# 二分图最大匹配的König定理-最小点集覆盖

假如我们已经通过匈牙利算法求出了最大匹配（假设它等于M），下面给出的方法可以告诉我们，选哪M个点可以覆盖所有的边。  
    匈牙利算法需要我们从右边的某个没有匹配的点，走出一条使得“一条没被匹配、一条已经匹配过，再下一条又没匹配这样交替地出现”的路（交错轨，增广路）。但是，现在我们已经找到了最大匹配，已经不存在这样的路了。换句话说，我们能寻找到很多可能的增广路，但最后都以找不到“终点是还没有匹配过的点”而失败。我们给所有这样的点打上记号：从右边的所有没有匹配过的点出发，按照增广路的“交替出现”的要求可以走到的所有点（最后走出的路径是很多条不完整的增广路）。那么这些点组成了最小覆盖点集：右边所有没有打上记号的点，加上左边已经有记号的点。看图，右图中展示了两条这样的路径，标记了一共6个点（用 “√”表示）。那么，用红色圈起来的三个点就是我们的最小覆盖点集。  
    首先，为什么这样得到的点集点的个数恰好有M个呢？答案很简单，因为每个点都是某个匹配边的其中一个端点。如果右边的哪个点是没有匹配过的，那么它早就当成起点被标记了；如果左边的哪个点是没有匹配过的，那就走不到它那里去（否则就找到了一条完整的增广路）。而一个匹配边又不可能左端点是标记了的，同时右端点是没标记的（不然的话右边的点就可以经过这条边到达了）。因此，最后我们圈起来的点与匹配边一一对应。  
    其次，为什么这样得到的点集可以覆盖所有的边呢？答案同样简单。不可能存在某一条边，它的左端点是没有标记的，而右端点是有标记的。原因如下：如果这条边不属于我们的匹配边，那么左端点就可以通过这条边到达（从而得到标记）；如果这条边属于我们的匹配边，那么右端点不可能是一条路径的起点，于是它的标记只能是从这条边的左端点过来的（想想匹配的定义），左端点就应该有标记。  
    最后，为什么这是最小的点覆盖集呢？这当然是最小的，不可能有比M还小的点覆盖集了，因为要覆盖这M条匹配边至少就需要M个点（再次回到匹配的定义）。

匈牙利算法：

有机会上，没机会创造机会也要上，确实没机会就算了

bool find(int x){ //find表示能不能找到或者腾出一个妹子分配给兄弟们

for (int i=1;i<=m;i++){ //扫描每个妹子

if (line[x][i]==true && used[i]==false)

//如果有暧昧并且还没有标记过(这里标记的意思是这次查找曾试图改变过该妹子的归属问题，但是没有成功，所以就不用瞎费工夫了）

{

used[i]=true;

if (girl[i]==0 || find(girl[i])) {

//名花无主或者能腾出个位置来，这里使用递归

girl[i]=x;

return true;

}

}

}

return false;

}

1. for (i=1;i<=n;i++)
2. {
3. memset(used,0,sizeof(used)); *//这个在每一步中清空*
4. if find(i) res+=1;
5. }