图论的基本思想及方法

湖南省长沙市长郡中学 任恺

【摘要】

文章着眼于图论基本思想及方法的讨论，不涉及高深的图论算法。

文章主要从两方面阐述图论的基本思想：一是合理选择图论模型；二是如何深入挖掘问题本质，充分利用模型的特性。同时还归纳了一些解决问题的普适性方法。

【关键字】

基本思想、图论模型、问题本质、定义法、分析法、综合法

【正文】

一、引论

图是用点和边来描述事物和事物之间的关系，是对实际问题的一种抽象。之所以用图来解决问题，是因为图能够把纷杂的信息变得有序、直观、清晰。因而图论中最基本的思想就是搭建合适的模型，深刻挖掘问题的本质，分析和利用图论模型各种性质，从而到达解决问题的目的。下面着重从模型的选择和发掘利用图的性质来阐述图的基本思想和运用方法。

二、合理选择图论模型

在解决一道实际问题时，往往先将实际问题抽象成一个数学模型，然后在模型上寻找合适的解决方法，最后再将解决方法还原到实际问题本身。而图论模型

就是一种特殊的数学模型。在搭建图论模型时，是通过图中的点和边来体现原问题的特点。搭建的模型务必要真实的、贴切的和透彻的反映出原问题的本质，同时也要做到力求简练、清晰。图论问题往往关系错综复杂，变化多端，因此搭建一个合适的模型实非易事。在选择图论模型时，应该深入分析实际问题的特点，大胆的猜想和验证。下面通过一个具体实例，来揭示选择合适图论模型的重要性和一些方法：

例一：滑雪者（Poland Olympiad of Informatics 2002 Stage III：Skiers）

题目大意：给出一个平面图，图中有 n（$2 \le n \le 5000$）个点，m 条有向边。每个点都有**不同的横坐标和纵坐标**，有一个最高点 v_h 和一个最低点 v_l。每条有向边连接着两个不同的点，方向是从较高点连到较低点。对于图中任意一点 u，都至少存在一条 v_h 到 u 的路径和一条 u 到 v_l 的路径。任务：图中由每个点发出的边都已经**按照结束点的位置从左到右给出**。要求你用若干条从 v_h 到 v_l 的路径覆盖图中所有的边，并且使路径数最少。所谓覆盖，就是指每条边至少在一条路径中出现。选取的路径之间可以有相同的边。（题目中和下面分析中所说的路径都是有向路径，若 a 到 b 存在一条路径，并不表示 b 到 a 一定存在一条路径。）

原题请见附录

样例： 图 2-1 中所示的平面图最少需要 8 条路径才能覆盖所有的边。

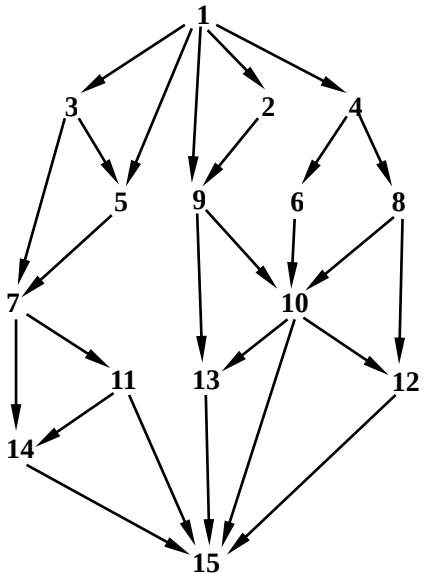


图 2-1

2.1 以网络流为模型

分析一下样例，很快联想到经典的网络流模型：最高点 v_h 是网络的源点，

而最低点 v_l 是网络的汇点。题目中的路径是网络中从源点到汇点的流。要求用路径覆盖图中所有的边，且路径数最少，就是要求网络中每条边的流量大于等于1，并且从源点流出的总流量最小。

因此解决这个问题只需要建立一个有容量下界的网络，然后求这个网络的最小可行流。大致做法：

首先求出可行流：枚举每条流量为 0 的边，设为(i,j)。找到一条从 s 到 i 的路径和一条从 j 到 t 的路径。对“$s-i-j-t$”路径上的每一条边流量加 1，这样既满足了每个点的流量平衡，又满足了边(i,j)的容量下界。然后在可行流上进行修改，从汇点到源点求一个最大可行反向流，就得到了一个最小可行流。

分析算法二的复杂度：求可行流时，可以先预处理源点和汇点到每个顶点的路径，因此构造可行流的时间复杂度为 $O(|E|+|V|)$。求最大流时，可以用朴素的增广路算法，复杂度为 $O(|E|C)$，C 是进行增广的次数。因为是平面图，根据欧拉公式有 $O(|E|)=O(|V|)$，而反向流的流量最大为 $O(|E|)$，所以总的复杂度为 $O(|V|^2)=O(n^2)$。此算法实际效率很高，能够迅速的通过测试数据。但是，这种模型并没有很好的挖掘原题中平面图的性质，所以改进的余地应该很大。

2.2 以偏序集为模型

题目中强调了每个点都有不同的横纵坐标，图是有向无环平面图。而且图中两点之间的有向边似乎反映着一种元素的比较关系。是否存在更好的模型描述此图呢？为了更好的揭示问题的本质，下面引入偏序集。

2.2.1 偏序集的相关概念

偏序集是一个集合 X 和一个二元关系 R，符合下列特性：

a) 对于 X 中的所有的 x，有 $x\,R\,x$，即 R 是**自反的**。

b) 对于 X 中的所有的 x 和 y，只要有 $x\,R\,y$ 且 $x\neq y$，就有 $y\not R\,x$，即 R 是**反对称的**。

c) 只要有 $x\,R\,y$ 和 $y\,R\,z$，就有 $x\,R\,z$，即 R 是**传递的**。

符合这些特性的关系叫做**偏序**，通常用≤标记 R。$a\le b$ 也可以记作 $b\ge a$。若有 $a\le b$，且 $a\neq b$，那么就记作 $a<b$ 或者 $b>a$。<又叫做**严格偏序**关系。含偏序≤的偏序集 X 用$(X,\le)$表示。

令 R 是集合 X 上的一个偏序，对于属于 X 的两个元素 x、y，若有 $x\,R\,y$ 或 y

R x，则 x 和 y 被称作是**可比的**，否则被称为**不可比的**。集合 X 上的一个偏序关系 R，如果使得 X 中的任意一对元素都是**可比的**，那么该偏序 R 就是一个**全序**。例如，正整数集上的小于等于关系就是一个全序。

令 a 和 b 是偏序集$(X, \leq)$中的两个元素。若有 $a < b$，且 X 中不存在另一个元素 c，使得 $a < c < b$，那么就称 a 被 b **覆盖**（或 b **覆盖** a），记作 $a <_c b$。若 X 是一个有限集，由偏序集的传递性易知，任一个偏序关系都可以用多个覆盖关系表示出来，也就是说可以用覆盖关系有效的表示偏序关系。

有限偏序集$(X, \leq)$的图表示（也就是哈斯图）是用平面上的点描述的。偏序集中的元素用平面上的点来表示。若有 $a < b$，那么 a 在平面上的位置（严格说是坐标平面中的纵坐标）就应当低于 b 在平面上的位置。若 $a <_c b$，那么 a 和 b 之间连一条边。也可以用有向图来表示偏序关系，图中的每个结点对应偏序集中的每个元素。若偏序集中的两个元素有 $a <_c b$，那么对应到图中的两个结点 a 和 b，就有一条从 b 到 a 的有向边(b, a)。因此可以看出原图事实上是一个偏序集的图表示。

链：链是 E 的一个子集 C，在偏序关系≤下，它的每一对元素都是可比的，即 C 是 E 的一个全序子集。

反链（或称杂置）：顾名思义，它和链的定义恰恰相反。反链是 E 的一个子集 A，在偏序关系≤下，它的每一对元素都是不可比的。

链和反链的大小是指集合中元素的个数。

2.2.2 构筑原问题的偏序集模型

有了上文有关偏序集的概念，不难搭建出原问题的偏序集模型：令原图表示的偏序集为$(X, \leq)$，而新构造的偏序集为$(E, \leq)$。则集合 E 满足 $E = \{(u, v) \mid u$ 或 v 且 $\geq u\}$，即 E 中的元素全部是图中的有向边。令 a、b 为 E 中的两个元素，设 $a = (u_a, v_a), b = (u_b, v_b)$。当且仅当 $u_a \leq u_b$ 且 $v_a \leq v_b$ 时，有 $a \leq b$，即存在一条从有向边 a 到有向边 b 的路径。当且仅当 $v_a = u_b$ 时，有 $a \leq_c b$。

原问题可以重新用偏序集语言表述为：将偏序集（$E, \leq$）划分成最少的链，使得这些链的并集包含所有 E 中的元素。直接计算链的个数似乎并不容易，好在有 Dilworth 定理揭示了链与反链的关系，从而使得问题的目标进一步转化。

定理 2.1 : Dilworth 定理 令（$E, \leq$）是一个有限偏序集，并令 m 是 E 中最大反链的大小，M 是将 E 划分成最少的链的个数。在 E 中，有 $m = M$。

证明过程请参见附录。

根据 Dilworth 定理，问题转化成求 E 中最长的反链的大小。也就是要求在原图中选尽量多的边，同时保证选出的边是互不可达的（即在$(E, \leq)$中不可比）。如何求解最长的反链呢？事实上，这和原题给出的平面图有很大关系，接下来，返回到原图上继续讨论。

2.2.3 从偏序集模型回归到原题

由于原题给出的图是平面图，而且图中顶点也是从左到右给出的，那么对于反链中的所有边都能按照从左到右的顺序排列好。如果用一条线将最长反链所对应的边从左到右连起来（如图 2-2 所示），那么这条线不会与平面图中的其它边相交。更加准确地说：

定理 2.2：将最长反链所对应的边从左到右排列好，相邻的两条边一定是在同一个域（闭曲面）中。所谓的域，是由从一个点到另一个点（一个是极高点，一个是极低点）的两条不同路径（两条路径没有公共边）围成的一个曲面，在这个曲面里没有其他的点和边（如图 2-3 所示），记作 F。在围成域 F 的两条路径中，左边的那条路径定义为 F 的左边界，右边的那条路径定义为 F 的右边界。

关于定理 2.2 的简略证明请参见附录。

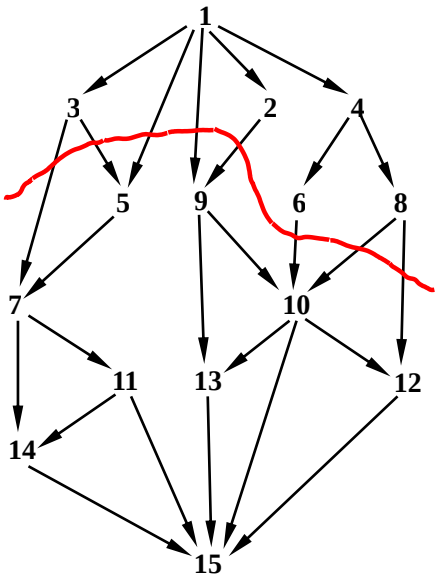


图 2-2

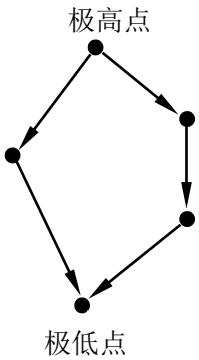


图 2-3

受定理 2.2 的启发，我们可以用递推的方法求得图中最长反链的长度。设 $f(x)$ 表示在边 x 左边的平面区域中以 x 结尾的最长反链的长度。设 x 在某个域 F 的右边界上，有 $f(x)=\max\{f(y)\}+1$(y是F左边界上的边)。因为根据定理 2.2，若 x 在某个最长反链中，那么反链中和 x 相邻且在 x 左边的边，只有可能在域 F 的左边界上。得到这个递推式后，只需要按照从左到右从上到下把每一个域求出进行递推即可。

2.2.4 相应的算法设计

可以用 DFS 深度优先遍历实现平面图中域的寻找。DFS 中需要记录两个信息：结点的颜色和扩展它的父结点。每个结点的颜色用 C[u]来记录。C[u]有三种状态：白色表示结点 u 尚未被遍历，一开始所有结点的颜色都是白色；灰色表示结点 u 已经被遍历，但是它尚未检查完毕，也就是说它还有后继结点没有扩展；黑色表示结点 u 不但被遍历且被检查完毕。扩展它的父结点用 pre 记录。

算法的大致框架如下：

DFS（结点 u）
begin
  C[u] ← 灰色
  while v 是 u 后继结点 **do** （按照从左到右的顺序扩展 u 的后继结点 v）
    if C[v] 是白色
    then begin
      pre[v] ← u
      DFS(v)
    end else begin
      v_l ← v
      v_h ← v
      while C[v_h] 是黑色 do
        v_h ← pre[v_h] (是黑色，说明是域的边界上的结点；灰色就是极高点)
      递推求出右边界的 f(x) (pre 回溯的边是左边界，v_h-u-v_l 是右边界)
      pre[v] ← u
    end

C[u] ← 黑色

end

计算上述算法的复杂度：*a*.对每一个点进行 DFS 遍历，复杂度为 $O(|E|)$；*b*.回溯寻找每个域边界上的边，并且进行递推求解。由于是平面图，每条边最多属于两个不同域的边界，因此这一步的复杂度为 $O(|E|+|F|)$。因为原题给出的图是平面图，根据欧拉定理，边数$|E|$和域数(或者说面数)$|F|$都是 $O(n)$级别的。因此总的时间复杂度为 $O(n)$。

2.3 小结

方法一利用网络流模型直接的体现了原题的网络（有向无环）特性，解法具有一般性，但是没有充分的体现出原题的平面图性质。而方法二很好的利用偏序集模型实现了问题目标的转变，从原来的网络流问题回归到问题本身的平面图上，完整的揭示了问题的本质。正是由于回归到问题的本质，后面才能用 DFS 充分挖掘平面图的性质，得到最优复杂度的算法。

从上述两种方法的比较可以看出，两种不同的图论模型导致了两种算法在时间复杂度上的较大差异，可见选择模型的重要性。正所谓“磨刀不误砍柴功”，在设计算法之前，选择一个正确的图论模型往往能够起到事半功倍的效果，不仅能降低算法设计的难度，还使设计出的算法简单高效。

在为实际问题选择合适的图论模型的时候，不能仅仅局限于问题表面所表现的某些性质，只根据自有的经验去解题，否则会落入俗套，得不到效率最高的算法。新题新作，应该创新思路，深入分析问题的各种性质，将这些性质结合在一起，从而寻找到最能体现问题本质的图论模型。

很多时候，最终算法并不是基于所选图论模型来设计的，就如方法二，偏序集并没有出现在 DFS 中，但一旦想到偏序集，问题可以说就解决了一半。图论模型更多的是思考的一种过渡，使思路变得清晰，就像一座灯塔，指引你到成功的彼岸。

三、充分挖掘和利用模型的性质

上一节讲述了选择合适模型的重要性和搭建图论模型的方法。建立模型仿佛

是为算法设计搭建一个平台，接下来的工作是在这个平台上充分挖掘和利用原题的性质，设计一个解决问题的好算法。如何挖掘和利用模型的性质呢？下面同样用一个具体实例来说明。

例二：爱丽丝和鲍勃（ACM *Central European Programming Contest* 2001 Problem A : Alice and Bob）

题目描述：爱丽丝和鲍勃在玩一个游戏。爱丽丝画了一个有 n（$n \le 10000$）个顶点的凸多边形，多边形上的顶点从 1 到 n 按任意顺序标号。然后她在多边形中画了几个不相交的对角线（公共点为顶点不算相交）。她把每条边和对角线的端点标号告诉了鲍勃，但没有告诉他哪条是边，哪条是对角线。鲍比必须猜出顶点顺（逆）时针的标号序列，任意一个符合条件的序列即可。

原题请见附录

例如：$n = 5$，给出的边或对角线是(1,4),(4,2),(1,2),(5,1),(2,5),(5,3),(1,3)。那么一个可能的顶点标号顺序为(1, 3, 5, 2, 4)。对应的多边形如图 3-1：

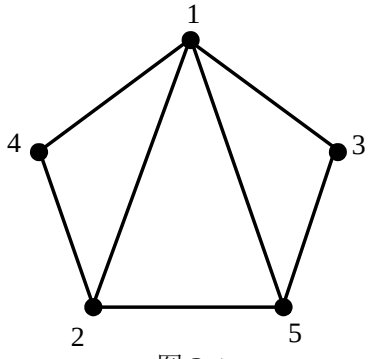


图 3-1

3.1 建立模型

根据题目意思，不难得到它的图论模型：凸多边形对应了一个有 n 个顶点的图 G，多边形的边和对角线对应着图 G 中的边。而且十分明显的是，图 G 是一个平面图，根据欧拉公式，图中边的数量级为 $O(n)$。

研究图 G 的平面图性质可以发现**结论 3.1：一条对角线将凸多边形分成了两部分，每部分都至少含有一个除对角线外顶点，而且这两部分分居在对角线的两侧**。由此可知，在图 G 中只存在惟一的一条哈密顿回路。因为对角线会将多边形分成不相关连的两部分，所以对角线不可能存于哈密顿回路上。因此哈密顿

回路上的边都是由多边形上的边组成，而多边形的边只有 n 条，可知哈密顿回路也就只有一条。不妨设这条哈密顿回路为 $H_1,H_2,H_3\cdots\cdots H_n$。问题的目标就是要找到这个哈密顿回路。下面讨论如何利用这些性质来设计算法。

3.2 利用边的连通性

由结论 3.1 可知，如果一条边是对角线，那么将对角线的两个端点从图 G 中删除，图 G 一定会变成两个互不可达的连通分块；而如果一条边是多边形上的边，那么将这条边的两个端点删除，图 G 将仍然是连通的。也就是说能够根据图 G 中边的连通性，来判断一条边是对角线还是边。因此，得到算法一：

算法一：

1．枚举图中的每条边(u, v)，进行如下判断：将 u 和 v 两个顶点从图 G 中删除得到新图 G’。在新图 G’任选一点 a，然后从 a 开始对图 G’进行遍历。如果 a 能够到达图 G’中的其它点，那么就说明图 G’是连通的，(u, v)是多边形的边；否则，图 G’不连通，(u, v)是多边形的对角线。

2．将图 G 中所有对角线删除，得到新图 C。这时的新图 C 便是图 G 中的哈密顿回路，因此只要对其进行一次遍历就可以得到多边形顶点的标号序列了。

分析一下算法一的复杂度：步骤 1 中，要枚举的边最多为 $2n$ 条，每判断一次图的连通性为 $O(n)$，所以复杂度为 $O(n^2)$；步骤 2 中，删除边和遍历新图的复杂度都为 $O(n)$。所以总的复杂度为 $O(n^2)$。n 最大时候达到 10000，因此算法一的复杂度稍稍高了一点，仍要继续优化。

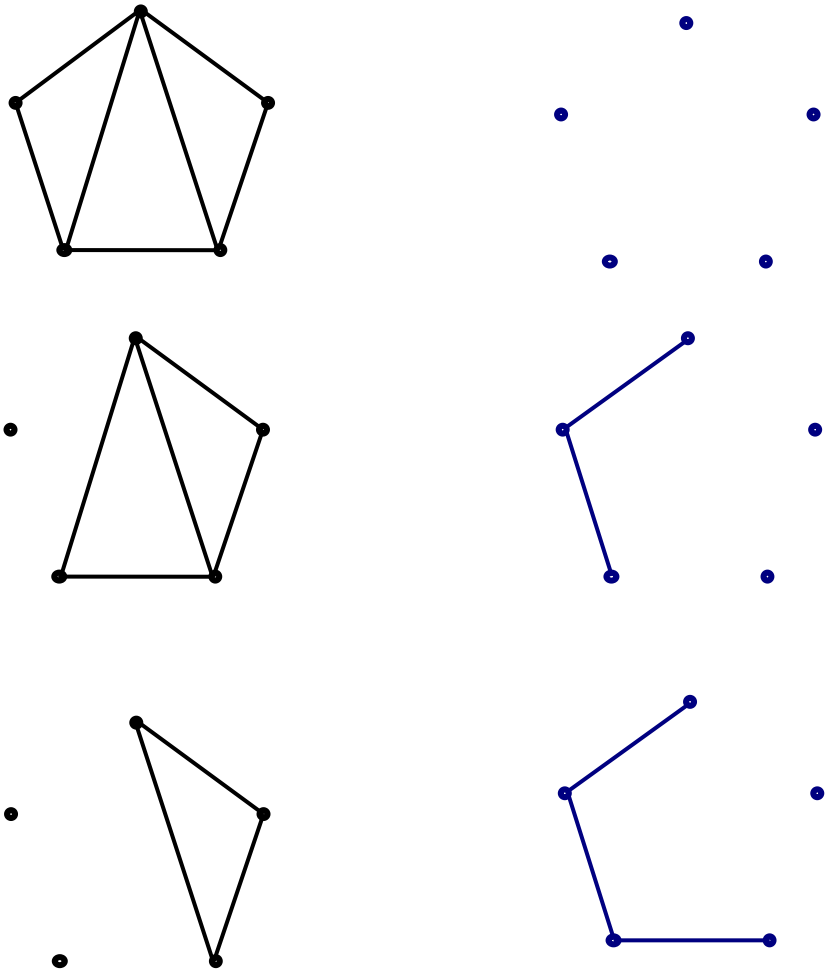
3.3 分而治之

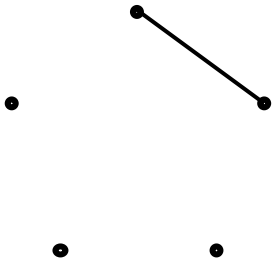
继续挖掘图 G 的性质发现，由于图 G 中的对角线是不相交的，那么必定存在一个顶点 u，它的度数为 2。而且可以肯定的是，和 u 相连的两条边一定是多边形的边。将以 u 为顶点的三角形从多边形上删除，剩下的图形仍然是一个多边形。是否能用这个性质判断一条边是多边形的边还是对角线呢？答案是肯定的。

由于图中一定存在一个顶点 u，它的度数为 2。设 u 是哈密顿回路上的顶点 H_i，其相邻的两个顶点为 H_{i-1} 和 H_{i+1}。而(H_{i-1}, H_i)和(H_i, H_{i+1})两条边都是多边形的边，将这两条边添加到一个附加图 C 中（附加图 C 一开始只有 n 个顶点，对应

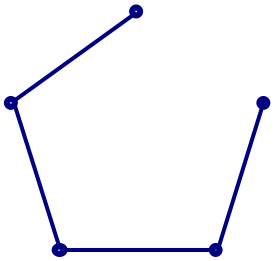
着多边形的顶点，顶点之间没有边相连)。将以 u 为顶点的三角形（即三角形 $H_iH_{i-1}H_{i+1}$）从多边形上砍掉之后，剩下的图形仍然是一个多边形。也就是说，可以把 H_i 从图 G 中删除，若 H_{i-1} 和 H_{i+1} 之间没有边相连，就添加一条新边（H_{i-1},H_{i+1}）（虚边），得到新图 G'。这时，新图 G'中仍唯一存在一条哈密顿回路……H_{i-2} ,H_{i-1}, H_{i+1}, H_{i+2}……。图 G'和图 G 具有同样的性质，因此也至少存在一个度为 2 的点，令其为 H_j。那么(H_{j-1}, H_j)和(H_j, H_{j+1})这两条边有可能为多边形的边、多边形的对角线或者新添加的虚边，将其中多边形的边添加到图 C 中。然后按照同样的方法把 H_j 从图 G'删除，得到一个新图……。以此类推，不断地从新图中找到度为 2 的点，然后将其相应的两条边中是多边形的边添加入图 C 中，接着从图中删除这个点。如此反复，直到图中不存在度为 2 的点。然后把图中剩下的边中，是多边形的边的添加到图 C 中。最后，图 C 便是要求的原始多边形。

样例的模拟过程如下：





左边是图 G 的变化过程



右边是图 C 的变化过程

图 3-2

还存在一个问题没有解决：如何判断移除的边是否是原多边形的边呢？首先考虑一下，一条边（H_i, H_j）什么时候才会被移除。一条边（H_i, H_j）当且仅当成为某个多边形的边时，它才有可能被移除。如果（H_i, H_j）是原始多边形的一条对角线，当且仅当哈密顿回路上从 H_{i+1} 到 H_{j-1} 的顶点都已经被删除，（H_i, H_j）才有可能成为新多边形的边。如何判断从 H_{i+1} 到 H_{j-1} 的顶点是否已经被删除呢？这时构造的附加图 C 就起到了作用！若从 H_{i+1} 到 H_{j-1} 的顶点已经被删除，则这些顶点在原多边形上相应的边都已经添加到图 C 中，H_i 到 H_j 在图 C 中一定存在一条路径。因此判断一条移除的边（H_i, H_j）是否是对角线，只需要判断在图 C 中 H_i 和 H_j 是否连通。值得注意的是，当图 C 中已经有了 n -1 条边时，剩下的那条边会被判断成对角线，但此时已经能够确定多边形顶点的标号序列了。

算法的大致轮廓已经清楚了，下面为算法选择合适的数据结构：

a. 图 G 的存储结构：由于图 G 是稀疏图，可以用邻接表存储，用来查找度为 2 的顶点与哪些边相连。还要用一个哈希表存下所有的边，用于查找任意两个点是否相连。设一条边为(u, v)，哈希表可以用$u \times n + v$作为关键字，哈希函数为 $f(u,v) = (u \times n + v) \bmod P$，$P$ 为大质数。这样在保证查找复杂度仍然是 $O(1)$的情况下，存储空间比邻接矩阵小了很多。

b. 枚举度为 2 的结点：很容易想到用最小堆来找出度为 2 的顶点，不过这样的复杂度稍稍高了一点。由于顶点的度数只减少不增加，而且真正有效的度数只有 1 和 2，所以可以建立四个桶来替代堆。将顶点按度数分别放到四个不同的桶中：度数为 0、度数为 1、度数为 2、度数大于 2。在每个桶中用双向链接表存储下不同的结点。当一个顶点的度数被修改的时候，从原桶中删除，放到相应的桶中。在双向链表中的插入和删除结

点，时间复杂度都是 $O(1)$。而因为每个结点的度数是不断减少的，所以每个结点最多进出每个桶一次。综上所述，在这些桶中查找和调整一个度为 2 的结点所需的时间复杂度为 $O(1)$。

c. 在图 C 中添加边，判断任意两点的连通性：可以采用并查集来实现边的添加（即将两个连通子图合并）和判断点的连通性（即判断两个点是否在同一个连通子图中）。添加一条边和判断一对点的连通性两种操作的均摊复杂度为 $O(\alpha(n))$。

下面是对算法二的总括：

算法二

1．根据每个顶点的度数将 n 个顶点放到四个桶中。初始化附加图 C。

2．如果存在一个度数为 2 的顶点 u，则接下来执行第 3 步，否则转步骤 6。

3．在邻接表中找到和 u 相连的两个顶点为 v_1 和 v_2。

4．首先判断(u,v_1)、(u,v_2)是否为虚边。若不是虚边，用并查集检查其是否为对角线。最后将是多边形的边插入到图 C 中。

5．将顶点 u 标记为不可用。用哈希表检查边(v_1,v_2)是否存在图 G 中。如果边(v_1,v_2)不存在，则添加一条虚边(v_1,v_2)，v_1 和 v_2 的度数不变；否则将 v_1 和 v_2 的度数减一。若 v_1 或 v_2 修改后的度数不符合所在桶的性质，则进行调整。然后转到步骤 2。

6．找到残图 G 中所有度数为 1 的顶点，一一检查这些顶点相应的边在附加图 C 中的连通性，把非对角线和虚边的边插入到图 C 中。

7．遍历图 C 得到原始多边形顶点的标号顺序。

在每个顶点删除时，最多添加一条虚边，因此原始边和虚边的总数仍然是$O(n)$的级别。现在分析一下时间复杂度：步骤 3 中，每个顶点最多查找一次，因此复杂度为边数，即 $O(n)$；步骤 4 和步骤 6 中，用并查集查找和插入的均摊时间复杂度为 $O(n\alpha(n))$；步骤 5 中，用哈希表查找一条边的复杂度为 $O(1)$，边的插入和桶的调整也是 $O(1)$，一共进行顶点数次，所以时间复杂度为 $O(n)$。综上所述，算法二的复杂度为 $O(n\alpha(n))$。

算法二的时间复杂度已经十分低了。但追求无止境的我们不能浅尝辄止，因为算法二仍存在需待改进的地方：编程复杂度实在太高了！算法二虽然已经较为

充分的挖掘出问题的各种特性，但在利用这些特性时，是通过不断分解成子问题然后一一击破的方法来实现的。我们不禁提出一个问题：是否能找到一种综合的方法，将所有特性一起利用，使得解决方法既简便又高效呢？

3.3 DFS 树反映的问题特性

回到多边形所在的平面图上。若按深度优先来遍历平面图，当经过一条对角线时，平面图便被分成了两个部分。由于是深度优先遍历，那么在这两部分剩下的点中，有一部分的点会被优先遍历到。因此可以模仿 DFS 寻找块的算法，利用顶点的遍历顺序和 low 函数来判断边的性质。

首先定义一下 DFS 中需要用到的两个函数：

dfn[*v*]表示 *v* 是深搜中第几个被遍历到的。

$\mathrm{low}[v]=\min\{\mathrm{dfn}[v],\mathrm{low}[w_1],\mathrm{dfn}[w_2]\}$ 其中 w_1 表示 v 的儿子结点，w_2 表示 DFS 树中异于 *v* 的父亲结点的其他祖先结点。

接下来分情况讨论如何用 dfn 和 low 函数判断一条边的性质：

考虑 DFS 树上的一条边(*u*, *v*)，其中 *u* 是 *v* 的父亲结点。由于图 *G* 是一个块，因此每个点的儿子个数不会超过 2。根据 *v* 的度数分下面四种情况：

1．*v* 的儿子数为 0：意味着(*u*,*v*)是原始多边形的边。否则的话，*v* 所有的祖先结点都在(*u*,*v*)的一侧，而另一侧一定仍有点存在，与儿子数为 0 矛盾。令 *x* 是与 *v* 直接相连的祖先结点中 dfn 值最小的一个结点，那么(*x*,*v*)一定也是原始多边形的边。略证：如图 3-3，另(*x*,*v*)为多边形的边。*v* 不是 *x* 的父亲结点，说明 *x* 一定先于 *u* 被遍历到，而且 *x* 是 *u* 的祖先结点。也就是说，*x* 到 *u* 存在一条路径，路径上的边都是由 DFS 树上的边组成。令 *y* 为异于 *x* 和 *u*，而

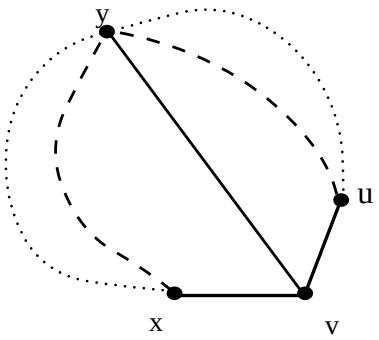


图 3-3

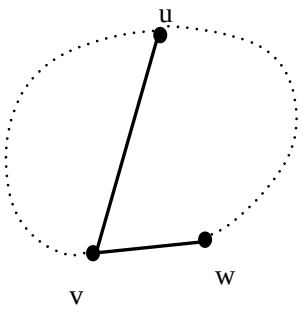


图 3-4

与 *v* 相连的点。由于图 *G* 是平面图，那么 *y* 一定在 *x* 到 *u* 路径上。因此 *y* 一

定后于 x 被遍历，即 dfn[y]小于 dfn[x]。所以 x 一定是与 v 相连的祖先结点中 dfn 值最小的。

2．v 的儿子数为 1，且 u 为 DFS 树的根：易知，(u, v)一定是多边形的边。

3．v 的儿子数为 1，且 u 不是 DFS 树的根：令 w 为 v 惟一的儿子。

若 low[w]大于等于 dfn[u]，则(u, v)为多边形的对角线。因为(u, v)将平面分成了两部分，w 和其子树结点是遍历不到另一部分的，如图 3-4。令 x 是与 v 直接相连的祖先结点中 dfn 值最小的一个结点，那么(x, v)一定是原始多边形的边（证明同情况 1 中的分析）。而与 v 相连的另一条多边形的边可以在遍历 w 时找到。

若 low[w]小于 dfn[u]，则(u, v)是多边形的边，而与 v 相连的另一条多边形的边可以在遍历 w 时找到。

4．v 的儿子数为 2，则与 v 相连的两条多边形的边可以在遍历儿子结点时找到。

考虑上述四种情况已经能够判断图 G 中所有边的性质了。因此只需要对图 G 进行一次深度优先遍历，就能确定多边形顶点的标号顺序。所以算法三的时间复杂度为线性的——$O(n)$！

3.4 小结

上述三种方法都是建立在同一个模型上的，不同的方法对模型性质挖掘的深度不同也就决定了不同的时间复杂度。算法一运用了定义法，利用对角线将多边形划分成两部分的性质找到多边形中所有的对角线。而在算法二的设计过程中，发现了每次删除多边形的一个角时仍是一个多边形，也就是找到了子问题的相似性，这样就可以不断地缩小问题的规模。算法二还体现了一种化归的思想，将未知问题分成几个步骤（或者子问题），每个步骤都是我们熟知的，因此可以为每一步选择最好的算法。由于每个子问题都解决了，原问题也就迎刃而解。可以说方法二是一种分析法。算法三体现了一种综合思想，不是单独考虑每一条边的连通性，而是从全局考虑，发现了多边形的边和对角线在 DFS 树中的区别。因此算法三的设计一气呵成，体现了图论中一种“序”的优美性。

三种不同的方法也给了我们不同的启示：定义法告诉我们，当找不到问题思考的方向时，可以考虑从问题最基本的性质入手寻找突破口。分析法的好处是，

分解后的子问题一般比原问题要简单，困难的是每一部分都会影响到最终算法的复杂度，因此每个部分的算法设计都要精益求精。综合法的运用需要有全局观，要能够发现最具代表性的模型特性。

四、总结

“模型”一词曾在无数论文中被提及，它是图论基本思想的精华，是解决图论问题的关键。建立模型要求我们熟悉经典模型，同时又要求我们能够勇于探索，大胆创新，敢于跳出经典模型的框框。利用模型的特性需要我们独具慧眼，能够抓住问题的本质，击中问题的要害。

同时文章中还介绍了三种解决问题的方法：定义法、分析法和综合法。这些方法和“模型”一样，具有解决信息学奥赛问题的普适性。它们不仅仅是一种方法，更是一种思维的方式和思考的角度。灵活地运用和掌握它们，有利于我们研究算法、探索科学。

例题二的解析过程，不仅仅体现了图论的基本思想，同时还展现了算法与数据结构的完美结合，以及算法的优化思想。算法与数据结构的结合是优秀程序设计的必然要求，是知识融会贯通的体现。优化是为了进一步的完善程序设计，是一种不断进取的精神。这些同样是今后科学研究的必备素养。

本文或许没有完美的概括图论的基本思想和方法，仅仅是作者对图论的一点理解和想法。但作者希望本文对读者有一定的启发，使读者能够对图论基本算法及方法进行深入思考和总结。所谓万变不离其宗，只有最基本的思想和方法才是解决问题的不二法门。

【参考文献】

[1]《Introductory Combinatorics (3rd Edition)》（美）Richard A. Brualdi 著

[2]《算法艺术与信息学竞赛》 刘汝佳 黄亮 著

[3] Poland Olympiad of Informatics 2002 Stage III：Skiers

[4] ACM Central European Programming Contest 2001 Problems and Solutions

【感谢】

衷心感谢向期中老师在学习上对我的指导和帮助
衷心感谢栗师同学在例题一上给我的启发
衷心感谢胡伟栋和王俊同学对我的大力帮助
由衷感谢周戈林、肖湘宁两位同学对我的论文提出宝贵的意见

【附录】

1．Dilworth 定理的证明：

易知 M 一定大于等于 m，因为最长反链中的元素一定分在不同的链中。因此只需要证明一定存在 M 小于等于 m。接下来，通过对 E 的元素个数 n 的归纳法证明之。

当 n=1 时，定理显然成立。

当 n>1 时，分两种情况讨论：

情况 1：存在一条大小为 m 的反链 $A=\{a_1,a_2,\cdots,a_m\}$，它既不是 P 中的所有极大元的集合，也不是所有极小元的集合。

现定义 $A^-=\{x\in P\mid \exists_i x\le a_i\}$，即 A^-中任一个元素 x 在 A 中能找到一个元素 a_i，使得 $x \leqslant a_i$。类似地，定义 $A^+=\{x\in P\mid \exists_i x\ge a_i\}$。这样直观上的看，$A^-$是所有 A“下方”的元素组成，而 A^+是所有 A“上方”的元素组成。而且还有下列性质成立：

1）由于至少存在一个极大元不在 A 中，所以这个极大元也不在 A^-中。因此 $A^-\neq P$，A^-中的元素个数小于 P 的元素个数 n。

2）由于至少存在一个极小元不在 A 中，所以这个极小元也不在 A^+中。因此 $A^+\neq P$，A^+中的元素个数小于 P 的元素个数 n。

3）$A^+\cap A^-=A$。反设存在一个 x 在 $A^+\cap A^-$中，而不属于 A。根据反设，A 中存在两个元素 a_i 和 a_j，满足 $a_i \leqslant x \leqslant a_j$，这与 A 是一条反链矛盾。因此性质 3 成立。

4）$A^+\cup A^-=P$。反设存在一个 x 在 $A^+\cup A^-$中，而不在 P 中。根据反设，对于 A 中的任一个元素 a_i，既不满足 $a_i \leqslant x$，也不满足 $a_i \geqslant x$ 。也就是

说 x 和 A 中的所有元素都是不可比的，那么 $A\cup x$ 是一条比 A 更长的反链，这与 A 是 P 中最长的反链矛盾。因此性质 4 成立。

因为 A^+、A^-中的元素个数小于 n，根据归纳假设，A^+、A^-一定能够划分成 m 条链。设 A^+划分成 $C_1^+,C_2^+\cdots C_m^+$，且 $a_i\in C_i^+$。类似地，设 A^-划分成 $C_1^-,C_2^-\cdots C_m^-$，且 $a_i\in C_i^-$。显然的有，对于所有 i，a_i是C_i^+中的极小元。因为若 C_i^+中的极小元不是 a_i 而是 x，那么根据 A^+的定义，存在一个 $a_j \leqslant x$，则有 $a_j \leqslant x \leqslant a_i$，这与 A 是反链矛盾，所以 a_i 是极小元。同样地，a_i是C_i^-中的极大元。a_i 是 C_i^+中结尾的那些元素，是 C_i^-开始的那些元素，将 C_i^+和 C_i^-接在一起形成了 m 条链，它们是 P 的一个划分。

情况 2：最多存在两条大小为 m 的反链，即所有的极大元集合和极小元集合中任一个或两个。令 x 是极小元，而 y 是极大元且 $x \leqslant y$(x 可以等于 y)。此时，$P-\{x,y\}$的反链的最大的大小为 $m-1$。根据归纳假设，$P-\{x,y\}$可以被划分成 $m-1$ 条链。这些链和链{x, y}一起构成了 P 的一个划分。

2．定理 1.2 的证明：

反设最长反链 A 中存在两条相邻的边不再同一个域中，令这两条边为 a 和 b，且 a 在 b 的左边。一条边最多属于两个不同的域。有 3 种情形：

1) a 是某个域 F_1 左边界上的一条边，那么 F_1 右边界上的一条边 c 与 a 和 b 都是不可比的，那么 $A\cup\{c\}$ 是一条更长的反链，矛盾。

2）b 是某个域 F_2 右边界上的一条边，那么 F_2 左边界上的一条边 d 与 a 和 b 都是不可比的，那么 $A\cup\{d\}$ 是一条更长的反链，矛盾。

3）a 是域 F_1 右边的那条路径，b 是域 F_2 左边的那条路径。设 $a=(u_1,v_1)$，$b=(u_2,v_2)$。由于平面图中的最高点 v_h 到 u_1 和 u_2 都至少存在一条路径，因此这些路径之间至少存在一个公共点既能够到达 u_1 也能够到达 u_2，令这些公共点中纵坐标最低的点为 u_h。类似地，设既能够被 v_1 到达也能被 v_2 到达的公共点中纵坐标最高的点为 u_l。由于 a 和 b 不在同一个域中，因此路径 1（$u_h-u_1-v_1-u_l$）和路径 2（$u_h-u_2-v_2-u_l$）之间至少存在另一条路径 3，这条路径要么是从路径 1 连到路径 2，要么是从路径 2 连到路径 1。因此路径 3 上一定存在一

条边 e，e 和 a、b 都是不可比的，那么 $A\cup\{e\}$ 是一条更长的反链，矛盾。

因此反链中相邻的两条边一定是在同一个域中。

3. 滑雪者原题

Skiers

On the south slope of Bytemount there is a number of ski tracks and one ski lift. All the tracks run from the top station of the ski lift to the bottom one. Every morning a group of ski lift workers examines the condition of the tracks. They together take the lift up to the top station, and next each of them skis down along a chosen track to the bottom station. Each worker skis down only once. The tracks of the workers may partially be the same. Each track examined by any of the workers leads always downwards.

The map of ski tracks consists of a network of clearings connected by cuttings in the forest. Each clearing lies on different height. Any two clearings may be directly connected by at most one cutting. Skiing down from the top to the bottom station one can choose a track to visit any one clearing (although probably not all of them in a single run). Ski tracks may cross only on clearings and do not run through tunnels nor on bridges.

Task

Write a program which:

- reads from the text file nar.in the map of ski tracks,
- computes the minimal number of workers to examine all the cuttings between the clearings,
- writes the result to the text file nar.out.

Input

In the first line of the input file nar.in there is one integer *n* equal to the number of clearings, 2<=*n*<=5 000. The clearings are numbered from 1 to *n*.

Each of the successive *n*-1 lines contains a sequence of integers separated by single spaces. The integers in the (*i*+1)-st line of the file specify which clearings the cuttings from the clearing number *i* lead down to. The first integer *k* specifies the number of those clearings.

The successive *k* integers are their numbers ordered in the direction from west to east, according to the arrangement of the cuttings leading to them. The top station of the ski lift lies on the clearing number 1, and the bottom one on the clearing number *n*.

Output

In the first and only line of the output file nar.out there ought to be exactly one integer - the minimal number of workers that are able to examine all the cuttings in the forest.

4.爱丽丝和鲍勃原题

Problem ： Alice and Bob

this is a puzzle for two persons, let's say Alice and Bob. Alice draws an n-vertex convex polygon and numbers its vertices with integers 1; 2;…; n in an arbitrary way. Then she draws a number of noncrossing diagonals (the vertice-s of the polygon are not considered to be crossing points). She informs Bob about the sides and the diagonals of the polygon but not telling him which are which. Each side and diagonal is specified by its ends. Bob has to guess the order of the vertices on the border of the
polygon. Help him solve the puzzle.

Example

If n = 4 and (1,3), (4,2), (1,2), (4,1), (2,3) are the ends of four sides and one diagonal then the order of the vertices on the border of this polygon is 1, 3, 2, 4 (with the accuracy to shifting and reversing).

Task

Write a program which for each data set:

- reads the description of sides and diagonals given to Bob by Alice,
- computes the order of the vertices on the border of the polygon,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets,$1 \le d \le 20$. The data sets follow.

Each data set consists of exactly two consecutive lines.

The first of those lines contains exactly two integers n and m separated by a single space, $3 \le n \le 10\,000$, $0 \le m \le n \le 3$. Integer n is the number of vertices of a polygon and integer m is the number ofits diagonals, respectively.

The second of those lines contains exactly 2(m+n) integers separated by single spaces. Those are ends of all sides and some diagonals of the polygon. Integers aj; bj on positions 2j -1 and 2j, $1 \le j \le m+n$,$1 \le a_j \le n$, $1 \le b_j \le n$, $a_j \ne b_j$,

specify ends of a side or a diagonal. The sides and the diagonals can be given in an arbitrary order. There are no duplicates.

Alice does not cheat, i.e. the puzzle always has a solution.

Output

The output should consist of exactly d lines, one line for each data set.

Line i, 1 ≤ i ≤ d, should contain a sequence of n integers separated by single spaces | a permutation of 1; 2; …; n, i.e. the numbers of subsequent vertices on the border of the polygon from the i-th data set; the sequence should always start from 1 and its second element should be the smaller vertex of the two border neighbours of vertex 1.