算法————艺术

二分图匹配剖析

很多人说，算法是一种艺术。但是对于初学者的我，对算法认识不是很深刻，但偶尔也能感受到他强大的魅力与活力。这让我追求算法的脚步不能停止。下面我通过分析匈牙利算法以及常用建图方式，与大家一起欣赏算法的美。

匈牙利算法

匈牙利算法是用来解决最大二分图匹配问题的，所谓二分图即 “一组点集可以分为两部分，且每部分内各点互不相连，两部分的点之间可以有边”。所谓最大二分图匹配即 ”对于二分图的所有边，寻找一个子集，这个子集满足两个条件，

1：任意两条边都不依赖于同一个点。

2：让这个子集里的边在满足条件一的情况下尽量多。

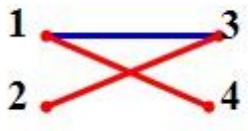
首先可以想到的是，我们可以通过搜索，找出所有的这样的满足上面条件的边集，然后从所有的边集中选出边数最多的那个集合，但是我们可以感觉到这个算法的时间复杂度是边数的指数级函数，因此我们有必要寻找更加高效的方法。目前比较有效的方法有匈牙利算法和通过添加汇点和源点的网络流算法，对于点的个数都在 200 到 300 之间的数据，我们是采取匈牙利算法的，因为匈牙利算法实现起来要比网络流简单些。下面具体说说匈牙利算法：

介绍匈牙利之前，先说说“**增广轨**”。

定义：

若 P 是图 G 中一条连通两个未匹配顶点的路径，并且属最大匹配边集 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 的一条增广轨

定义总是抽象的下面通过图来理解它。



图中的线段（2->3, 3->1, 1->4）便是上面所说的 p 路径，我们假定边（1,3）是以匹配的边，（2,3）（1,4）是未匹配的边，则边（4,1）边（1,3）和 边（3,2） 在路径 p 上交替的出现啦，那么 p 就是相对于 M 的一条增广轨，这样我们就可以用边 1,4 和 边 2,3 来替换边 1,3 那么以匹配的边集数量就可以加 1,。

匈牙利算法就是同过不断的寻找增广轨实现的。很明显如果二分图的两部分点分别为 n 和 m，那么最大匹配的数目应该小于等于 MIN（n,m）；因此我们可以枚举任第一部分（的二部分也可以）里的每一个点，我们从每个点出发寻找增广轨，最后吧第一部分的点找完以后，就找到了最大匹配的数目，当然我们也可以通过记录找出这些边。

下面给出关于二分图最大匹配的两个定理

1:最大匹配数 + 最大独立集 = n + m

2:二分图的最小覆盖数 = 最大匹配数

3:最小路径覆盖 = 最大独立集

最大独立集是指求一个二分图中最大的一个点集，该点集内的点互不相连。

最小顶点覆盖是指 在二分图中，用最少的点，让所有的边至少和一个点有关联。

最小路径覆盖是指一个不含圈的有向图 G 中，G 的一个路径覆盖是一个其结点不相交的路径集合 P，图中的每一个结点仅包含于 P 中的某一条路径。路径可以从任意结点开始和结束，且长度也为任意值，包括 0

下面给出二分图最大匹配的伪代码：

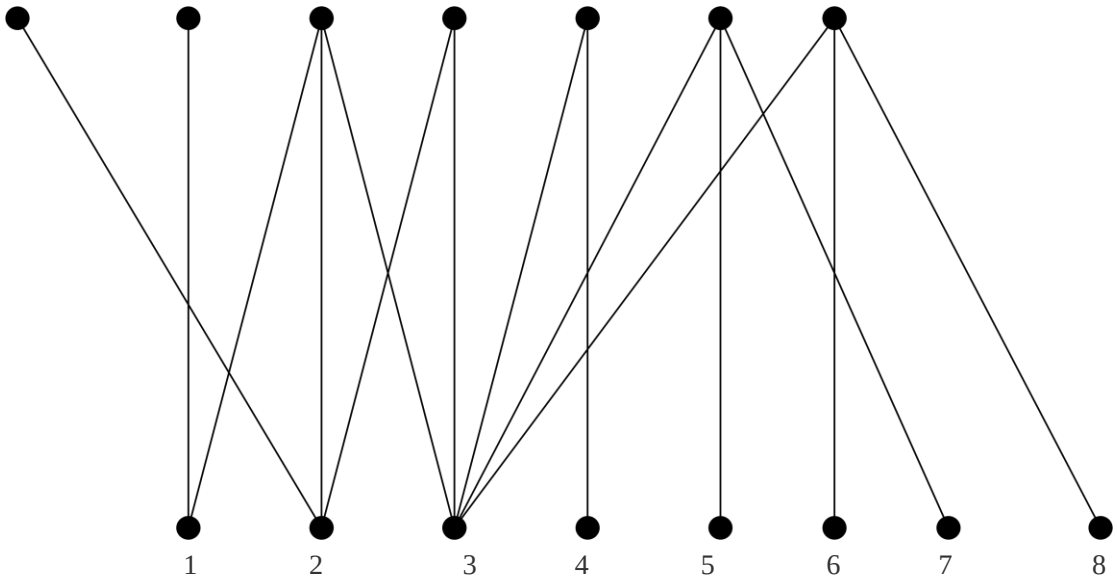
```
/*
    use[i]   数组时标记第二部分点中的第 j 个点是否使用过。
    match[i] 与第二部分中的点 i 匹配的点是   match[i]
*/
Int GetAugmentPath ( number )   //通过 number 这个点寻找增广轨，找到返回 1 找不到返回 0；
    P  ←  related[number] . next; //  与  number  相连的点；
    While（p   !=   NULL）{
        If   not used[p] then     // p  点没有被使用过
            If match[p] == 0 or GetAugmentPath(match[p]) // p 点没有和另一部分的点配对
                                  //  或  和 p 配对的那个点可以找到其它点和他配对（找到增广轨  ）
                Return 1;
    p  ←  p . next //            与  number  相连的下几个点
    }
    Return 0;
} end GetAugmentPath;
```

我们只需要在主程序中对某一部分点集的每个点进行增广轨的寻找；

```
Int main{
    ans ← 0
    For   I = 1   →   n   //   枚举第一部分的 n 个点；
        For   j = 1 →   m
            use[j]   ←   false //   把第二部分的 m 个点表示为没有使用过。
        if   GetAugmentPath (i) then
            ans ← ans + 1       //  找到增广轨就给边数加一；
}
```

程序非常好写，也非常好懂。

1　　2　　3　　4　　5　　6　　7



每次我们从上面的第 i 个点出发尽量去找一个能唯一和它匹配的点 p，策略有两种，一是直接在下面的点中找到一个点 p，他没有和上面的点匹配过（即 match[p] =0）。二是当 p 和上面的某个点匹配过，即（match[p]） 那么我们就从 match[p]出发，去给他找下面另外的点和他匹配，把 p 点留给点 i。这样我们不就多找到了一条？

然而 对于匈牙利算法来说，难点并不在与程序本身，而是在如何把实际问题转化为最大二分图匹配的模型，然后再利用匈牙利算法解决。 下面我们说几种常见的**建图模型**。

常见建图模型

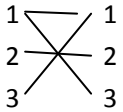
法一　行列匹配法

1	**0**	**1**
0	**1**	**0**
1	**0**	**0**

上图是一个 3 * 3 的矩阵，方格内的 1 表示这个地方有敌人，0 表示没有敌人，现在我们有很多箭，每根箭可以杀死一行或者一列的敌人，问题是，我们要杀死所有的敌人至要用到几根箭？

初看似乎和最大二分图没有什么相关联的，然而如果我们转换一下视角，这样思考：我们要杀死某个敌人，只要让他所在的位置有箭经过就行。也就是所有的位置都被箭覆盖就行，对就是覆盖，就是顶点的最小覆盖，既然是顶点的最小覆盖，而且我们要杀的是敌人，那么我们的点就应该是敌人的位子，即（行列）对于上面那个图我么可以建立下面这个模型

有人在的坐标是（1,1　1,3　 2,2　 3,1） 我们就用这几个点的横纵坐标建图



（每个点的横坐标是第一部分，纵坐标是另一部分，被边相连的两个数字表示一个点）

上面我们说过，一个二分图的最小顶点覆盖就是要找到最少的边把所有的顶点覆盖，正好符合这个题的要求，上面还给出了一个性质，即二分图的最小顶点覆盖是等于二分图的最大匹

配。所以我们只需要对上面的那个二分图就最大匹配就行，这样把原本的问题转变的很简单了。

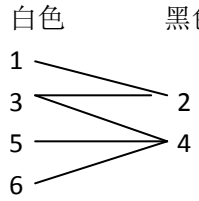
法二　黑白染色法

1	0	1
1	1	1
0	0	1

又是一个图，要求是把方格里的所有的 1 改为零，一次最多只能修改相邻的两个，为最少需要修改几次？ 有是一个求最值得问题，但是似乎用于求最值的算法（贪心，动态规划……）都派不上用场，既然在这里提出，那么他肯定能用二分图最大匹配解决，关键是如何建图？既然是每次只能拿相邻的两个，是两个，正好我们匹配的时候也是找两个进行匹配，这是否就是这个题和最大二分图匹配相联系的地方呢？ 对 就是这里。但是每个点能和他四周的四个点匹配，那么我们怎么把所有的点分成来那个部分呢？ 对 就是要把第 i 个点放到第一部分，第 I 个点周围的四个点放到第二部分，再把这四个点周围的 16 点放到第 1 部分有了这样的思想，我们只需对原图做这样的改动：黑白染色使四周和中间的颜色不同。

白 1	**黑**	白 5
黑 2	白 3	**黑 4**
白	**黑**	白 6

图中黑白的意思是就是把点分类，图里的 1,2,3,4,5,6 表示的就是上面那个 0,1 图的 1 的个数然后建图，把相邻的点相连，比如说 1 和 2　 2 和 3 。。。。。。。



然后要把所有一改为零，也就是要对每个点都操作，每个点都要有，那不就是最小顶点覆盖吗？对，这个问题有解决了。

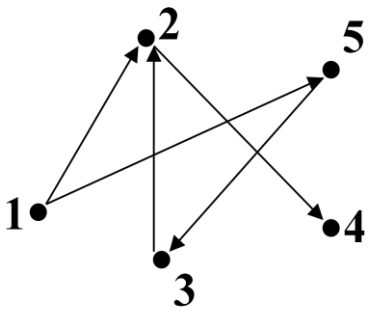
法三　反建法

问题背景：一个极度封建的老师要带同学们出去玩，但是他怕在途中同学之间发生恋情
老师研究了一下发现，满足下面几种条件的两个同学之间发生恋情的可能性很小
1》身高差 > 40　　2>性别相同
3》爱好不同的音乐　　4>爱好同类型的运动
显然如果我们用满足上面条件的同学之间建边那么最后建立起来的就不是二分图了。稍微观察一下，男生之间我们是随便带的，女生也是，因为他们彼此性别相同。因此我们就可以把男女分为两部分，那么男女之间如何建边？如果我们把男女满足不发生恋情的连起来，那么求出来的最大匹配没有代表性，不能得到我们想要的结果。因此我们用反建法，把男女中可能发生恋情的建立边。也就是说把身高差<=40 或 爱好相同音乐 或 爱好不同类型运动的男

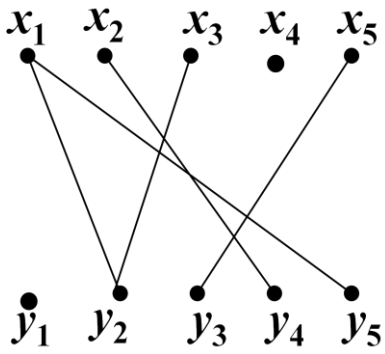
女同学之间用边连起来。然后求一个最大独立集，最大独立集的原则不就是找到一个点集，使得集合内的点互不相连且点尽量多吗？我们把可能发生恋情的男女相连，那么最大独立集不就是我们要找的不可能发生恋情的人的集合吗？ 那么， 这个问题解决了！

法四　拆点法

拆点法是用于解决最小路径覆盖问题的，给出一个图



要找到几条路径，可以把所有的点经过，并且路径之间不可以交叉。我们的做法是把点拆成两部分（点 1 拆为 x1,y1. 点 2 拆为 x2,y2……）

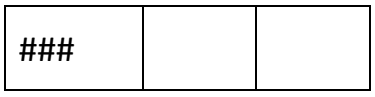


如果我们对这个图求二分图的最大匹配，你会发现每个匹配对应着一个路径覆盖，因此，此二分图的最大匹配即：原图中的最小路径覆盖上的边的个数（路径是由 0 条，1 条或多条边组成的）。那么原图的最小路径覆盖数 = 原图顶点数 – 最小路径上的边数　也就是 原图的最小路径覆盖数 = 原图顶点数 – 二分图最大匹配数。

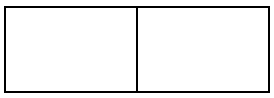
法五　一行变多行，一列变多列

		###	
###			
		###	###
	###		

上面是一个 4*4 的方格，方格内的###表示墙，我们要在表格内没有墙的地方建立碉堡，而且要保证任何两个碉堡之间互相不能攻击，问最多能建多少个碉堡？是否感觉像第一个题呢？如果我们向第一个题那样建图，那么最后求出来的最大匹配也就是行和列的匹配。而且这个匹配满足了所有匹配都是不同行不同列（匹配本身的性质就是每个点至多属于匹配中的某个边）。但是这样的建图的话，我们墙怎么处理？　有墙的地方就相当于把这一行和这一列分成了两行，两列。例如



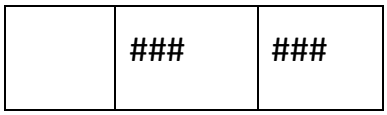
等价于



还是一行

等价于

和



一行变成了两行

对就是这么分的。

1,1	1,2	###	1,4
###	2,2	2,3	2,4
3,1	3,2	###	###
4,1	###	4,3	4,4

原图

1,1	1,2	###	2,4
###	3,2	3,3	3,4
4,5	4,2	###	###
5,5	###	6,6	6,7

（因为有了墙所以第一行变为两行）

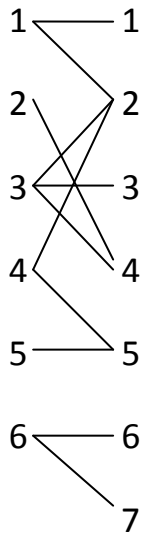
（因为上面有了二行固只能从第三行开始）

（因为第一列有了墙，固列数增加为 5）

（第 3,4 列有了墙，固列数增加到了 6 和 7）

一行变多行，一列变多列后的图

然后我们按照这个编号见图即可



对这个图求二分图最大匹配即可。这个问题也解决了！

一个应用于求二分图匹配的算法—匈牙利算法，巧妙的应用找增广轨的方式，使得求二分图最大匹配即变得高效，而且也变得容易理解，这就有了艺术的感觉，然而，一个应用于二分图的操作，却可以把其他看似和二分图没关系的事情，通过巧妙的，艺术性的手笔转化为二分图问题，这是多么的美妙。算法是很神奇的东西，总是让人摸不着头脑，当你认为自己对某种算法很熟悉的时候，你总会立即发现其实你只是对他有了一点点认识，这就是算法。