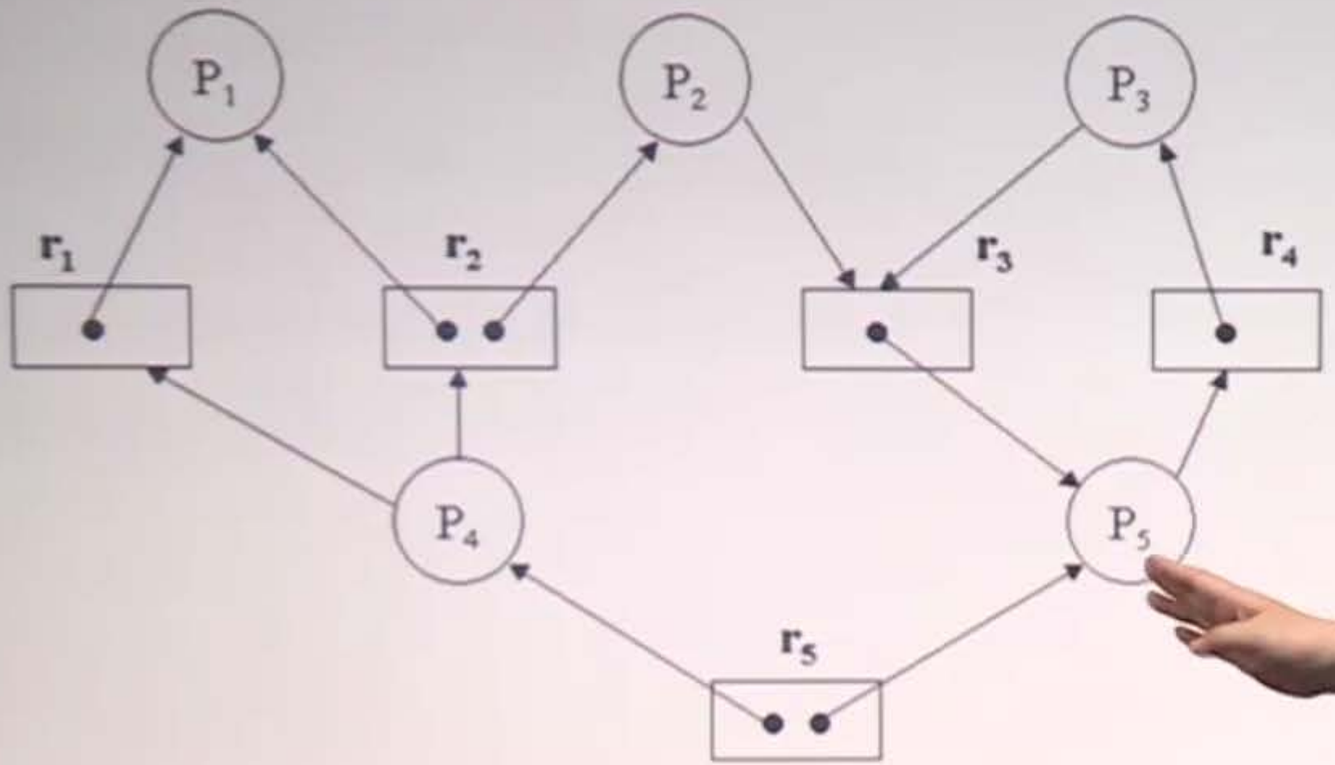
资源分配图化简的例子

最后我们可以看到

资源分配图化简的例子

这个图完全被化简了，所以没有死锁

练习



0:00 下面我们来探讨资源分配图这个概念 资源分配图呢，实际上是为了解决死锁问题 我们用有向图来描述系统的资源和进程的状态 那么从图的角度看看能不能为解决死锁问题提供相应的理论 和实践的依据。 那我们定义了一个二元组 它有两个集合，第一个集合呢 V 是结点的集合。 我们的结点分成了两类，一类是进程 一类是资源，啊，所以我们可以看到有两个，啊 结点的集合，一个是进程结点的集合，一个是资源结点的集合 另外呢，有一个是有向边的集合 有向边的集合呢，它就是相当于说，这个集合当中的每个元素呢，是一个有序的二元组 或者是从进程结点指向 资源的这样一个有向边 或者是反过来，从资源指向进程结点的一个有向边 那么我们可以用资源分配图 来刻画，啊，某一时刻系统的状态 资源是什么情况，进程是什么情况 但是呢，这样去做呢很抽象，所以我们呢就把它简化成一种图示的方法 下面我们来看一下，资源分配图画图的一个简单的说明 [空白_录音] 我们先做一下分类 整个系统呢，是由若干类资源组成 那么一类资源呢，就称为一个资源类 那么每个资源类当中呢其实可能会包含了若干个同种类型的资源 我们就把它称之为资源的实例。 那么如果我们 这样进行了划分之后，那么我们的这个 资源分配图它的这个图示呢，就需要这样几个要素 一个呢，是资源类，那我们是用一个方框来表示资源类 一个是资源的实例，那么资源的实例呢是在资源类当中的 啊，成分。 所以我们是用方框当中的一个实心的黑圆点来表示 还有一类呢，是叫进程。 我们是通过一个圆圈加上进程名来表示 我们给一个例子，那么在这个例子当中，我们可以看到这个方框 就表示是一类资源，当然这里头这一类资源当中只有一个资源实例 然后呢，这是进程，啊，进程用一个 圆圈加上进程名。 那么 如果我们有了这样几个成分之后呢，我们就可以定义，啊 或者是画出来在系统当前，啊，某一时刻所处的一个状态 那么这个状态当中呢，除了我们有这样一些要素之外呢，我们还有这些要素之间的一个关系 这个关系的第一种关系就叫分配边，也就是从资源实例 指向进程的一个有向边，我们来看，这是资源 那么这是其中一个资源实例 这个实例如果分配给了这个进程 P1 那么 就从这个实例引出一条有向边，指向这个进程 第二个呢，是另外一条叫 申请边，也就是从进程啊，指向资源类的这样一个有向边 下面我们看一个例子，这个例子当中呢 这个进程呢，想申请这个资源的某一个 实例，那么它不关心它申请的哪一个实例 它是从进程指向这个资源类的 啊，给画出一个有向边，那么这就是一个申请边 也就是说，这个进程申请这类资源 那么我们用了这样一个 描述呢，我们就可以把系统当中的进程 还有资源，它们之间的关系、 状态把它刻画出来 这是一个用资源分配图来 刻画

的，啊，十字路口出现的死锁状态 我们来看一下，进程 P1, P2, P3, P4，分别代表的是不同方向的 汽车，那么 P1 拿到了一个路口 进入了一个路口，它呢，需要另外一个路口，啊，通过 因此呢，我们可以看，它得到了一个资源，然后它去申请 另外一个资源，所以有一个分配边是它得到的路口 还有一个申请边，是它要想申请的那个路口 那么第二辆车，第三辆车和第四辆车都是如此 那么我们看到第四辆车，它得到了一个路口，然后但是它要 申请另外一个路口，所以我们可以看到这里头，啊，出现了死锁 啊我们看到有一个环路，啊，我们可以这样来看，从这儿 有一个环路 这四个进程，一个互相等待的这么一个循环 下面，针对资源分配图给出一个死锁定理 所谓死锁定理是指 如果资源分配图当中没有环路 那么系统中就没有死锁发生 如果资源分配图当中存在着环路 则系统中可能存在死锁，因为我们说 环路是死锁存在的一个必要条件，而不是充分条件 但是，如果每个资源类中 只有一个资源实例，那么环路就是死锁存在的充分必要条件 我们来看两个例子 左边这张图呢，有环路 我们可以看，P1 它去申请了 R1 那么这个时候 R1 呢被 P2 所占有，所以 P1 等 P2 然后 P2 呢，又等待 P3 P3 呢又等待 P1，我们可以看到有这么一个环 右边这张图呢，实际上我们也看到也可以存在一个环 比如说 P1 等待着 P3 P3 呢在等 P1，那么好像也有这么一个环 可以存在，但是呢，我们讲过了，那么由于这个 其中啊，有的资源类中，啊，不止一个资源实例 所以我们说，有环路可能存在死锁 所以在这两张图当中，左边这张图呢，是有环有死锁 右边这张图呢，是有环路但是没有死锁 那么怎么样能够判断出来 真的有死锁发生呢？我们来给一个 资源分配图的化简方法，通过对资源分配图进行化简，来判断这个 系统当前是不是出现了死锁。 我们给出一个化简的步骤 首先，我们在资源分配图当中，找一个非孤立的 且只有分配边的这样一个进程结点，也就是说这个进程 它不是一个孤立的结点，它有边，但这个边呢，只是分配边 然后，我们把分配边去掉 去掉分配边就把这个进程变成了一个孤立结点 第二步呢，是把 相应的资源分配给一个等待该资源的进程，因为我们知道 去掉了分配边，啊，那么相当于把这个资源归还给系统 所以呢，就把这个资源 分配给需要它的进程，把这个进程的申请边呢，就变为分配边 第一步和第二步呢，要反复的进行 直到我找不到满足这样条件的 进程结点，那么这个资源分配图的化简过程就完成了 那么完成了之后，如果系统中的所有进程结点，都是 孤立的了，那么这个时候我们就说系统中没有死锁发生 否则的话呢，系统中一定有环路存在 下

面呢，我们来看一个例子，啊，来 演示一下这个资源分配图的化简过程 但是由于这个，这么多的进程 我们从画法上呢，给一个简化，我们假设 进程 P1 等待某一个资源，就申请某一个资源，得不到，等待 那么这个资源呢，被 P2 所占有那么我们就 把资源的这个部分给它省略 直接画出来，1 等 2，就说明进程 1 等待进程 2 所占有的资源，这样呢我们简化一下资源分配图的画法 好那么我们这样，这是一个当时的系统的一个状态 我们按照刚才的资源分配图化简的这个过程，我们可以看到 3 号儿进程，它是只有分配边的 那么这样的话，我们就把 3 号进程的分配边给它去掉 然后把相应的资源分配给其它的进程，那我们可以做到这样一个过程 [空白_录音] 然后我们再选择 系统中剩下当然这是孤立结点了，然后还有只有分配边的，然后我们继续啊 把它化简 这又是只有分配边的，继续化简 最后我们可以看到 这个图完全被化简了 所以呢我们说这个当时，刚才我们那张图的状态，啊，系统状态 是没有死锁状态 这里给出一个练习 大家可以看一下，啊，在这样一个 资源分配图中当中是不是有死锁发生。