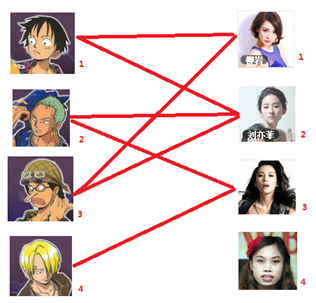
匈牙利算法（解决二分图最大匹配问题）

匈牙利算法是由匈牙利数学家Edmonds于1965年提出，因而得名。匈牙利算法是基于Hall定理中充分性证明的思想，它是部图匹配最常见的算法，该算法的核心就是寻找增广路径，它是一种用增广路径求二分图最大匹配的算法。

-------等等，看得头大？那么请看下面的版本：

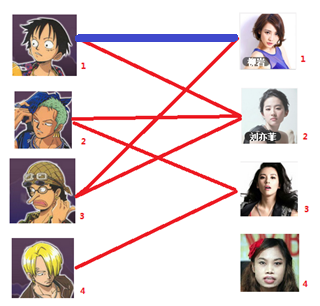
通过数代人的努力，你终于赶上了剩男剩女的大潮，假设你是一位光荣的新世纪媒人，在你的手上有N个剩男，M个剩女，每个人都可能对多名异性有好感（惊讶-\_-||暂时不考虑特殊的性取向），如果一对男女互有好感，那么你就可以把这一对撮合在一起，现在让我们无视掉所有的单相思（好忧伤的感觉快哭了），你拥有的大概就是下面这样一张关系图，每一条连线都表示互有好感。



本着救人一命，胜造七级浮屠的原则，你想要尽可能地撮合更多的情侣，匈牙利算法的工作模式会教你这样做：

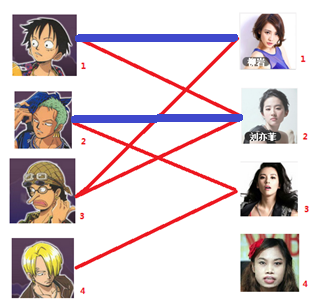
===============================================================================

**一：** 先试着给1号男生找妹子，发现第一个和他相连的1号女生还名花无主，got it，连上一条蓝线



===============================================================================

**二**：接着给2号男生找妹子，发现第一个和他相连的2号女生名花无主，got it

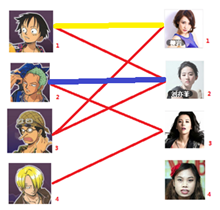


===============================================================================

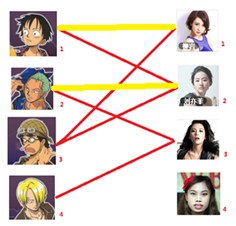
**三**：接下来是3号男生，很遗憾1号女生已经有主了，怎么办呢？

我们试着给之前1号女生匹配的男生（也就是1号男生）另外分配一个妹子。

(黄色表示这条边被临时拆掉)

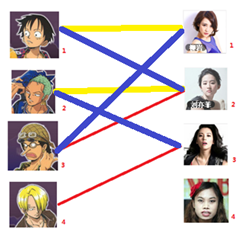
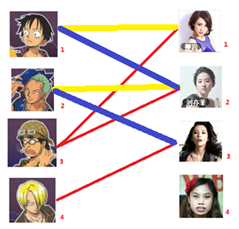
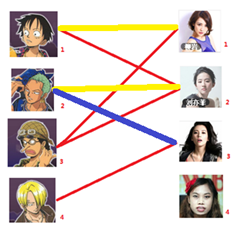


与1号男生相连的第二个女生是2号女生，但是2号女生也有主了，怎么办呢？我们再试着给2号女生的原配(发火发火)重新找个妹子(注意这个步骤和上面是一样的，这是一个递归的过程)

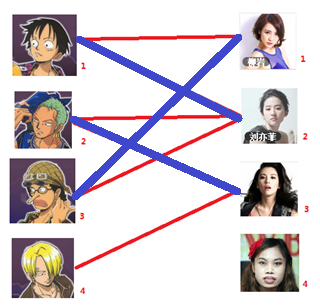


此时发现2号男生还能找到3号女生，那么之前的问题迎刃而解了，回溯回去

2号男生可以找3号妹子~~~                  1号男生可以找2号妹子了~~~                3号男生可以找1号妹子



所以第三步最后的结果就是：



===============================================================================

**四**： 接下来是4号男生，很遗憾，按照第三步的节奏我们没法给4号男生腾出来一个妹子，我们实在是无能为力了……香吉士同学走好。

===============================================================================

这就是匈牙利算法的流程，其中找妹子是个递归的过程，最最关键的字就是“腾”字

其原则大概是：有机会上，没机会创造机会也要上

【code】

**[cpp]** [view plain](http://blog.csdn.net/dark_scope/article/details/8880547/) [copy](http://blog.csdn.net/dark_scope/article/details/8880547/)

1. bool find(int x){
2. int i,j;
3. for (j=1;j<=m;j++){    //扫描每个妹子
4. if (line[x][j]==true && used[j]==false)
5. //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）
6. {
7. used[j]=1;
8. if (girl[j]==0 || find(girl[j])) {
9. //名花无主或者能腾出个位置来，这里使用递归
10. girl[j]=x;
11. return true;
12. }
13. }
14. }
15. return false;
16. }

在主程序我们这样做：每一步相当于我们上面描述的一二三四中的一步

**[cpp]** [view plain](http://blog.csdn.net/dark_scope/article/details/8880547/) [copy](http://blog.csdn.net/dark_scope/article/details/8880547/)

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) all+=1;
5. }