浅析二分图匹配在信息学竞赛中的应用

湖南省长沙市长郡中学　王俊

[摘要]

本文通过对几道信息学竞赛题目的分析，举例说明了二分图匹配在信息学竞赛中的应用。二分图匹配的应用一般是通过分析某些最优化问题的性质，构造出二分图，再通过求得该二分图的最大匹配，最佳匹配等各种形式的匹配从而解决原问题。

[关键字]

匹配　二分图　最小权　最大权　优化

[正文]

一　引 言

二分图匹配是信息学竞赛中一类经典的图论算法，在近年来信息学竞赛中有广泛应用。如果可以以某一种方式将题目中的对象分成两个互补的集合，而需要求得它们之间满足某种条件的一一对应的关系时，往往可以抽象出对象以及对象之间的关系构造二分图，然后利用匹配算法来解决。这类题目通常需要考察选手对原题进行建模，构造二分图，设计匹配算法，并对其算法进行适当优化等多方面能力。下面就通过两道例题来说明二分图匹配在信息学竞赛中的一些应用。

二　Railway Communication[1]

2.1　问题描述

某国有 n 个城镇，m 条单向铁路。每条铁路都连接着两个不同的城镇，且该铁路系统中不存在环。现需要确定一些列车运行线，使其满足：

I)　每条铁路最多属于一条列车运行线；
II)　每个城镇最多被一条列车运行线通过（通过包括作为起点或终点）；
III)　每个城镇至少被一条列车运行线通过；
IV)　列车运行线的数量应尽量小。
V)　在满足以上条件下列车运行线的长度和应该尽量小。

[1] Saratov State University 252. Railway Communication

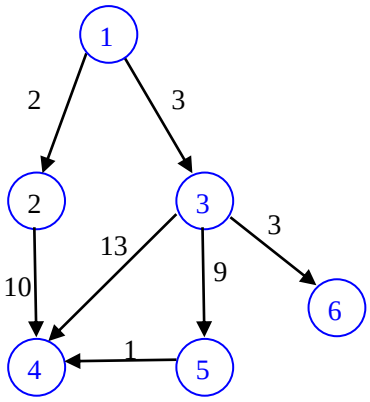

图 1

2.2　问题分析

题目要求列车运行线数最少，又要求在此条件下列车运行线的长度和最小，不便于一起考虑，我们不妨分步研究，先考虑列车运行线数最少的子问题。则该子问题可建立如下数学模型：给定一个有向无环图 $G^0=(N^0, A^0)$，用尽量少的不相交的简单路径覆盖 N^0。

我们可以给问题建立一个二分图 $G=(N, A)$，如图 2。

a) 建立两个互补的结点集合 X 和 Y，把点 $i(i \in N^0)$拆成 X 结点 i 和 Y 结点 i'。$N = X \cup Y$。

b) 对于图 G^0 中有向边(i, j)，$(i, j) \in A^0$，则在 A 中加入边(i, j')。如果在 G^0 中选定(i, j)作为某条覆盖路径中的边，则在 G 中选定边(i, j')。

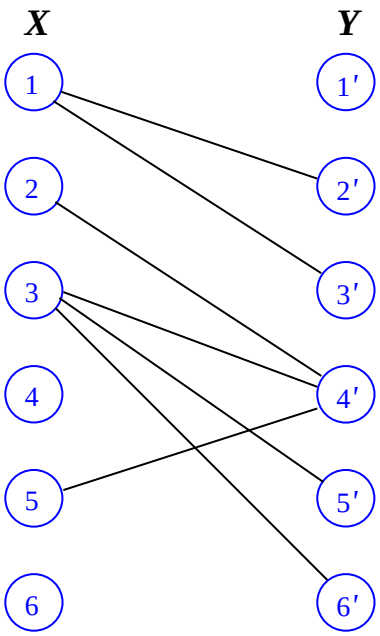

图 2

对于图 G^0 中的任意一个结点 i，可分为三类：

I) 某条覆盖路径的起点，即它没有前驱结点，那么在二分图 G 中点 i' 的邻边均没有选。

II) 某条覆盖路径内部的点，即它有一个前驱结点和一个后继结点，那么在二分图 G 中 i，i' 的邻边各选了 1 条。

III) 某条覆盖路径的终点，即它没有后继结点，那么在二分图 G 中点 i 的邻边均没有选。[2]

这样问题就转化成在二分图 G 中选一些边，且每个点的邻边中至多有一条被选中，显然这是一个二分图匹配的问题。又因为题目要求路径数最少，即路径终点数最少，即尽量多的匹配，所以是求该二分图的最大匹配，可以套用经典的匈牙利算法求解。

再来考虑求列车运行线总长度最小的问题。设原图 G^0 中边$(i，j)$的边权为 $W^0_{i,j}$，则给图 G 的边$(i，j')$加入边权 $W_{i,j}$ ，$W_{i,j}=W^0_{i,j}$ (如图 3)。原问题是求图 G^0 中在保证覆盖路径数最少时求覆盖路径总长度最小，即在二分图 G 中求保证匹配数最大时匹配边的权值和最小。显然就是求图 G 的最小权最大匹配， 由于经典的 *KM* 算法是求最大权最大匹配，那么我们再对图 G 进行一定修改，使得 $W_{i,j}=w-W^0_{i,j}\quad\left((i,j)\in A\right)$，且如果$(i,j)\notin A$，则添加边$(i，j)$，$W_{i,j}=0$。其中 w 可以取一个比较大的正整数，但需要满足 $w>\max\left\{W^0_{i,j}\right\}\times n\quad(i,j)\in A^0$。这样用经典的 *KM* 算法求出二分图 G 的最大权最大匹配，即可轻易转化得到最小权最大匹配，从而解决原问题。

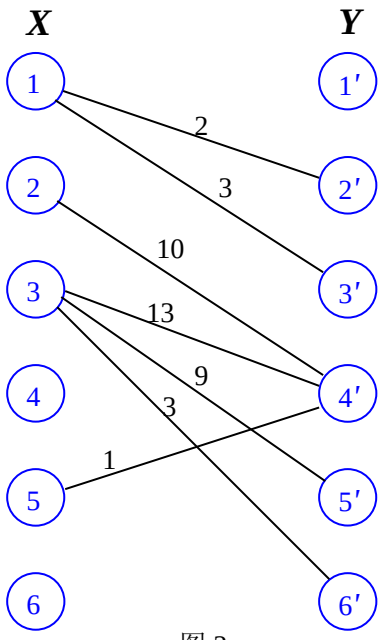

图 3

注：为了使图更简单清晰，省略了边权为无穷大的边。

[2] 如果某条覆盖路径只有一个结点的话，它显然满足性质 I 和性质 III。

2.3 小结

这道题目的数学模型很容易建立，就是最小路径覆盖问题的扩展。在分析该问题的时候抓住每个点在一条覆盖路径中至多有一个前驱一个后继这个条件，可以联系到匹配中每个点也至多和一个点匹配，于是顺利转化成匹配的问题。

一一对应是匹配重要的性质。

三 Roads[3]

3.1 问题描述

一个遥远的王国有 m 条道路连接着 n 个城市。m 条道路中有 n-1 条石头路，m-n+1 条泥土路,任意两个城市之间有且仅有一条完全用石头路连接起来的道路。

每条道路都有一个唯一确定的编号，其中石头路编号为 1..n-1，泥土路编号为 n..m。每条道路都需要一定的维护费，其中第 i 条道路每年需要 C_i 的费用来维护。最近该国国王准备只维护部分道路以节省费用。但是他还是希望人们可以在任两个城市间互达。

国王需要你提交维护每条道路的费用，以便他能让他的大臣来挑选出需要维护的道路，使得维护这些道路的费用是最少的。

尽管国王不知道走石头路和走泥土路的区别，但是对于人民来说石头路似乎是更好的选择。为了人民的利益，你希望维护的道路是石头路。这样你不得不在提交给国王的报告中伪造维护费用。你需要给道路 i 伪造一个费用 D_i，使得这些石头路能够被挑选。为了不让国王发现，你需要使得真实值与伪造值的差值和

$f=\sum_{i=1}^{m}|D_i-C_i|$ 尽量小。

国王的大臣当然不是白痴，全部由石头路组成的方案如果是一种花费最小的方案，那么他会选择这种方案。

求出真实值与伪造值的差值和的最小值，以及得到该最小值的方案，即每条边的修改后的边权 D_i。

[3] Saratov State University 206. Roads

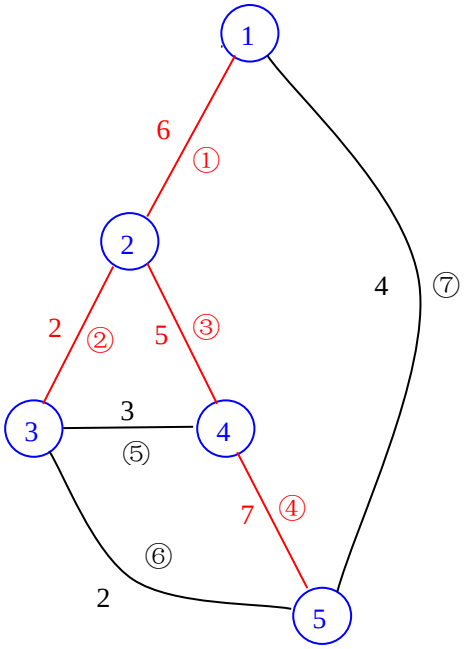

图 4

3.2　问题分析

设原图为 $G^0=(N^0, A^0)$。其中 N^0 表示顶点集合，A^0 表示边集合，即 $N^0=\{1, 2, \cdots\cdots, n\}$，$A^0=\{a_1, a_2, \cdots\cdots, a_m\}$。用 C_i 和 D_i 表示原始及修改后边 a_i 的边权。

由于任意两点都有且仅有一条道路完全是石头路，所以 n-1 条石头路必定是图 G^0 中的一棵生成树，我们设为树 T。而题目则是要求对图中的某些边权进行修改，对于边 a_i，将边权由 C_i 修改成 D_i，使得树 T 成为图中的一棵最小生成树，且 $f=\sum_{i=1}^{m}|D_i-C_i|$ 最小。

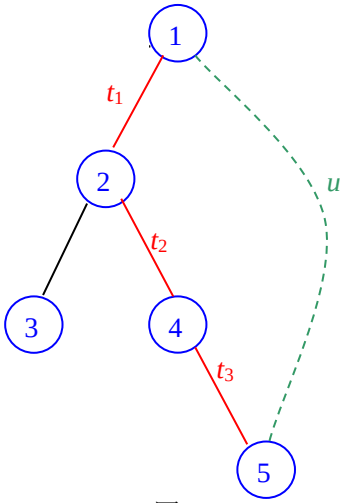

图 5

根据与树 T 的关系，我们可以把图 G^0 中的边分为树边和非树边两类。我们先通过对最小生成树性质的研究来挖掘 D 所需满足的条件。设 $P[e]$表示边 e 的

两个端点之间树的路径中边的集合。如图 5 中，$P[u]=\{t_1,t_2,t_3\}$，即$u\notin T$，而$t_1,t_2,t_3\in T$，且 u 与 t_1, t_2, t_3 构成一个环，所以用非树边 u 替换树边 t_1，t_2，t_3 中任意一条都可以得到一棵新的生成树。而如果 u 的边权比所替换的边的边权更小的话，则可以得到一棵权值更小的生成树。那么只有满足条件$D_{t_1}\le D_u, D_{t_2}\le D_u, D_{t_3}\le D_u$ 才能使得原生成树 T 是一棵最小生成树。

那么推广到一般情况，如果对于边 v，u，其中$v\in P[u], u\notin T$，则必须满足$D_v\le D_u$，否则用边 u 替换边 v 后能得到一棵新的权值更小的生成树$T-v+u$。

我们得到了 D 的限制条件，而问题需要求得$\sum_{i=1}^{m}|D_i-C_i|$最小，其中绝对值符号的存在是一个拦路虎。根据以上的分析，要使树 T 是一棵最小生成树，得到的不等式$D_v\le D_u$中 v 总为树边而 u 总为非树边。也就是树边的边权应该尽量小，而非树边的边权则应该尽量大。

设边权的修改量为Δ，即$\Delta_e=|D_e-C_e|$

I) 当$e\in T$，则应该将边权 C_e 减去一个非负的值，即$\Delta_e=C_e-D_e$

II) 当$e\notin T$，则应该将边权 C_e 加上一个非负的值，即$\Delta_e=D_e-C_e$

这样成功去掉了绝对值符号，只要求得Δ的值，那么 D 值就可以唯一确定，而问题也由求 D 转化成求Δ。

那么任意满足条件 v，u($v\in P[u], u\notin T$)的不等式 $D_v\le D_u$ 等价于$C_v-\Delta_v\le C_u+\Delta$，即$\Delta_v+\Delta_u\ge C_v-C_u$。那问题就是求出所有的$\Delta_i(i\in[1,m])$使其满足这个不等式组且 $f=\sum_{i=1}^{m}\Delta_i$ 最小。

由于不等式$\Delta_v+\Delta_u\ge C_v-C_u$右边$C_v-C_u$是一个已知量，你或许会发现这个不等式似曾相识，这就是我们在求二分图的最佳匹配算法时用到的 *KM* 算法中不可或缺的一个不等式。*KM* 算法中，首先给二分图的每个点都设一个可行顶标，X 结点 i 为 l_i，Y 结点 j 为 r_i。从初始时可行顶标的设定到中间可行顶标的修改直至最后算法结束，对于边权为$W_{v,u}$的边(v，u)始终需要满足$l_v+r_u\ge W_{v,u}$。

于是我们可以构造一个带权的二分图 G=(N，A)，用 W 表示边权，如图 6。

a) 构造两个互补的结点结合 X，Y。把 a_i($a_i\in T$)作为 X_1 结点 i，a_j($a_j\notin T$)

作为 Y_1 结点 j。

b) 如果图 G^0 中 $a_i \in P[a_j], a_j \notin T$，且 $C_i - C_j > 0$，则在 X_1 结点 i 与 Y_1 结点 j 之间添加边$(i，j)$，$W_{i,j} = C_i - C_j$ 。

c) 如果 $|X_1| < |Y_1|$，则添加 $|Y_1| - |X_1|$ 个 X_2 结点，$Y_2 = \varnothing$；如果 $|Y_1| < |X_1|$，则添加 $|X_1| - |Y_1|$ 个 Y_2 结点，$X_2 = \varnothing$。$X = X_1 \cup X_2, Y = Y_1 \cup Y_2, N = X \cup Y$ 。

d) 如果 $i \in X, j \in Y$ 且 $(i, j) \notin A$，则添加边$(i，j)$，且 $W_{i,j} = 0$

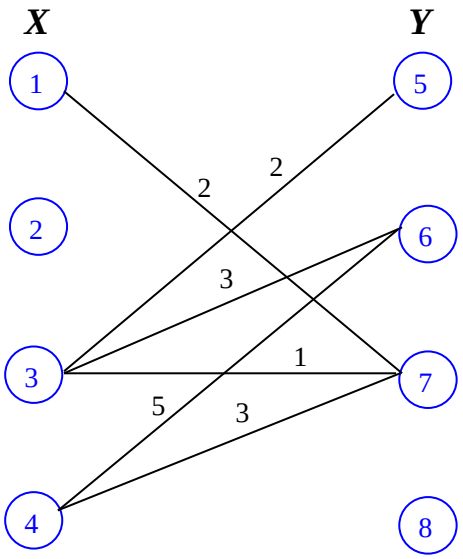

图 6

注：为了使图更简单清晰，省略了边权为 0 的边。

这样图 G 是一个二分完全图，设 Q 为图 G 的一个完备匹配，X 结点 i 的可行顶标为 l_i，Y 结点 j 的可行顶标为 r_j，则：

$$l_i + r_j \geq W_{i,j} \quad ((i, j) \in Q)$$

设 M 为图 G 的最大权匹配，显然 M 也是完备匹配，则满足

$$l_i + r_j = W_{i,j} \quad ((i, j) \in M)$$

设完备匹配 Q 的匹配边的权值和为 S_Q，显然

$$S_M = \sum_{(i,j) \in M} W_{i,j} = \sum_{i \in X} l_i + \sum_{j \in Y} r_j \quad 。$$

显然此时可行顶标之和取到最小值。

若 $i \in X_2$，且 $(i, j) \in M$，

由 $W_{v,u} = 0 \quad ((v,u) \in A, v \in X_2)$ 和 $M \subseteq A$

可知 $W_{i,j}=0$，所以 $l_i+r_j=W_{i,j}=0$

又因为 $l_i\geq 0 \quad (i\in X)$，$X_2\subseteq X$

所以 $l_i=0 \quad (i\in X_2)$，　同理 $r_j=0 \quad (j\in Y_2)$

所以 $S_M=\sum_{i\in X} l_i+\sum_{j\in Y} r_j=\sum_{i\in X_1} l_i+\sum_{j\in Y_1} r_j=\sum_{i=1}^{m}\Delta_i=f$

显然 S_M 即是满足 T 是图 G^0 的一棵最小生成树的最小代价，而此时的 D 值则是修改后的一种可行方案，那么问题就转化为求图 G 的最大权完备 M。

至此，问题已得到解决，我们来分析一下该算法的复杂度。预处理的时间复杂度为 $O(|E|)$，而 *KM* 算法的时间复杂度为 $O(|M||E|)$，所以总的时间复杂度为 $O(|M||E|)$。由于 *KM* 算法是在完备匹配基础上的，所以 $|V|=2\max(n,m)$，$|M|=\frac{|V|}{2}=O(m)$，又由于图 G 是二分完全图，所以 $|E|=|M|^2=O(m^2)$，所以总的时间复杂度为 $O(m^3)$。空间复杂度为 $O(m^2)$。

3.3　扩展思考

如果题目只要求出最少的修改量，而不需要求出修改方案，那么是否有更好的算法呢？

3.3.1　算法 1

现在需要求得的是原问题的一个子问题，利用 *KM* 算法来求出二分图 G 的最大权最大匹配，而 $f_{\min}$ 即为所求，显然可以很好的解决该问题，其时间复杂度为 $O(m^3)$。

3.3.2　算法 2

可以发现，在构造二分图 G 时，其中 *c*)，*d*)两个步骤中构造的都是一些虚结点和虚边，完全是为了符合 *KM* 算法要求完备匹配的条件，没有太多实际的意义。而利用 *KM* 算法解此题最大的优势就在于能求出修改方案，而如果题目不要求修改方案，则毫无意义，因此可试探不添加这些虚结点和虚边。

因为 $f=\sum_{i=1}^{m}\Delta_i=\sum_{i\in X} l_i+\sum_{j\in Y} r_j$，且 $l_i+r_j=W_{ij}\ (i,j)\in M$），所以 $f=\sum_{(i,j)\in M} W_{i,j}$，即最小修改量就是最大权最大匹配的匹配边的权值和，所以问题只需求出最大权最大匹配的值。

构造有向图 $G^f=(N^f, A^f)$，W^f 表示边权，U^f 表示容量，R^f 表示流量，如图 7。

a) 构造两个互补的结点集合 X^f，Y^f。把 $a_i(a_i \in T)$作为 X^f 结点 i，$a_j(a_j \notin T)$作为 Y^f 结点 j。

b) 如果图 G^0 中 $a_i \in P[a_j], a_j \notin T$，且 $C_i - C_j > 0$。则在 X^f 结点 i 与 Y^f 结点 j 之间加入有向边(i, j)，$W_{i,j}^f = C_i - C_j$，$U_{i,j}^f = 1$ 。

c) 构造源点 s 和汇点 t，$N^f = X^f \cup Y^f \cup \{s,t\}$

d) 添加有向边(s, i) $\left(i \in X^f\right)$，$W_{s,i}^f = 0$，$U_{s,i}^f = 1$；添加有向边(j, t) $\left(j \in Y^f\right)$，$W_{j,t}^f = 0$，$U_{j,t}^f = 1$。

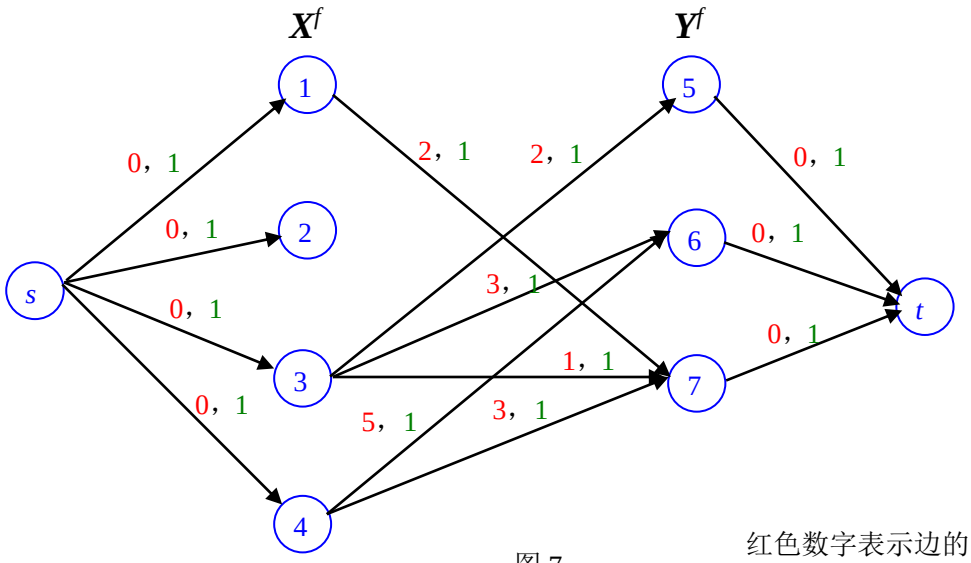

图 7

红色数字表示边的费用
绿色数字表示边的容量

[引理一] 设流 F 的费用和为 $Cost_F$。图 G 上的任意一个完备匹配 M，都能在图 G^f 上找到可行流 F 与其对应，且 $S_M = Cost_F$。而对于图 G^f 上的任意可行流 F，在图 G 上的也都能找到一组以 M 为代表的完备匹配与其对应，且 $Cost_F = S_M$。

[证明] 对于图 G 中任意一条匹配边 $(i, j) \in M$ 且 $W_{i,j} > 0$，都可以找到图 G^f 中一条容量为 1 的流 $s \to i \to j$，其中 $i \in X^f, j \in Y^f$。因为 $W_{s,i}^f = 0$，$W_{j,t}^f = 0$，$W_{i,j} = 0$，所以 $W_{i,j} = W_{s,i}^f + W_{i,j}^f + W_{j,t}^f$；而当图 G 中 $(i, j) \in M$ 且 $W_{i,j} = 0$，则对 S_M 的值不构成影响。所以当流 F 为匹配 M 所对应的流的并时，

$S_M = \sum_{(i,j)\in M} W_{i,j} = \sum_{(i,j)\in F} W^f_{i,j} = Cost_F$。而反过来对于任意可行流 F，同样可以找到完备匹配 M 与其对应且 $Cost_F = S_M$。

设图 G^f 的最大费用最大流为 F，显然图 G^f 的最大费用最大流则和图 G 的最大权最大匹配对应，此时最大费用就等于匹配的权值和，即 $Cost_F = \sum_{(i,j)\in F} W^f_{i,j} \times R^f_{i,j} = \sum_{(i,j)\in M} W_{i,j}$。这样问题就转化为求图 G^f 的最大费用最大流。

如果用 *bellman_ford* 求最大费用路的算法来求最大费用最大流，则复杂度为 $O(|V||E|s)$,其中 s 表示流量。$|V| = O(m),\ |E| = O(nm),\ s = O(n)$，所以复杂度为 $O(n^2m^2)$。

3.3.3　算法 3

由于 $m = O(n^2)$，所以算法二的时间复杂度 $O(n^2m^2)$ 和算法一的 $O(m^3)$ 是一个级别的。观察可发现，用 *bellman_ford* 求最大费用路的复杂度为 $O(|V||E|)$，成为算法 2 的瓶颈，如何减少求最大费用路的代价成为优化算法的关键，但由于残量网络中有负权边，导致类似 *Dijkstra* 等算法没有用武之地。这里介绍一种高效求单源最短路的 *SPFA*（*Shortest Path Faster Algorithm*）算法。

设 L 记录从源点到其余各点当前的最短路径值。*SPFA* 算法采用邻接表存储图，方法是动态优先逼近法。算法中设立一个先进先出的队列用来保存待优化的顶点，优化时每次取出队首结点 p，并且用 p 点的当前路径值 L_p 去优化调整其他顶点的最优路径值 L_j，若有调整，即 L_j 变小了，且 j 点不在当前的队列中，就将 j 点放入队尾以待进一步优化。就这样反复从队列取出点来优化其他点路径值，直至队列空不需要再优化为止。

由于每次优化都是将某个点 j 的最优路径值 L_j 变小，所以算法的执行会使 L 越来越小，所以其优化过程是正确的。只要图中不存在负环，则每个顶点都必定有一个最优路径值，所以随着 L 值逐渐变小，直到其达到最优路径值时，算法结束，所以算法是收敛的，不会形成无限循环。这样，我们简要地证明了该算法的正确性。

算法中每次取出队首结点 p，并访问点 p 的所有邻结点的复杂度为 $O(d)$，其中 d 为点 p 的出度。对于 n 个点 e 条边的图，结点的平均出度为 $\frac{e}{n}$，所以每处理一个点的复杂度为 $O\left(\frac{e}{n}\right)$。设顶点入队的次数为 h，显然 h 随图的不同而不同，但它仅与边的权值分布有关，设 $h = kn$，则算法 *SPFA* 的时间复杂度为：$T = O\left(h \times \frac{e}{n}\right) = O\left(\frac{h}{n} \times e\right) = O(ke)$。经过实际运行效果得到，$k$ 在一般情况下是比

较小的常数（当然也有特殊情况使得 k 值较大），所以 *SPFA* 算法在普通情况下的时间复杂度为 $O(e)$。

而此题中需要求的最大费用路显然可将 *SPFA* 算法稍加修改得到。这样我们得到了一个时间复杂度为 $O(|E|s)$，即 $O(n^2m)$ 的算法。

3.3.4　算法 4

刚才用网络流来求匹配以提高效率，但未对匹配的本质进行深究，现在再回到匹配上来，继续挖掘匹配的性质。前面用 *KM* 算法解此题的时候是构造了一个边上带权的二分图，其实不妨换一种思路，将权值由边上转移到点上，或许会有新的发现。

构造二分图 $G'=(N', A')$，如图 8。

a) 构造两个互补的结点集合 X' 和 Y'，把 $a_i(a_i \in T)$ 作为 X' 结点 i，$a_j(a_j \in A')$ 作为 Y' 结点 j。

b) 在 X' 结点 i 和 Y' 结点 i 之间添加边(i, i)。

c) 如果图 G^0 中 $a_i \in P[a_j], a_j \notin T$，且 $C_i - C_j > 0$。则在 X' 结点 i 与 Y' 结点 j 之间加入有向边(i, j)。

d) 给 Y' 结点 i 一个权值 C_i，即如果某点被匹配则得到其权值。

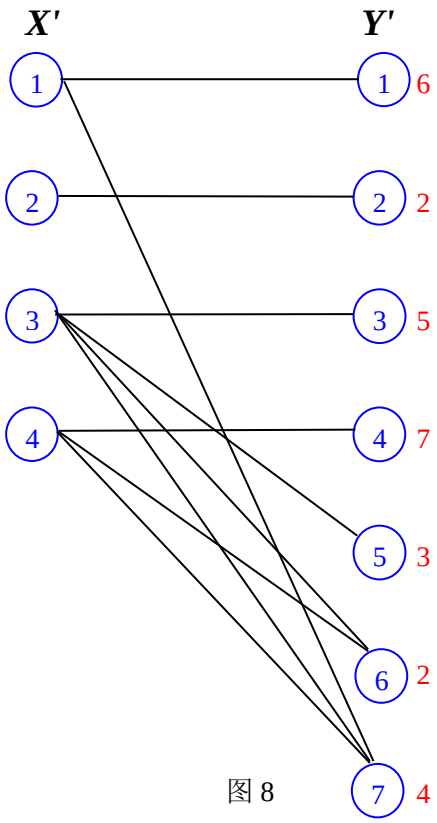

图 8

[引理二] 设 $\mu=\sum_{a_i\in T}C_i$。对于图 G 中一个完备匹配 M，都可以在图 G'中找到一个完备匹配 M'与其对应，且 $S_M=\mu-S_{M'}$。而对于图 G'的任意一个完备匹配 M'，也可以在图 G 中找到一组以 M 为代表的完备匹配与其对应，且 $S_M=\mu-S_{M'}$。

[证明] 设 Y_1' 表示存在 $i(i\neq j)$ 使得 $(i,j)\in M'$ 的点 j 的集合；设 Y_2' 表示存在 i$(i=j)$ 使得 $(i,j)\in M'$ 的点 j 的集合。

对于图 G 中一条匹配边$(i,\ j)$ $(i,j)\in M$且$i\in X_2$，则 $W_{i,j}$=0，对 S_M 的值没有影响。

对于图 G 中一条匹配边$(i,\ j)$ $(i,j)\in M$且$i\in X_1$，

若 $W_{i,j}$=0，则在对应图 G'中可找到一条边$(i,j)\in M'$ $(i\in X',j\in Y_2'$且$i=j)$与其对应；

若 $W_{i,j}$>0，则在对应图 G'中可找到一条边$(i,j)\in M'$ $(i\in X',j\in Y_1')$与其对应。所以

$$\begin{aligned}S_M&=\sum_{(i,j)\in M}W_{i,j}\\&=\sum_{(i,j)\in M,W_{i,j}>0}W_{i,j}\\&=\sum_{(i,j)\in M,W_{i,j}>0}C_i-C_j\\&=\left(\sum_{a_i\in T}C_i-\sum_{j\in Y_2'}C_j\right)-\sum_{j\in Y_1'}C_j\\&=\sum_{a_i\in T}C_i-\left(\sum_{j\in Y_2'}C_j+\sum_{j\in Y_1'}C_j\right)\\&=\mu-S_{M'}\end{aligned}$$

同理，图 G'上的匹配 M'也可以在图 G 上找到对应的匹配 M 且 $S_M=\mu-S_{M'}$。

因为 $S_M+S_{M'}=\mu$ 为定值。显然，当 S_M 取到最大值时，$S_{M'}$取到最小值。又因为 M 和 M'均为完备匹配，所以图 G 的最大权最大匹配就对应了图 G'的最小权完备匹配。那么问题转化为求图 G'的最小权完备匹配。

由于图 G' 中的权值都集中在 Y' 结点上，所以 $S_{M'}$只与 Y' 结点中哪些点

被匹配有关。那么可以使 Y' 结点中权值小的结点尽量被匹配。算法也渐渐明了，将 Y' 结点按照权值大小非降序排列，然后从前往后一个个找匹配。

用 R 来记录可匹配点，如果 X'结点 $i \in R$，则 i 未匹配，或者从某个未匹配的 X'结点有一条可增广路径到达点 i，其路径用 $Path[i]$来表示。设 B_j 表示 j 的邻结点集合，每次查询 Y'结点 j 能否找到匹配时，只需看是否存在点 i，$i \in B_j$且$i \in R$。而每次找到匹配后马上更新 R 和 $Path$。下面给出算法的流程：

Begin
　将 Y'结点按照权值非降序排列；
　$M' \leftarrow 0$；
　计算 R 及 $Path$,并标记点 $i(i \in R)$为可匹配点；
　For　$j \leftarrow 1$　*to m do*
　Begin
　　$q \leftarrow$第 j 个 Y'结点；
　　If　存在 q 的某个邻结点 p 为可匹配点 *then*
　　Begin
　　　将匹配边(p，q)加入匹配 M'；
　　　更新 R 以及 $Path$，并且重新标记点 $i(i \in R)$为可匹配点；
　　End；
　End；
　M' 是一个最小权最大匹配；
End；

下面来分析一下该算法的复杂度。算法中执行了如下操作：
1）将所有 Y'结点按权值非降序排列；
2）询问是否存在 q 的某个邻结点 p 为可匹配点；
3）更新 M'；
4）更新 R 以及 $Path$；
5）标记点 $i(i \in R)$为可匹配点；

操作 1 的时间复杂度为 $O(m\log_2 m)=O(n^2\log_2 n)$。设 $d_{\max}=\max\{|B_j|\}\quad j\in[1,m]$，操作 2 单次执行的复杂度为$O(|B_j|)$，最多执行 m 次，所以复杂度为$O(md_{\max})=O(n^2 d_{\max})$。操作 3 单次执行的复杂度为$O(1)$，最多执行 n-1 次，所以复杂度为$O(n)$。操作 5 单次执行的复杂度为$O(n)$，最多执行 n-1 次，所以复杂度为$O(n^2)$。

接下来讨论操作 4 的复杂度。我们知道，如果某个点在某次更新中是不可匹配点，那么以后无论怎么更新它都不可能变成可匹配点。又如果某个点为可匹配点，则它的路径必然为$i_0 \to j_1 \to i_1 \to j_2 \to i_2 \to \cdots \to j_k \to i_k \quad (k \ge 0)$，其中 i_0 为未匹配点而且$(j_t, i_t)\left(t \in [1,k]\right)$是当前的匹配边，所以 *Y'*结点中未匹配点是不可能出现在某个 *X'*点 *i* 的 *Path*[*i*]中的。也就是说我们在更新 *R* 和 *Path* 的时候只需要在 *X'*结点中原来的可匹配点以及 *Y'*结点中已匹配点和它们之间的边构成的一个子二分图中进行，显然任意时刻图 *G'*的匹配边数都是不超过 *n*-1 的，所以该子图的点数是$O(n)$的，边数是$O(nd_{\max})$，显然单次执行操作 4 的复杂度即为$O(nd_{\max})$，最多执行 *n* 次，所以其复杂度为$O(n^2 d_{\max})$。

算法总的时间复杂度为$O(n^2 \log_2 n)+O(n^2 d_{\max})+O(n)+O(n^2 d_{\max})+O(n^2)$
$=O(n^2(d_{\max}+\log_2 n))$，因为 d_{max} 是$O(n)$级别的，所以该算法的时间复杂度为$O(n^3)$，其空间复杂度为$O(nm)$。

其实本题还有更优秀的算法，但其推导与证明相对比较复杂，这里就不详细介绍了，大家可以参考相关论文。

3.4 小结

该题是一道最优化的问题，尝试发现动态规划，贪心等算法都无从下手。但经过一步步的分析，思路渐渐清晰，在得到了若干重要不等式后，问题豁然开朗，是一道求最佳匹配的问题，可以用经典的 *KM* 算法求解。

而在对不求方案的问题的研究中，不局限于直观的原图，而是从各个方向各个角度入手，构造不同的图，以展现题目各个方面的性质。也将算法复杂度由$O(m^3)$，优化到$O(n^2 m)$，再到$O(n^3)$，使效率大大提高。

下表是对上文研究的几个算法的简单比较：

	时间复杂度	空间复杂度	算法核心
算法一	$O(m^3)$	$O(m^2)$	*KM* 算法求最大权最大匹配
算法二	$O(n^2 m^2)$	$O(nm)$	用 *bellman_ford* 算法求网络最大费用最大流增广路
算法三	$O(n^2 m)$	$O(nm)$	用 *SPFA* 算法求网络最大费用最大流增广路
算法四	$O(n^3)$	$O(nm)$	求结点带权的二分图的最小权匹配

四　总结

通过对以上的例题的分析可见，在信息学竞赛中，二分图匹配算法的应用往往不是显而易见的，而是需要挖掘出问题的本质，从而构造合适的二分图并用匹配算法来求解。而在求匹配的时候往往也不是简单的套用经典算法，而是需要充分利用题目的特有性质，将经典匹配算法加以变形，从而得到更高效的算法。

信息学竞赛中的各种题目，往往都需要通过对题目的仔细**观察**，构造出合适的数学模型，然后通过对题目以及模型的进一步**分析**，挖掘出问题的本质，进行大胆的**猜想**，转化模型，设计合适的算法解决问题。

[感谢]

衷心感谢向期中老师在学习上对我的指导和帮助
衷心感谢任恺、胡伟栋和周源同学对我的大力帮助
衷心感谢肖湘宁和周戈林两位同学对我的论文提出宝贵的意见

[参考文献]

[1] ***Ravindra K. Ahuja & James B. Orlin ，A Faster Algorithm for the Inverse Spanning Tree Problem***
[2] **段凡丁，关于最短路径的*SPFA*快速算法**

[附录]

例一原题：

Saratov State University 252. Railway Communication

time limit per test: 1 sec.
memory limit per test: 65536 KB
input: standard
output: standard

There are N towns in the country, connected with M railroads. All railroads are one-way, the railroad system is organized in such a way that there are no cycles. It's necessary to choose the best trains schedule, taking into account some facts.
Train path is the sequence of towns passed by the train. The following conditions must be satisfied.
1) At most one train path can pass along each railroad.
2) At most one train path can pass through each town, because no town can cope with a large amount of transport.

3) At least one train path must pass through each town, or town economics falls into stagnation.
4) The number of paths must be minimal possible.
Moreover, every railroad requires some money for support, i-th railroad requires c[i] coins per year to keep it intact. That is why the president of the country decided to choose such schedule that the sum of costs of maintaining the railroads used in it is minimal possible. Of course, you are to find such schedule.

Input
The first line of input contains two integers N and M (1<=N<=100; 0<=M<=1000). Next M lines describe railroads. Each line contains three integer numbers a[i], b[i] and c[i] - the towns that the railroad connects (1<=a[i]<=N; 1<=b[i]<=N, a[i]<>b[i]) and the cost of maintaining it (0<=c[i]<=1000). Since the road is one-way, the trains are only allowed to move along it from town a[i] to town b[i]. Any two towns are connected by at most one railroad.

Output
On the first line output K - the number of paths in the best schedule and C - the sum of costs of maintaining the railroads in the best schedule.
After that output K lines, for each train path first output L[i] (1<=L[i]<=N) - the number of towns this train path uses, and then L[i] integers identifying the towns on the train path. If there are several optimal solutions output any of them.

Sample test(s)

Input
```
4 4
1 2 1
1 3 2
3 4 2
2 4 2
```

Output
```
2 3
2 1 2
2 3 4
```

例二原题：

Saratov State University 206. Roads

time limit per test: 2 sec.
memory limit per test: 65536 KB
input: standard
output: standard

The kingdom of Farland has N cities connected by M roads. Some roads are paved with stones, others are just country roads. Since paving the road is quite expensive, the roads to be paved were chosen in such a way that for any two cities there is exactly one way to get from one city to another passing only the stoned roads.

The kingdom has a very strong bureaucracy so each road has its own ordinal number ranging from 1 to M: the stoned roads have numbers from 1 to N-1 and other roads have numbers from N to M. Each road requires some money for support, i-th road requires c_i coins per year to keep it intact. Recently the king has decided to save some money and keep financing only some roads. Since he wants his people to be able to get from any city to any other, he decided to keep supporting some roads in such a way, that there is still a path between any two cities.

It might seem to you that keeping the stoned roads would be the good idea, however the king did not think so. Since he did not like to travel, he did not know the difference between traveling by a stoned road and travelling by a muddy road. Thus he ordered you to bring him the costs of maintaining the roads so that he could order his wizard to choose the roads to keep in such a way that the total cost of maintaining them would be minimal.

Being the minister of communications of Farland, you want to help your people to keep the stoned roads. To do this you want to fake the costs of maintaining the roads in your report to the king. That is, you want to provide for each road the fake cost of its maintaining d_i in such a way, that stoned roads form the set of roads the king would keep. However, to lower the chance of being caught, you want the value of sum(i = 1..M, $|c_i-d_i|$) to be as small as possible.

You know that the king's wizard is not a complete fool, so if there is the way to choose the minimal set of roads to be the set of the stoned roads, he would do it, so ties are allowed.

Input
The first line of the input file contains N and M ($2 \le N \le 60$, $N-1 \le M$

≤ 400). Next M lines contain three integer numbers a_i, b_i and c_i each — the numbers of the cities the road connects ($1 \le a_i \le N$, $1 \le b_i \le N$, $a_i \ne b_i$) and the cost of maintaining it ($1 \le c_i \le 10\ 000$).

Output

Output M lines — for each road output d_i that should be reported to be its maintainance cost so that he king would choose first N-1 roads to be the roads to keep and the specified sum is minimal possible.

Sample test(s)

Input

```
4 5
4 1 7
2 1 5
3 4 4
4 2 5
1 3 1
```

Output

```
4
5
4
5
4
```