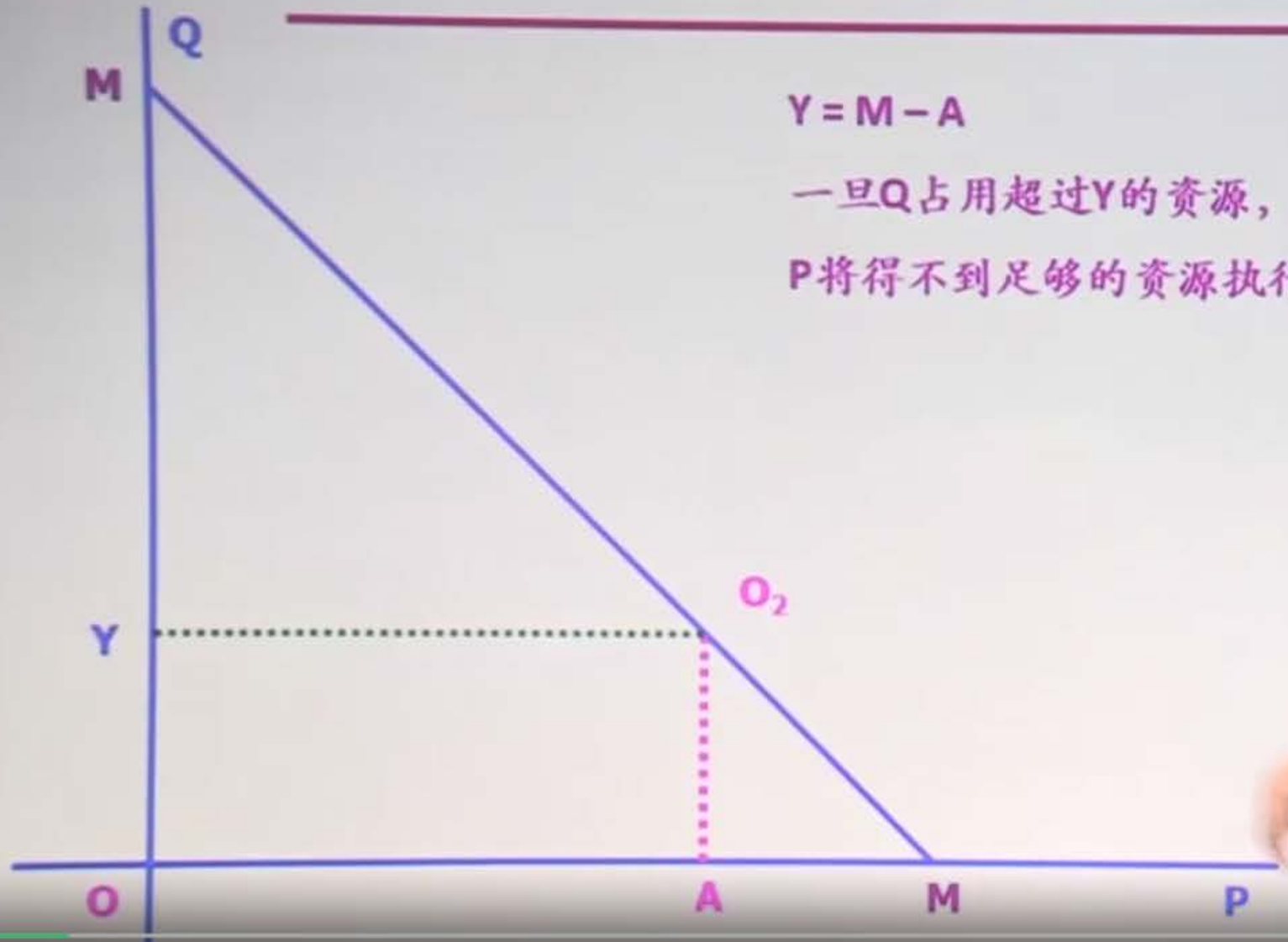
安全状态、不安全状态

死锁避免

下面我们介绍解决死锁问题的第二类方案：死锁避免

死锁避免讨论

A为P总资源需求量

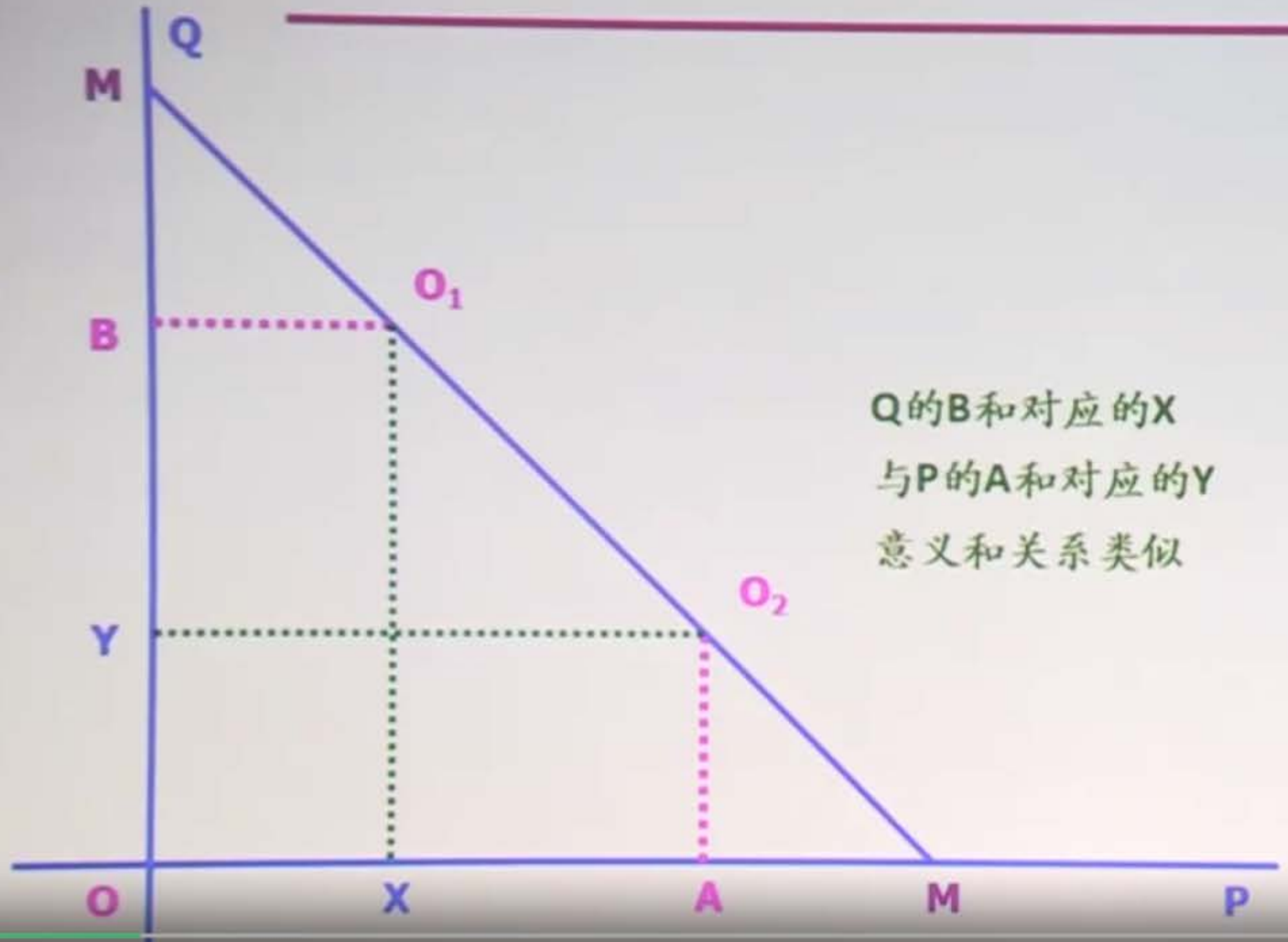


Y=M−A

一旦Q占用超过Y的资源，

P将得不到足够的资源执行

死锁避免讨论

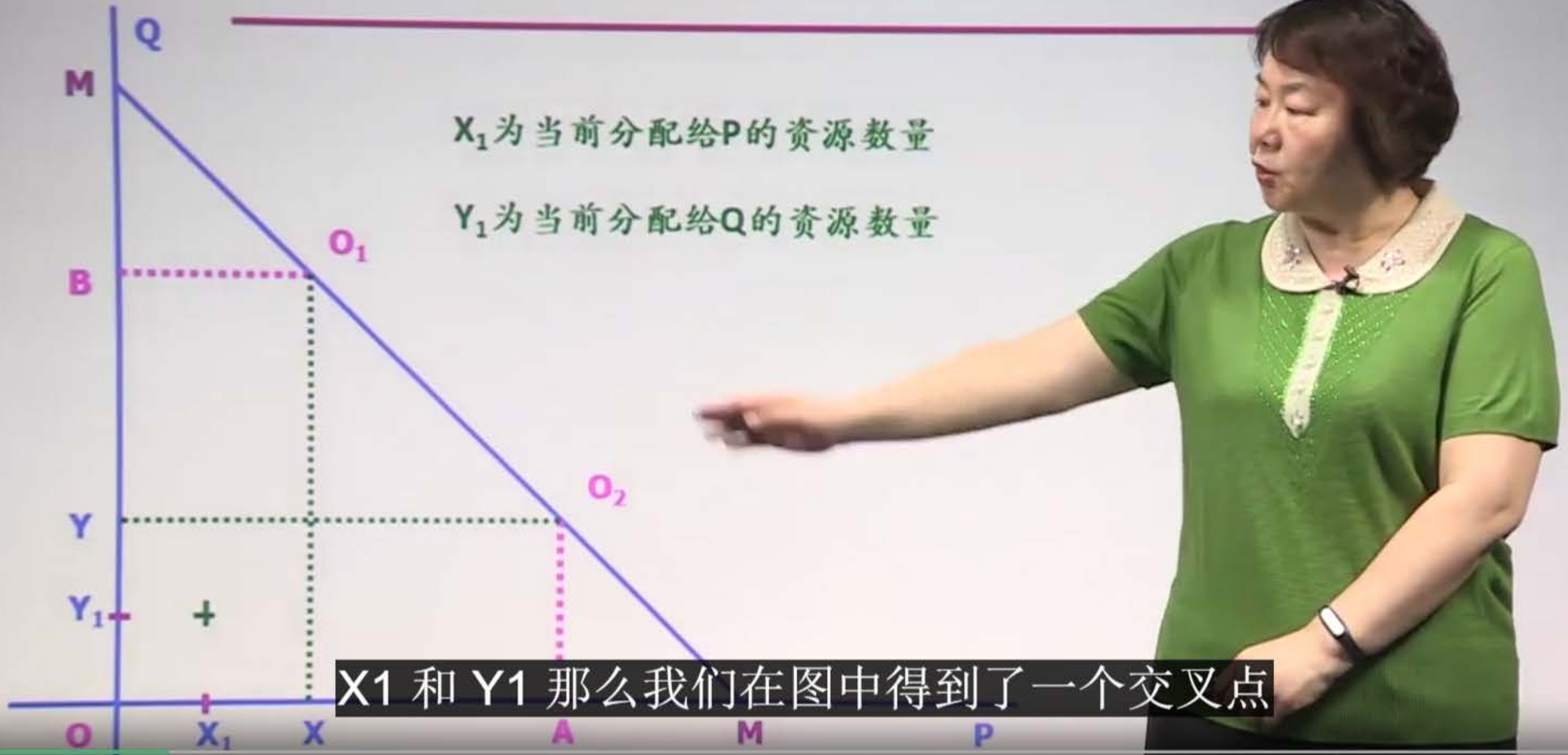


Q的B和对应的X

与P的A和对应的Y

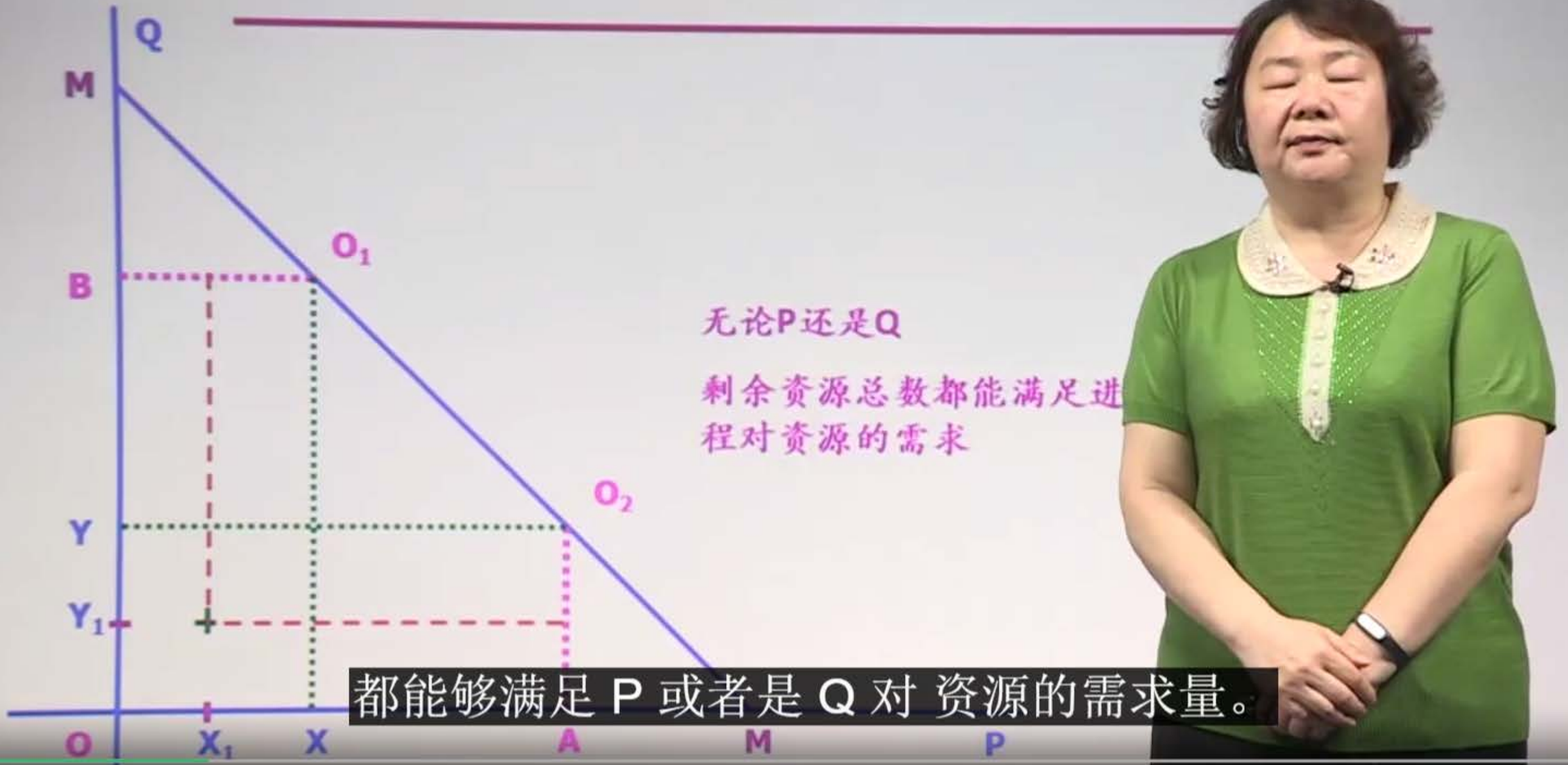
意义和关系类似

死锁避免讨论



X1 和 Y1 那么我们在图中得到了一个交叉点

死锁避免讨论

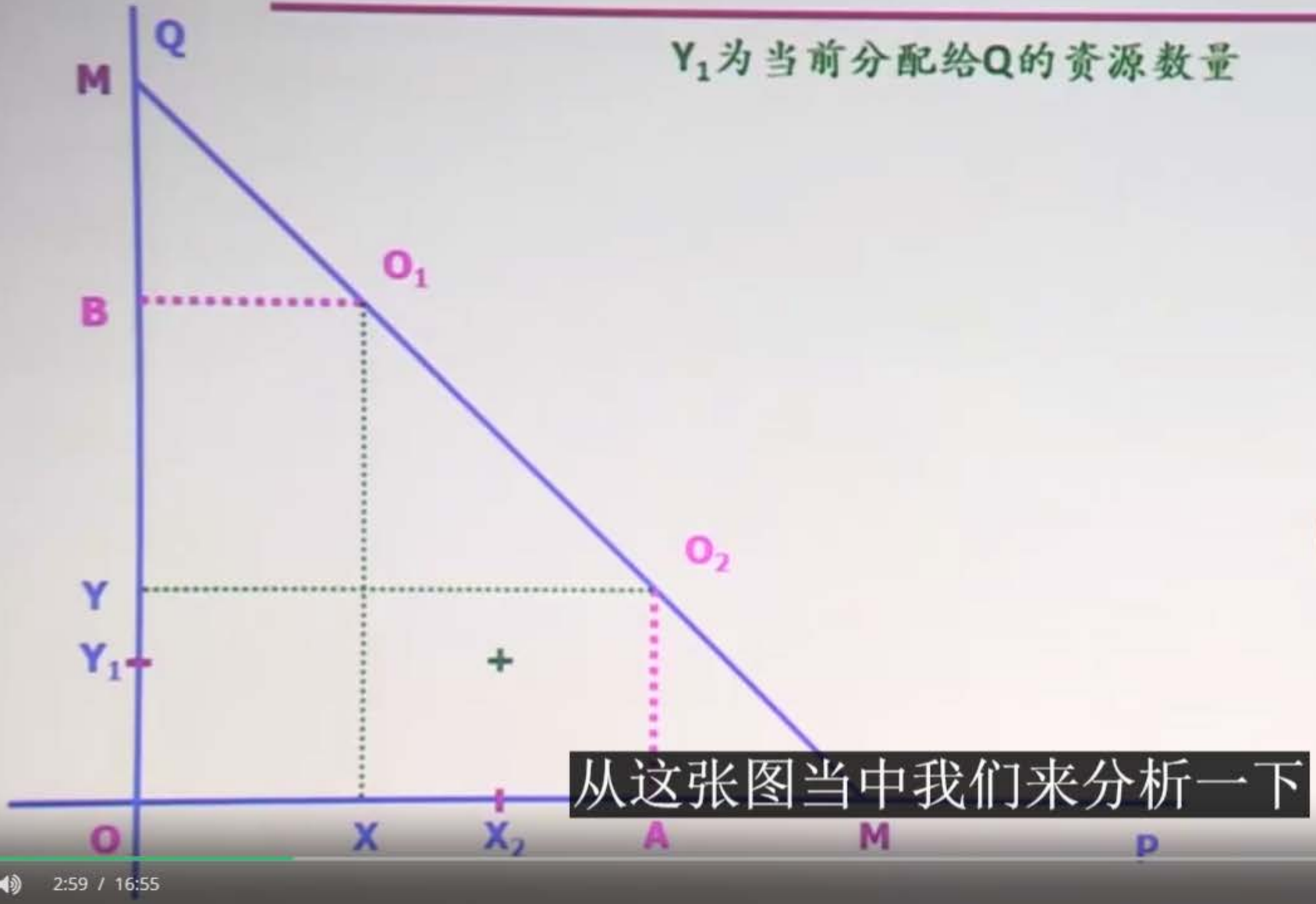


都能够满足 P 或者是 Q 对 资源的需求量。

死锁避免讨论

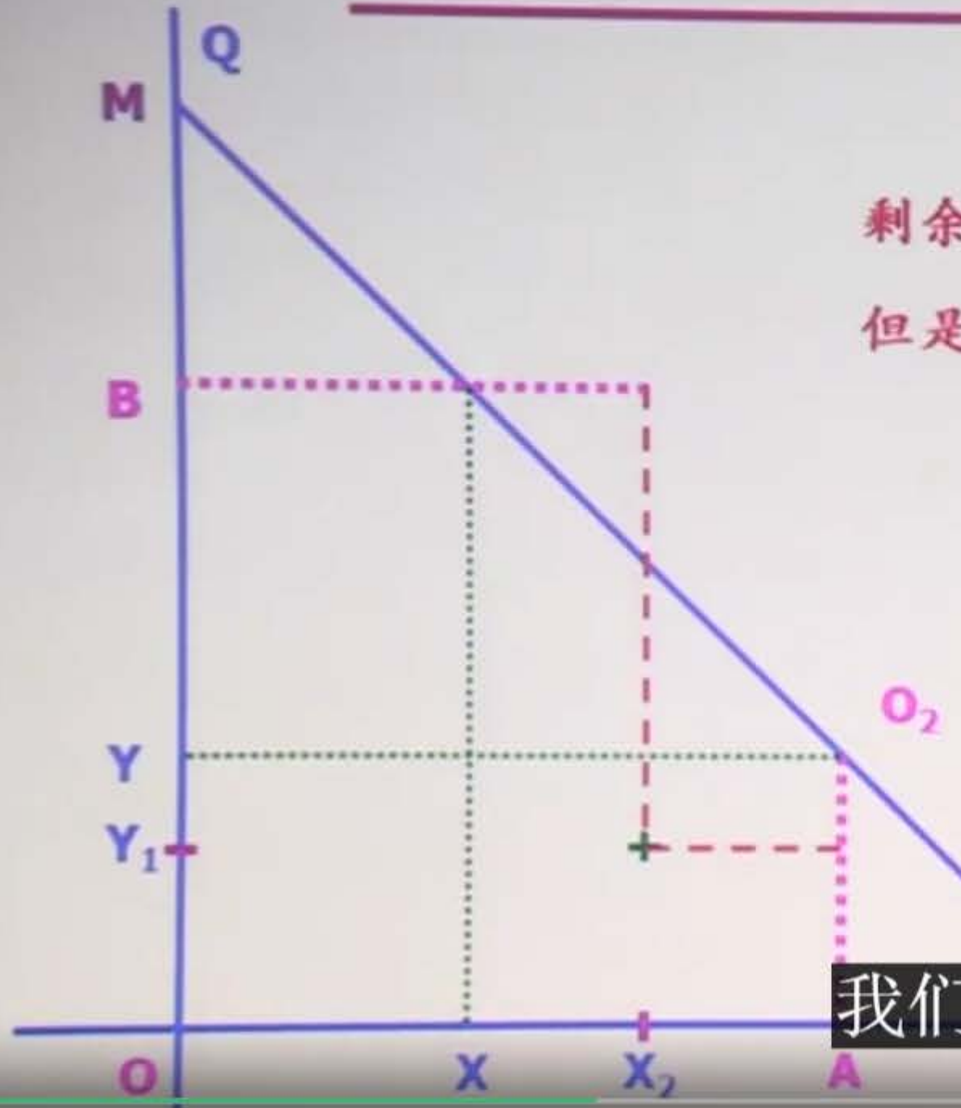
X_2为当前分配给P的资源数量

Y_1为当前分配给Q的资源数量



从这张图当中我们来分析一下

死锁避免讨论



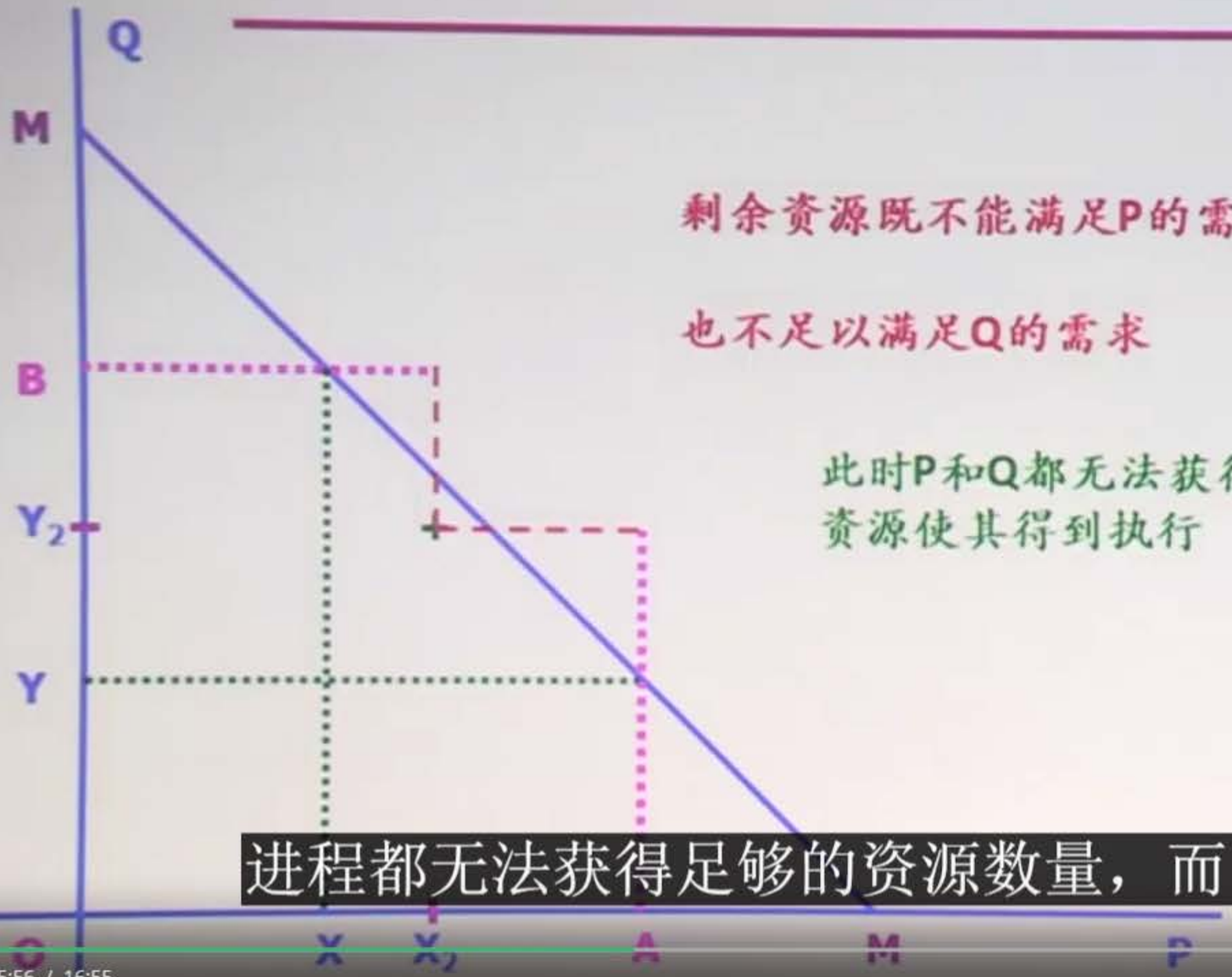
剩余资源不能满足Q的需求

但是剩余资源足够P的需求

P得到资源执行结束，
释放资源，从而使得有
足够资源满足Q的需求

我们接着进行分析 假设

死锁避免讨论



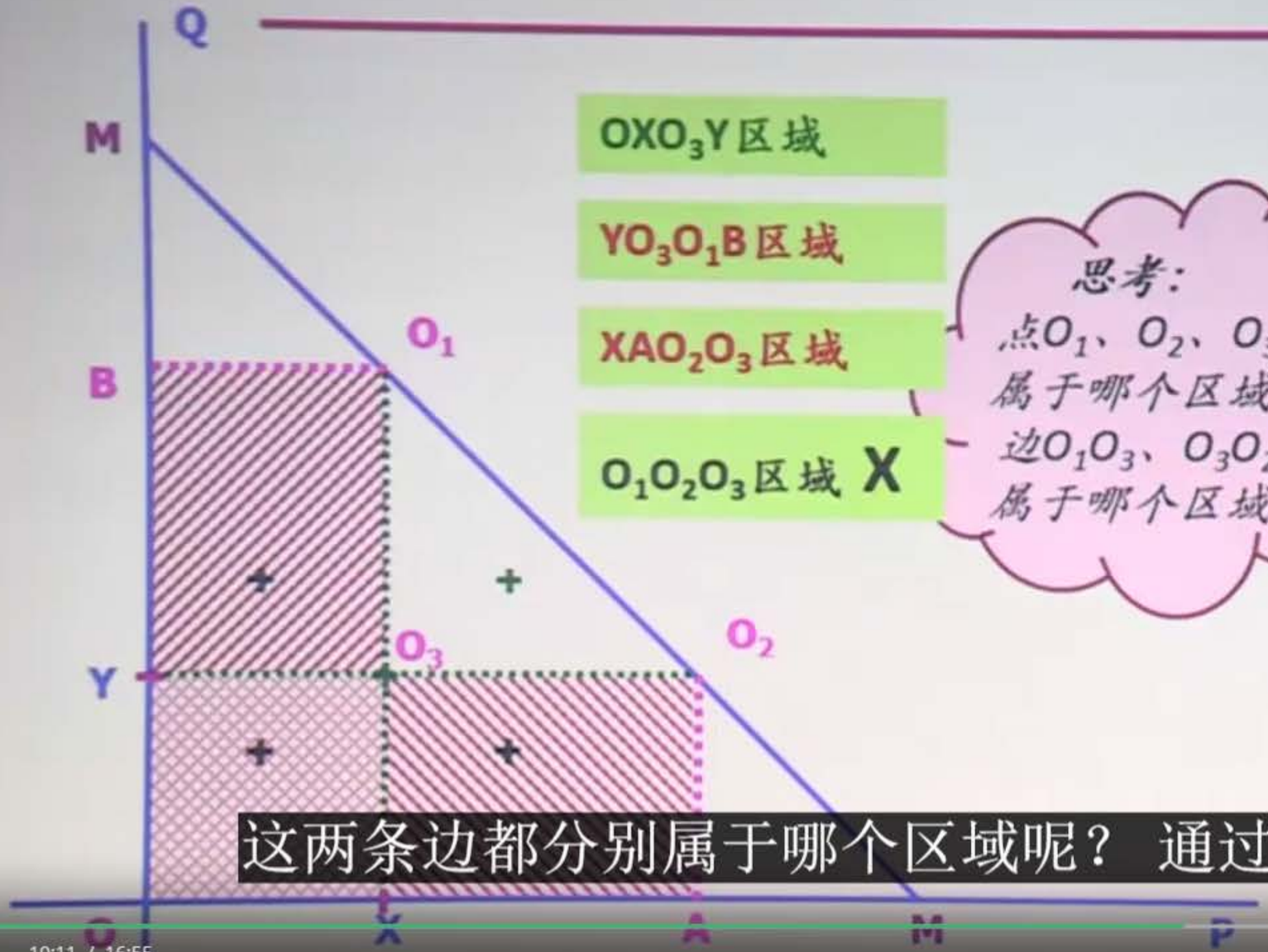
剩余资源既不能满足P的需求

也不足以满足Q的需求

此时P和Q都无法获得足够的
资源使其得到执行

进程都无法获得足够的资源数量，而 能够继续执行。

小结



OXO_3Y区域

YO_3O_1B区域

XAO_2O_3区域

$O_1O_2O_3$区域 X

思考：
点O_1、O_2、O_3
属于哪个区域？
边O_1O_3、O_3O_2
属于哪个区域？

这两条边都分别属于哪个区域呢？ 通过上面的这个例子

死锁避免定义

定义：

在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统发生死锁或可能发生死锁，则不予分配，否则予以分配

安全状态：

如果系统中存在一个由所有进程构成的安全序列P_1，…，P_n，则称系统处于安全状态

安全序列

一个进程序列{P_1，…，P_n}是安全的，如果对于每一个进程P_i($1 \leqslant i \leqslant n$)：

它以后还需要的资源量不超过系统当前剩余资源量与所有进程P_j（$j < i$）当前占有资源量之和

则称系统处于安全状态

安全状态一定没有死锁发生

我们的 结论就是，如果系统目前处于安全状态，就是一定没有死锁发生

安全状态与不安全状态

不安全状态：系统中不存在一个安全序列

不安全状态一定导致死锁

分配资源是不是进行

0:00 下面我们介绍解决死锁问题的第二类方案：死锁避免 我们先通过一个例子来讨论一下 死锁避免解决方案的设计思想 在这个例子当中有两个进程，进程 P 和进程 Q，系统当中一共有 M 个资源，假设 A 为 P 进程 对资源的需求的最大量，那么我们用 系统中资源总数减去 A 就得到了一个 Y 点。 我们可以看一下 Y 点 我们分析一下，对于进程 Q 一旦 Q 进程占有的资源数量 超过了 Y 这一点，那就意味着 P 进程得不到它所需要的最大的需求量 A 了 同样的道理 如果 B 是 Q 进程对资源的最大需求量 那么我们得到了一个 X 点 也就是说如果 P 进程 对资源的占有量超过了 X 点 也就意味着 Q 进程得不到它所需要的最大需求量 B 好，那么我们接着来讨论 通过刚才的分析，我们在 这张图中得到了 4 个区域 我们假设 X1 是进程 P 当前占有的资源量 Y1 是进程 Q 当前占有的资源量 那么 X1 和 Y1 那么我们在图中得到了一个交叉点 那么从这个交叉点出发 无论是 P 进程还是 Q 进程 系统中剩余资源的总数 都能够满足 P 或者是 Q 对 资源的需求量。 我们又假设 X2 是当前进程分配给 P 进程的 资源数量。 那么 Y1 呢是当前 系统分配给进程 Q 的资源数量 那么我们在这里又得到了另外一个交叉点 从这张图当中我们来分析一下 如果在这个交叉点 那么进程 Q 它 还需要的资源量，也就是说它要得到最多的 B 这么多个资源 它现在得到了 Y1 这么多个资源，但是系统中 剩余的资源量已经不能够满足 进程 Q 的剩余的需求了。 但是呢 在这一点剩余的资源可以满足 P 的需求 因此，我们分析得出当 P 得到了它所需要资源，然后 执行结束，释放资源，从而使得 Q 进程也能够满足它的资源需求 因此我们可以看到，如果 这个交叉点进入了这样一个区域 那么系统当中对资源的分配就是有条件的了 那么在这里头对 P 进程可以无条件地分配 但是对于 Q 进程它最多不能超过 Y 这么多个资源 我们接着进行分析 假设 X2 是系统当前分配给 P 进程的资源数量 Y2 呢是当前 Q 进程占有的资源数量 那么它们的交叉点就落入到这样一个区域，一个三角区域 从这一点出发，我们看到 无论是 P 进程还是 Q 进程 系统中剩余的资源数量 既不能满足 P 进程对资源的最大需求量 也不能够满足 Q 进程对资源的最大需求量 因为我们可以看到如果 P 进程 想得到 A 这么多个数量，那么这个时候已经 这条线已经穿过了这个这条边了 而这条边实际上是系统中总共的资源数量 因此对于 P 进程它拿不到剩余的资源数量 Q 进程也是如此，因此从 这个三角区域当中的某一点出发 P 进程和 Q 进程都无法获得足够的资源数量，而 能够继续执行。 小结一下 通过刚才的分析，得出结论 当进程执行过程中，提出资源申请时 操作系统应该根据当时系统所处的状态 或者把资源分配给这个进程之后 进入的一个新的状态来调整 资源分配

策略。 具体分析一下 我们看到左下角这个区域，也就是 QXO3Y 这个矩形区域 如果在这个区域里头 当任何一个进程，P 进程或者是 Q 进程 提出资源申请，操作系统都可以去满足 下面我们讨论另外两个区域 一个是左上角这样一个矩形区域 一个是右下角这样一个矩形区域 我们以右下角这个矩形区域为例进行讨论 假如说目前的 资源分配状态是在右下角这个矩形区域里头 如果 P 进程提出资源申请，操作系统 可以无条件地满足这个进程的对资源的 需求。 但是如果 Q 进程提出了资源申请 那么当它得到了资源的总数只要不超过 Y 那么操作系统就可以给它分配，如果它 提出了资源申请，操作系统分配给它以后，那么 Q 进程所占有的资源数量超过了 Y 这么多个资源数量 那么操作系统就不应该把资源分配给 Q 进程。 最重要的是操作系统 对资源的分配要控制，不能够出现 让进程 P 和 Q 对资源的 申请数量的交叉点进入这样一个三角区域 因为刚才我们已经看到，一旦进入了三角区域 那么系统剩余的资源数量既不能满足 P 进程也不能满足 Q 进程对资源的最大的 需求量。 那么就意味着会导致死锁的结果 我们来看，如果这个交叉点落到了这条线上，也就是 O1O3 这条线，但是不包括 O1,O3 这两个点 那么这个时候系统中的所有资源都被分配完了 而每个进程 P 和 Q 都没有 达到它对资源的最大需求量。 这时候 两个进程都在等资源，那么这个时候就出现了死锁现象 所以 操作系统要在进程申请资源的过程中 进行相应的控制，不能够让结果落入到这样一个三角区域 那么我们提出一个思考 刚才我们讨论的是这样几个区域 我们没有探讨点和边 那么我们提出一个思考，大家回去可以 自己想一下，那么点 O1，O2，O3 这个点它应该属于哪个区域 那么边 O1O3 O3O2 这两条边都分别属于哪个区域呢？ 通过上面的这个例子 我们可以给出死锁避免的 设计思想。 所谓死锁避免 就是指在系统的运行过程中 进程给提出各种各样的资源申请，对于 进程发出的每一个系统能够满足的资源申请 要进行动态检查 并且根据检查的结果 来决定是否分配，这就是 死锁避免的一个基本的设计思路。 如果分配以后 系统发生了死锁或者是可能发生死锁 那么就不予分配。否则呢就 给予分配。 那么我们这里头就 从刚才的分析过程中我们看到了有不同的区域 有些区域呢是随意，可以不管哪个进程提出申请都是可以分配的 有些区域呢是有限制条件的，比如说 可能只能给其中某些进程，而其它进程就不能给 有一些区域呢，如果一旦进入这些区域，那么 往前执行一定会导致死锁的发生 因此我们要对系统处于什么样的状态进行相应的 界定。 我们给出了一个安全状态这样一个概念 所

谓安全状态指的是如果系统当中存在着一个 由所有进程构成的一个安全序列 如果存在任何一个安全序列，我们就称 这个系统目前是处于安全状态 如果分配给 这个进程资源之后，系统处于安全状态，那么这个分配就 完成。 如果分配给这个进程这个 资源之后，系统就不是安全状态了，那么就不予分配 所以我们要根据系统是否处于安全状态，就是这种动态检查 来决定这次分配是否成立 那么什么是安全序列呢？我们给出安全序列的定义 在一个系统当中，一个进程序列 不是一般性的，我们把它表示成 P1 到 Pn 如果这个进程序列是安全的，指的是对于 序列当中的每一个进程 Pi 这个进程它以后还需要的资源数量 不超过当前系统剩余的资源数量以及 在这个进程，因为是序列，排在这个进程之前的 若干进程当前占有的资源数量之和 那么我们就说系统是处于安全状态的 我们简单地来描述一下这样一个场景 这个序列当中的第一个进程 P1 我们就可以把它描述成 P1 以后还需要的资源数量 不超过系统当前剩余的资源数量 因为它是第一个，所以不存在它之前的这些进程了，所以这句话 就到此结束，说 P1 所需要的资源数量 没有超过系统当前剩余的资源数量 那么很显然这个资源就可以分配给这个进程 那么让 P1 结束，执行结束，还回 之前的分配给它的进程，然后 剩下的集合里头，我们再看 P2 P2 呢就描述成 P2 以后还 需要的资源数量不超过系统当前剩余的资源数量 以及它前面的进程 P1 所占有的资源数量之和 那么 P2 也能够完成。 然后我们依次来推 就发现这个序列当中的每个进程都能够完成 这就是安全序列找到了，那么就意味着系统 不会出现死锁，是个安全状态。 我们的 结论就是，如果系统目前处于安全状态，就是一定没有死锁发生 有了安全状态，自然就要有不安全状态。 什么是不安全状态呢？ 也就是如果在系统当中，我找不到一个安全序列 那么这个时候系统就是不安全状态 不安全状态我们给出来它一定会导致死锁发生 当然了，目前它还没有进入死锁，因为系统中还有一些资源 但是继续往前走 不管系统采用什么样的资源分配策略 已经无力挽回走向死锁状态的这样一个目标了 因此我们来 刻画一下系统的各种状态 其中有一种是叫安全状态，安全状态呢是说我们找到了 一个，至少找到了一个安全序列。 当然也可能存在多个安全序列 还有一类状态叫不安全状态，而不安全状态 当中的一个特定的一个子集呢，我们就称之为死锁了 所以这是对死锁避免这个讨论的过程中 我们界定的，对系统状态的界定，并且根据不同的系统状态来决定 分配资源是不是进行