KM 算法是通过给每个顶点一个标号（叫做顶标）来把求最大权匹配的问题转化为求完备匹配的问题的。设顶点 X_i 的顶标为 $A[i]$，顶点 Y_i 的顶标为 $B[i]$，顶点 X_i 与 Y_j 之间的边权为 $w[i,j]$。在算法执行过程中的任一时刻，对于任一条边(i,j)，$A[i]+B[j]>=w[i,j]$始终 成立。KM 算法的正确性基于以下定理：

若由二分图中所有满足 $A[i]+B[j]=w[i,j]$的边(i,j)构成的子图（称做相等子图）有完备匹配，那么这个完备匹配就是二分图的最大权匹配。

这个定理是显然的。因为对于二分图的任意一个匹配，如果它包含于相等子图，那么它的边权和等于所有顶点的顶标和；如果它有的边不包含于相等子图，那么它的边权和小于所有顶点的顶标和。所以相等子图的完备匹配一定是二分图的最大权匹配。

初始时为了使 $A[i]+B[j]>=w[i,j]$恒成立，令 $A[i]$为所有与顶点 X_i 关联的边的最大权，$B[j]=0$。如果当前的相等子图没有完备匹配，就按下面的方法修改顶标以使扩大相等子图，直到相等子图具有完备匹配为止。

我们求当前相等子图的完备匹配失败了，是因为对于某个 X 顶点，我们找不到一条从它出发的交错路。这时我们获得了一棵交错树，它的叶子结点全部是 X 顶点。现在我们把交错树中 X 顶点的顶标全都减小某个值 d，Y 顶点的顶标全

都增加同一个值 d，那么我们会发现：
两端都在交错树中的边(i,j)，A[i]+B[j]的值没有变化。也就是说，它原来属于相等子图，现在仍属于相等子图。
两端都不在交错树中的边(i,j)，A[i]和 B[j]都没有变化。也就是说，它原来属于（或不属于）相等子图，现在仍属于（或不属于）相等子图。
X 端不在交错树中，Y 端在交错树中的边(i,j)，它的 A[i]+B[j]的值有所增大。它原来不属于相等子图，现在仍不属于相等子图。
X 端在交错树中，Y 端不在交错树中的边(i,j)，它的 A[i]+B[j]的值有所减小。也就说，它原来不属于相等子图，现在可能进入了相等子图，因而使相等子图得到了扩大。

现在的问题就是求 d 值了。为了使 A[i]+B[j]>=w[i,j]始终成立，且至少有一条边进入相等子图，d 应该等于 min{A[i]+B[j]-w[i,j]|Xi 在交错树中，Yi 不在交错树中}。

以上就是 KM 算法的基本思路。但是朴素的实现方法，时间复杂度为 O(n4)——需要找 O(n)次增广路，每次增广最多需要修改 O(n)次顶 标，每次修改顶标时由于要枚举边来求 d 值，复杂度为 O(n2)。实际上 KM 算法的复杂度是可以做到 O(n3)的。我们给每个 Y 顶点一个“松弛量”函数 slack，每次开始找增广路时初始化为无穷大。在寻找增广路的过程中，检查边(i,j)时，如果它不在相等子图中，则让 slack[j]变

成原值与 A [i]+B[j]-w[i,j]的较小值。这样，在修改顶标时，取所有不在交错树中的 Y 顶点的 slack 值中的最小值作为 d 值即可。但还要注意一点：修改 顶标后，要把所有的 slack 值都减去 d。