二分图（Bipartite Matching）指的是这样一种图，其所有顶点可以分成两个集合X和Y，其中X或Y中任意两个在同一集合中的点都不相连，所有的边关联在两个顶点中，恰好一个属于集合Ｘ，另一个属于集合Ｙ。给定一个二分图G，M为G边集的一个子集，如果M满足当中的任意两条边都不依附于同一个顶点，则称M是一个匹配。图中包含边数最多的匹配称为图的最大匹配。

二分图的最大匹配有两种求法，第一种是**最大流**；第二种就是我现在要讲的**匈牙利算法**。这个算法说白了就是最大流的算法，但是它跟据二分图匹配这个问题的特点，把最大流算法做了简化，提高了效率。

1. **增广路径的定义(也称增广轨或交错轨)：**

　　若P是图G中一条连通两个未匹配顶点的路径，并且属M的边和不属M的边(即已匹配和待匹配的边)在P上交替出现，则称P为相对于M的一条增广路径。

由增广路径的定义可以推出下述4个结论：

1. P的路径长度必定为奇数，第一条边和最后一条边都不属于M。
2. P上所有第奇数条边都不在M中，所有第偶数条边都出现在M中。
3. 经过取反操作可以得到一个更大的匹配M’。所谓“取反”即把P上所有第奇数条边(原不在M中)加入到M中，并把P中所有第偶数条边(原在M中)从M中删除，则新的匹配数就比原匹配数多了1个。（增广路顾名思义就是使匹配数增多的路径）
4. M为G的最大匹配当且仅当不存在相对于M的增广路径。

最大流算法的核心问题就是找增广路径（augment path）。匈牙利算法也不例外，它的基本模式就是：

**初始时最大匹配为空**

**while 找得到增广路径**

**do 把增广路径加入到最大匹配中去**

可见和最大流算法是一样的。但是这里的增广路径就有它一定的特殊性。（注：匈牙利算法虽然根本上是最大流算法，但是它不需要建网络模型，所以图中不再需要源点和汇点，仅仅是一个二分图。每条边也不需要有方向。）

算法的思路是不停的找增广路径, 并增加匹配的个数,增广路径顾名思义是指一条可以使匹配数变多的路径。在匹配问题中,增广路径的表现形式是一条"交错路径"，也就是说这条由图的边组成的路径，它的第一条边是目前还没有参与匹配的，第二条边参与了匹配，第三条边没有……最后一条边没有参与匹配，并且始点和终点还没有被选择过。这样交错进行，显然他有奇数条边。那么对于这样一条路径，我们可以将第一条边改为已匹配，第二条边改为未匹配……以此类推。也就是将所有的边进行"反色"，容易发现这样修改以后，匹配仍然是合法的，但是匹配数增加了一对。另外，单独的一条连接两个未匹配点的边显然也是交错路径。可以证明。当不能再找到增广路径时，就得到了一个最大匹配，这也就是匈牙利算法的思路。