```
6 6
1 4
1 3
4 2
3 2
2 5
5 6
```

运行结果是：

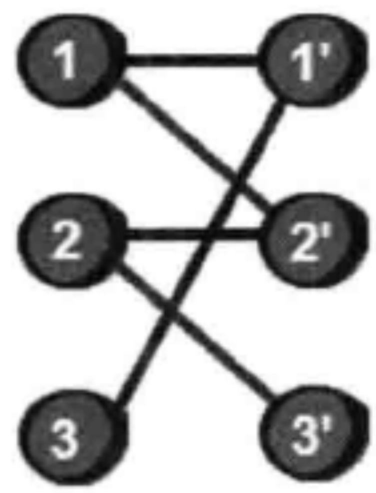
```
5-6
2-5
```

同割点的实现一样，这里也是用的邻接矩阵来存储图的，实际应用中需要改为使用邻接表来存储，否则这个算法就不是 $O(N+M)$ 了，而至少是 $O(N^2)$。如果这样的话，这个算法就没有意义了。因为你完全可以尝试依次删除每一条边，然后用深度优先搜索或者广度优先搜索去检查图是否依然连通。如果删除一条边后导致图不再连通，那么刚才删除的边就是割边。这种方法的时间复杂度是 $O(M(N+M))$。可见一个算法要选择合适的数据结构是非常重要的。

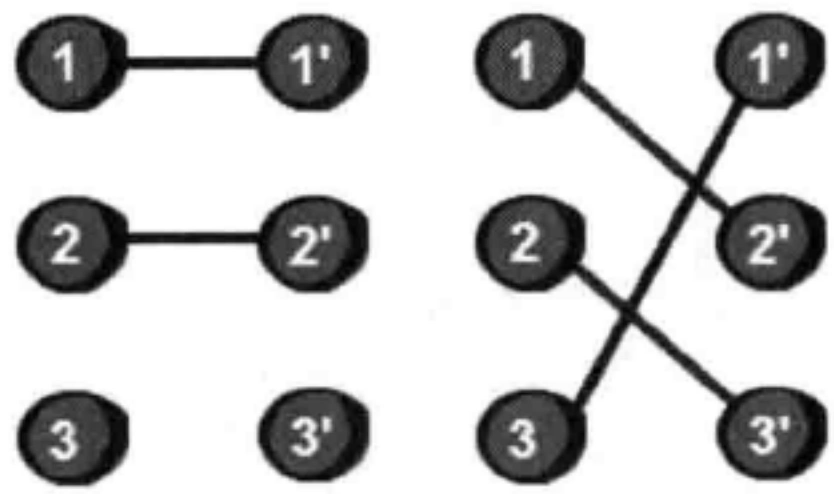
割点和割边算法也是由 Robert E. Tarjan 发明的，不得不说这位同学真是神犇啊！

第 5 节　我要做月老——二分图最大匹配

小哼今天和小伙伴们一起去游乐场玩，终于可以坐上梦寐以求的过山车了。过山车的每一排只有两个座位，为了安全起见，是每个女生必须与一个男生坐一排。但是，每个人都希望与自己认识的人坐在一起。举个例子吧，1 号女生与 1 号男生相互认识，因此 1 号女生和 1 号男生可以坐在一起。另外 1 号女生与 2 号男生也相互认识，因此他们也可以坐一起。像这样的关系还有 2 号女生认识 2 号和 3 号男生，3 号女生认识 1 号男生。请问如何安排座位才能让最多的人满意呢？这仅仅是一个例子。实际情况要复杂得多，因为小哼的小伙伴们实在是太多了。



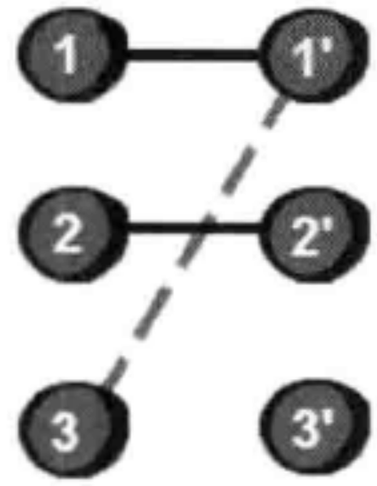
首先我们还是将问题模型化，如上图，左边的顶点是女生，右边的顶点是男生。如果顶点之间有边，则表示他们可以坐在一起。像这样特殊的图叫做二分图（注意二分图是无向图哦）。二分图的定义是：如果一个图的所有顶点可以被分为 X 和 Y 两个集合，并且所有边的两个顶点恰好一个属于集合 X，另一个属于集合 Y，即每个集合内的顶点没有边相连，那么此图就是二分图。对于上面的例子，我们很容易找到两种分配方案，如下。



很显然右边的分配方案更好。我们把一种分配方案叫做一种匹配。那么现在的问题就演变成求二分图的最大匹配（配对数最多）。求最大匹配最容易想到的方法是：找出全部匹配，然后输出配对数最多的。这种方法的时间复杂度是非常高的，那还有没有更好的方法呢？

我们可以这么想，首先从左边的第 1 号女生开始考虑。先让她与 1 号男生配对，配对成功后，紧接着考虑 2 号女生。2 号女生可以与 2 号男生配对，接下来继续考虑 3 号女生。此时我们发现 3 号女生只能和 1 号男生配对，可是 1 号男生已经配给 1 号女生了，怎么办？3

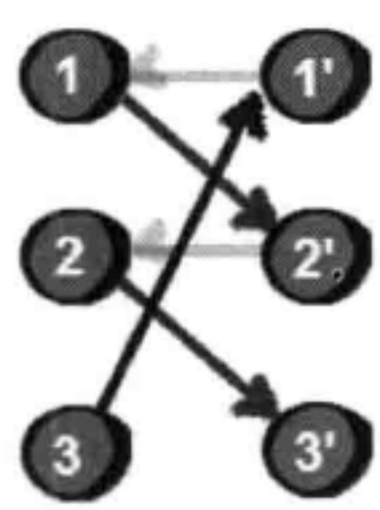
号女生是不是就此放弃了呢？可不能就这么放弃啊，放弃了就玩不了过山车了……



此时 3 号女生硬着头皮走到了 1 号男生面前，貌似 1 号男生已经看出了 3 号女生的来意，这个时候 1 号男生对 3 号女生说："我之前已经答应了与 1 号女生坐一起，你稍等一下，我让 1 号女生去问问看她能否与其他认识的男生坐一起，如果她找到了别的男生，那我就和你坐一起。"接下来，1 号女生便尝试去找别的男生啦。

此时 1 号女生来到了 2 号男生面前问："我可以和你坐在一起吗？"2 号男生说："我刚答应和 2 号女生坐一起，你稍等一下，我让 2 号女生去问问看她能否与其他认识的男生坐一起，如果她找到了别的男生，那我就和你坐一起。"接下来，2 号女生又去尝试找别的男生啦。

此时，2 号女生来到了 3 号男生面前问："我可以和你坐一起吗？"3 号男生说："我正空着呢，当然可以啦！"此时 2 号女生回过头对 2 号男生说："我和别的人坐在一起啦。"然后 2 号男生对 1 号女生说："现在我可以和你坐在一起啦。"接着，1 号女生又对 1 号男生说："我找到别的男生啦。"最后 1 号男生回复了 3 号女生："我现在可以和你坐在一起啦。"



真是波折啊～～是不是有点连锁反应的感觉。最终通过这种连锁反应，配对数从原来的 2 对变成了 3 对，增加了 1 对。刚才的过程有个专业名词叫做增广路，不难发现如果找到一条增广路，那么配对数将会加 1。增广路的本质就是一条路径的起点和终点都是未被配对的点。

既然增广路的作用是"改进"匹配方案（增加配对数），如果我们已经找到一种匹配方案，如何确定当前这个匹配方案已经是最大匹配了呢？如果在当前匹配方案下再也找不到增广路，那么当前匹配就是最大匹配了，算法如下。

1. 首先从任意一个未被配对的点 u 开始，从点 u 的边中任意选一条边（假设这条边是 $u \rightarrow v$）开始配对。如果此时点 v 还没有被配对，则配对成功，此时便找到了一条增广路（只不过这条增广路比较简单）。如果此时点 v 已经被配对了，那就要尝试进行“连锁反应”。如果尝试成功了，则找到一条增广路，此时需要更新原来的配对关系。这里要用一个数组 match 来记录配对关系，比如点 v 与点 u 配对了，就记作 match[v]=u 和 match[u]=v。配对成功后，记得要将配对数加 1。配对的过程我们可以通过深度优先搜索来实现，当然广度优先搜索也可以。
2. 如果刚才所选的边配对失败，要从点 u 的边中再重新选一条边，进行尝试。直到点 u 配对成功，或者尝试过点 u 所有的边为止。
3. 接下来继续对剩下没有被配对的点一一进行配对，直到所有的点都尝试完毕，找不到新的增广路为止。
4. 输出配对数。

这种方法的正确性留给你来证明，嘿嘿，并不难证，思考一下吧，代码如下。

```
#include <stdio.h>
int e[101][101];
int match[101];
int book[101];
int n,m;
int dfs(int u)
{
    int i;
    for (i=1; i<=n; i++)
    {
        if (book[i] == 0 && e[u][i] == 1)
        {
            book[i] = 1;    //标记点i已访问过
            //如果点i未被配对或者找到了新的配对
            if ( match[i] == 0  || dfs(match[i]) )
            {
                //更新配对关系
                match[i] = u;
                match[u] = i ;
                return 1;
            }
        }
```

```
    }
    return 0;
}

int main()
{
    int i, j, t1, t2, sum=0;

    scanf("%d %d",&n,&m);//n个点m条边

    for (i=1; i<=m; i++)//读入边
    {
        scanf("%d%d", &t1, &t2);
        e[t1][t2]=1;
        e[t2][t1]=1; //这里是无向图
    }

    for(i=1;i<=n;i++)  match[i]=0;

    for (i=1; i<=n; i++)
    {
        for(j=1;j<=n;j++)  book[j]=0;  //清空上次搜索时的标记
        if (dfs(i))  sum++;//寻找增广路，如果找到，配对数加1
    }

    printf("%d", sum);

    getchar();getchar();
    return 0;
}
```

可以输入以下数据进行验证。注：1、2、3 为女生，4、5、6 为男生。

```
6 5
1 4
1 5
2 5
2 6
3 4
```

运行结果是：

```
3
```

如果二分图有 n 个点，那么最多找到 $n/2$ 条增广路径。如果图中共有 m 条边，那么每找一条增广路径最多把所有边遍历一遍，所花时间是 m。所以总的时间复杂度是 $O(NM)$。

二分图在任务调度、工作安排等方面有较多应用。那么如何判断一个图是二分图呢？上面的例子是比较明显的。有些时候则很难看出一个图是二分图，因此首先需要判断这个图是不是二分图。判断一个图是否为二分图也非常简单，首先将任意一个顶点着红色，然后将其相邻的顶点着蓝色，如果按照这样的着色方法可以将全部顶点着色的话，并且相邻的顶点着色不同，那么该图就是二分图。此处我就不实现了，留给你来尝试吧。

该算法称为匈牙利算法，由 Jack Edmonds 给出。目前最快的二分图最大匹配算法是由 John E. Hopcroft（怎么又是这个人！）提出的。有兴趣的同学可以去看看他的论文“An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs，SIAM *Journal on Computing*，1973”。下面即将进入本书的最后一章：还能更好吗。