

浅析二分图匹配

在信息学竞赛中的应用


长郡中学 王俊

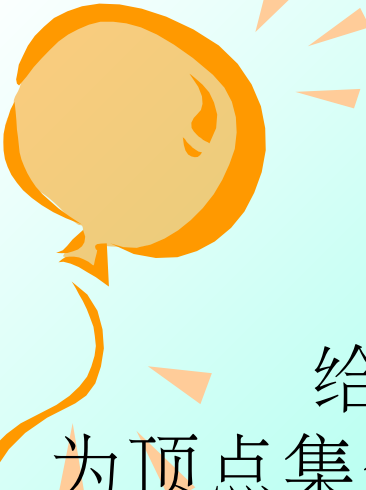


引言

二分图匹配是一类经典的图论算法，在近年来信息学竞赛中有广泛的应用。

二分图和匹配的基础知识已经在前辈的集训队论文中有过介绍，本文主要通过一道例题研究其应用。






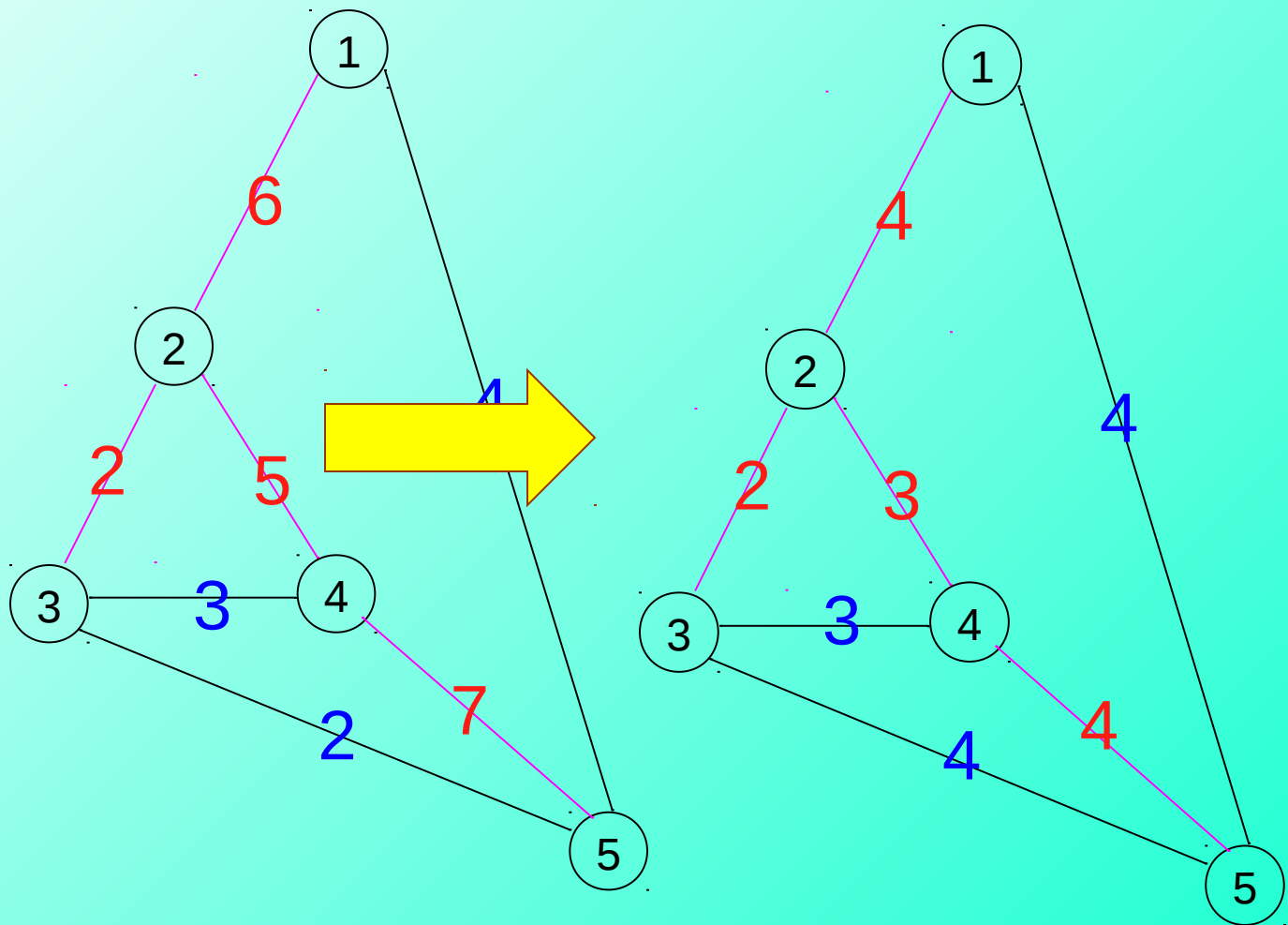
[例题] *Roads*

给定一个无向图 $G^0 = (V^0, E^0, C)$ ， V^0 为顶点集合， E^0 为边集合（无重边）， C 为边权（非负整数）。设 $n = |V^0|$ ， $m = |E^0|$ ， E^0 中前 $n-1$ 条边构成一棵生成树 T 。请将边权进行如下修改，即对于 $e \in E$ ，把 C_e 修改成 D_e （ D_e 也为非负整数），使得树 T 成为图 G 的一棵最小生成树。修改的代价定义为：

$$f = \sum_{e \in E} |C_e - D_e|$$

请求出修改的最小代价。






$$f = |6-4| + |2-2| + |5-3| + |7-4| + |3-3| + |2-4| + |4-4| = 9$$



初步分析

- 根据与树 T 的关系，我们可以把图 G^0 中的边分成树边与非树边两类。
 - 设 P_e 表示边 e 的两个端点之间的树的路径中边的集合。
- 

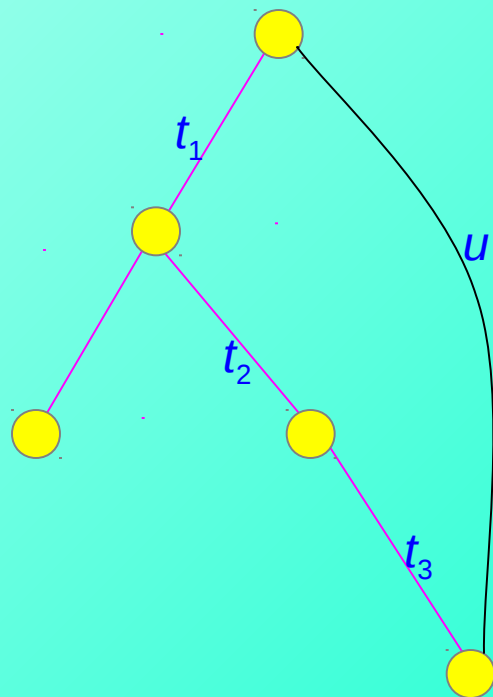
初步分析

● 那么用非树边 u 代替树边 t_1, t_2, t_3 中的任意一条都可以得到一棵新的生成树，所以

$P = \{t_1, t_2, t_3\}$ 。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。

● 那么要使原生成树 T 是一棵最小生成树，必须满足条件

$$D_{t_1} \leq D_u ; \quad D_{t_2} \leq D_u ; \quad D_{t_3} \leq D_u$$

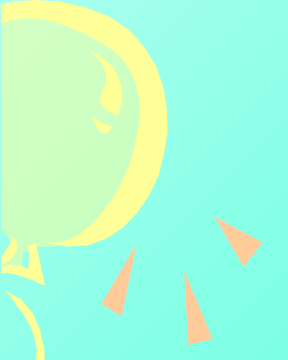




初步分析

对边 v , u 如果满足条件 $u \in T$,
 $v \in P_u$, 则称 u 可替换 v 。

如果边 v , $u(u$ 可替换 $v)$, 则必须满足 $D_v \leq D_u$, 否则用 u 替换 v 可得到一棵权值更小的生成树 $T-v+u$ 。





初步分析

不等式 $D_v \leq D_u$ 中 v 总为树边，而 u 总为非树边。

那么显然树边的边权应该减小（或不变），而非树边的边权则应该增大（或不变）。设边权的修改量为 Δ ，即

$$\Delta_e = |D_e - C_e|$$

当 $e \in T$ ， $\Delta_e = C_e - D_e$ ，即 $D_e = C_e -$

Δ_e 当 $e \notin T$ ， $\Delta_e = D_e - C_e$ ，即 $D_e = C_e +$

Δ_e





初步分析

那么当 u 可替换 v 时，由不等式

$$D_v \leq D_u$$



$$C_v - \Delta_v \leq C_u + \Delta_u$$




$$\Delta_v + \Delta_u \geq C_v - C_u$$

那问题就是求出所有的 Δ 使其满足以上不等式且：

$$f = \sum_{i=1}^m \Delta_i$$



最小。




观察此不等式

大家或许会发现这个不等式似曾相识！

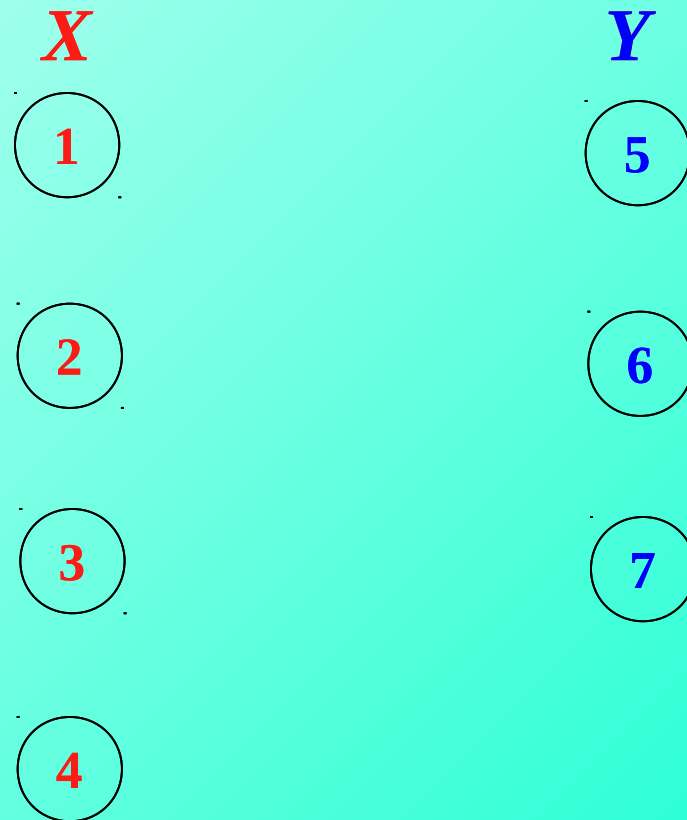
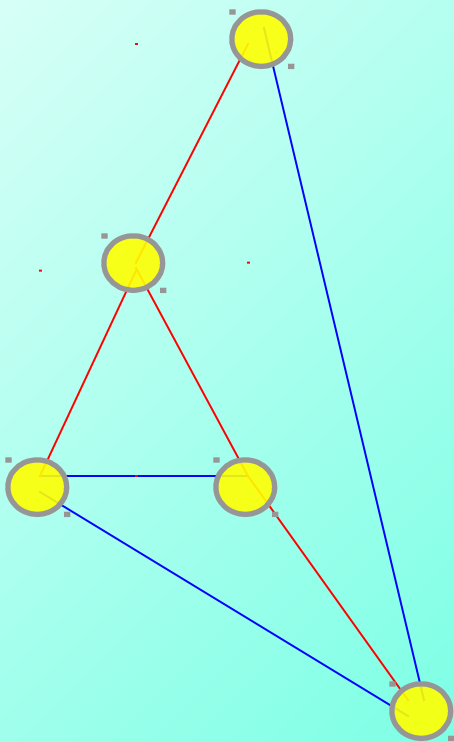
$$\Delta_v + \Delta_u \geq C_v - C_u$$

这就是在求二分图最佳匹配的经典 **KM** 算法中不可或缺的一个不等式。其中不等式右侧 C_v 是一个已知量！

KM 算法中，首先给二分图的每个顶点都设一个可行顶标，**X** 结点 i 为 l_i ，**Y** 结点 j 为 r_j 。从始至终，边权为 $W_{v,u}$ 的边 (v,u) 都需要满足 $l_v + r_u \geq W_{v,u}$ 。



我们来构造二分图 G



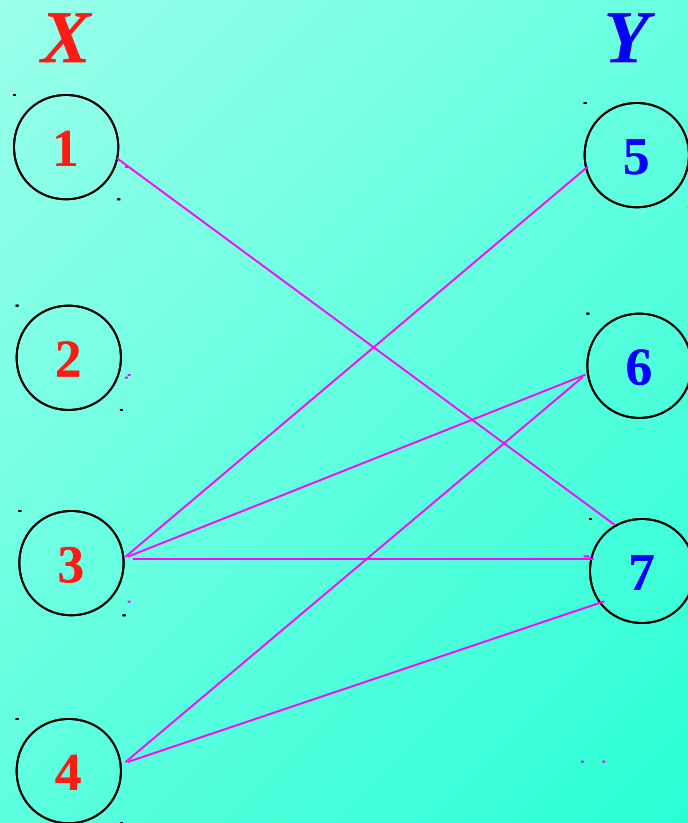
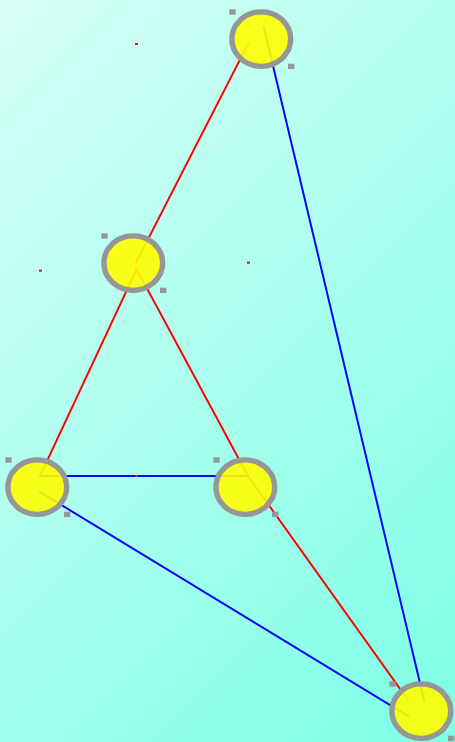
建立两个互补的结点集合

X 结点 i 表示图 G^0 中树边

Y 结点 j 表示图 G^0 中非树边 $a_j (a_j \in T)$ 。

设这些结点均为实点

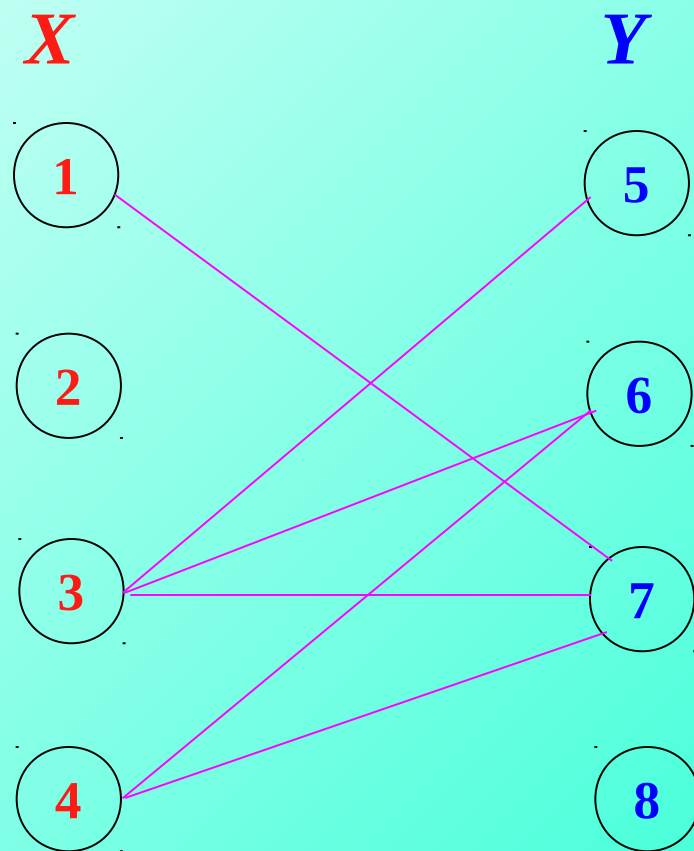
构造二分图 G



如果图 G^0 中, a_j 可替换 a_i , 且 $C_i - C_j > 0$, 则在 X 结点 i 和 Y 结点 j 之间添加边 (i,j) , 边权 $W_{i,j} = C_i - C_j$ 。

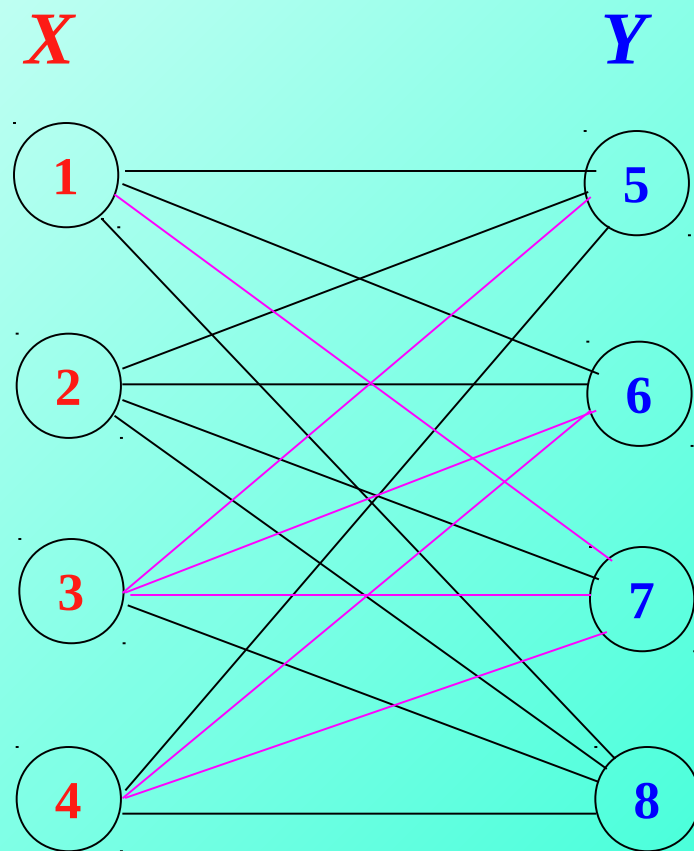
▶ 设这些边均为实边。

构造二分图 G



在结点数少的一侧添加虚结点，使得 X 结点和 Y 结点的数目相等。

构造二分图 G



如果 X 结点 i 和 Y 结点 j 之间没有边，则添加一条权值为 0 的虚边 (i,j) 。



算法分析

对于图 G 的任意一个完备匹配 X ，都有

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

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 X 的所有匹配边的权值和为 S_X ，则

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



显然，此时的可行顶标之和取到最小值。

算法分析

因为虚结点 X_i 的匹配边肯定是权值为 0 的虚边，所以 $l_i=0$ 。同理对于虚结点 Y_j ， $r_j=0$ 。

$$S_M = \sum_{i \in X} l_i + \sum_{j \in Y} r_j = \sum_{i \in X \text{ 且为实点}} l_i + \sum_{j \in Y \text{ 且为实点}} r_j = \sum_{i=1}^m \Delta_i = f$$

问题解决

显然， S_M 即是满足树 T 是图 G^0 的一棵最小生成树的最小代价。那么问题就转化为求图 G 的最大权完备匹配 M ，即可用 **KM** 算法求解。



复杂度分析

我们来分析一下该算法的时间复杂度。


- 预处理的时间复杂度为 $O(|E|)$
- KM 算法的时间复杂度为 $O(|V||E|)$

由于图 G 是二分完全图。

$$|V|=2\max\{n-1, m-n+1\}=O(m)$$

$$|E|=|V|^2=O(m^2)$$

所以算法总时间复杂度为 $O(m^3)$ 。



思考



用 ***KM*** 算法解此题在构图时添加了许多虚结点和虚边，但其并没有太多实际意义。


那么，算法中是否存在大量冗余呢？还有没有优化的余地呢？



算法分析

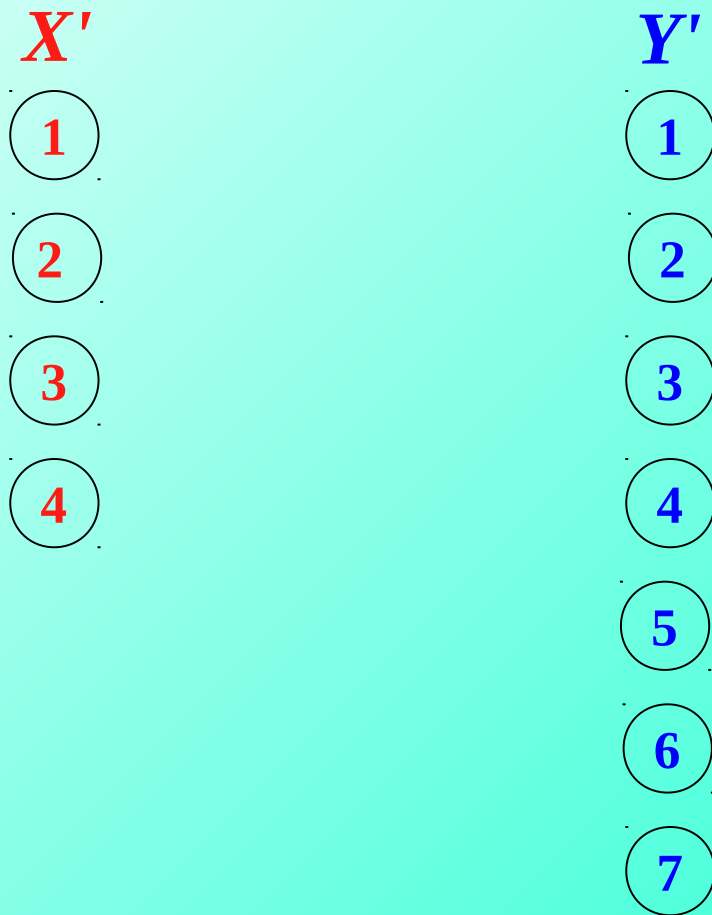
答案是肯定的，如果不添加这些虚结点和虚边，可以得到更好的算法。

下面就介绍一种更优秀的**匹配** 算法！



前面用 **KM** 算法解此题时构造了一个边上带有权值的二分图。其实不妨换一种思路，将权值由**边**转移到**点**上，或许会有新的发现。

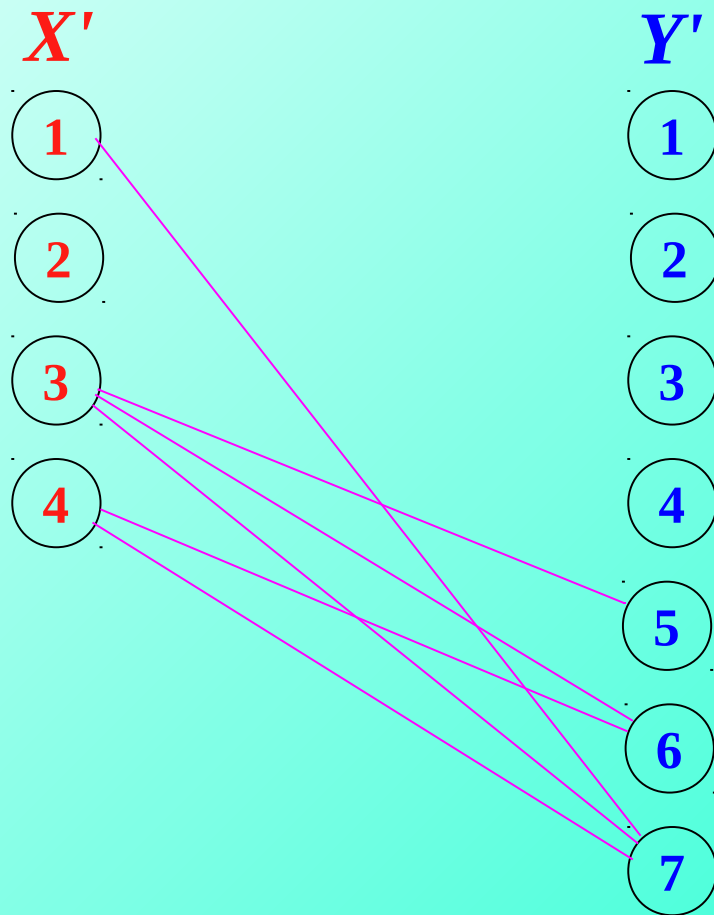
构造二分图 G'



同样建立两个互补的结点集合 X' , Y' 。

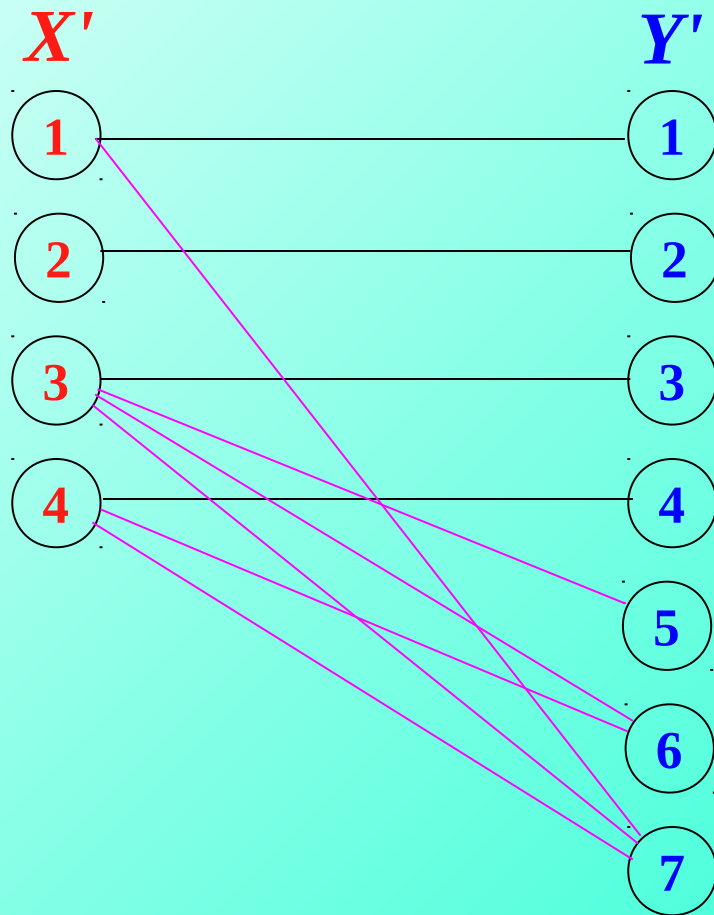
X' 结点 i 表示树边 $a_i(a_i \in T)$, Y' 结点 j 表示任意边 $a_j(a_j \in V^0)$ 。

构造二分图 G'



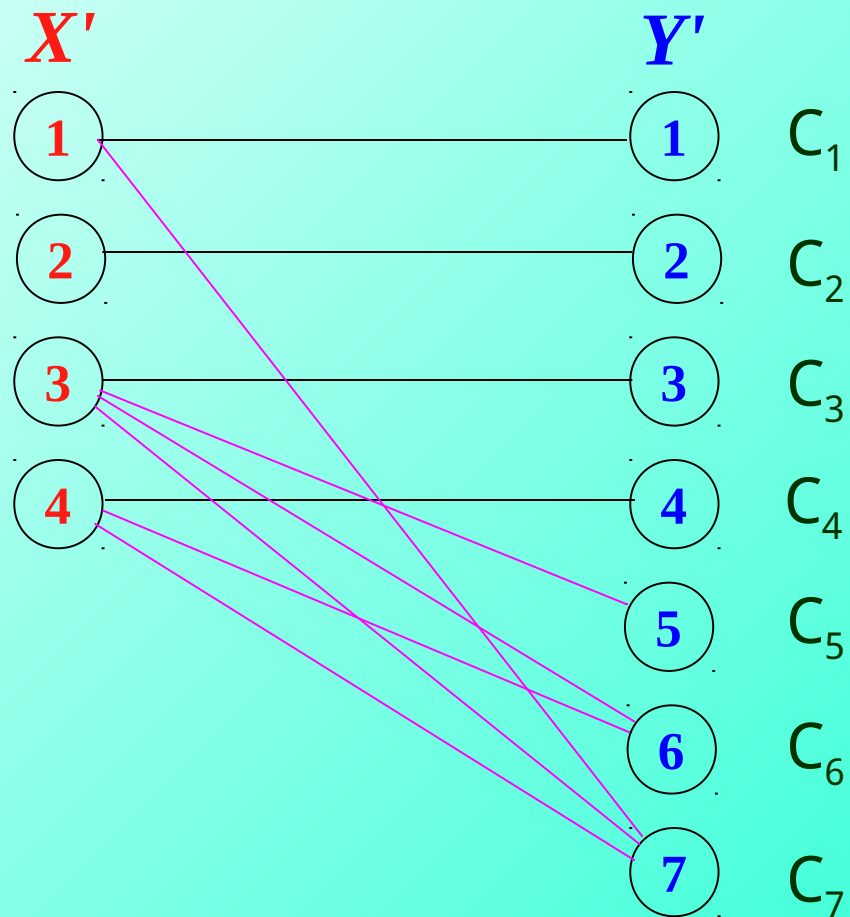
如果图 G^0 中, a_j 可替换 a_i , 且 $C_i - C_j > 0$, 则在 X' 结点 i 和 Y' 结点 j 之间添加边 (i,j) 。

构造二分图 G'



在 X' 结点 i 和 Y' 结点 i 之间添加边 (i,i) 。

构造二分图 G'




给每个 Y' 结点 i 一个权值 C_i 。如果点 i 被匹配则得到权值 C_i ，否则得到权值 0 。



算法分析

设 $\mu = \sum_{a_i \in T} C_i$

[引理] 对于图 G 中的任何一个完备匹配 M ，都可以在图 G' 中找到一个唯一的完备匹配 M' 与其对应，且 $S_M = \mu - S_{M'}$ 。对于图 G' 中的任何一个完备匹配 M' ，同样可以在图 G 中找到一组以 M 为代表的匹配与其对应，且 $S_M = \mu - S_{M'}$ 。





证明引理

这里将介绍如何找到图 G 中匹配 M 对应的图 G' 中匹配 M' 。

对于图 G 中虚结点 X_i 的匹配边 $(i, j) \in M$ ，显然有 $W_{ij}=0$ ，对 S_M 值没有影响。

对于图 G 中实结点 X_i 的匹配边 $(i, j) \in M$ ，

若 $W_{ij}>0$ ，则对应图 G' 中的一条匹配边 (i, j)

若 $W_{ij}=0$ ，则对应图 G' 中的一条匹配边 (i, i)



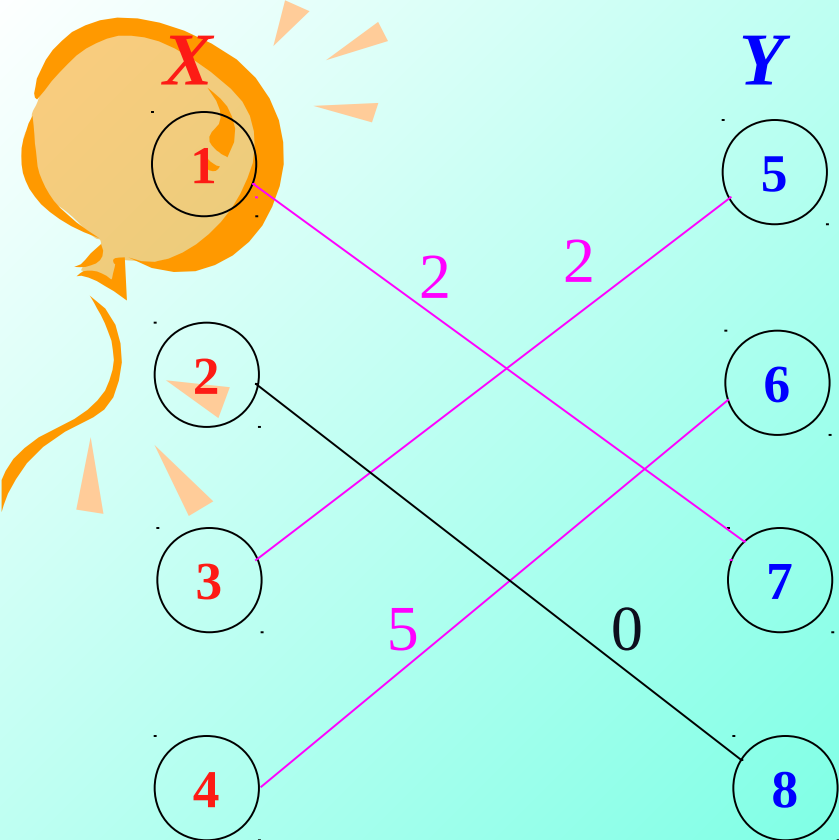


图 G

边权为 **2** 的匹配边 (1,7)
 边权为 **0** 的匹配边 (2,8)
 边权为 **2** 的匹配边 (3,5)
 边权为 **5** 的匹配边 (4,6)

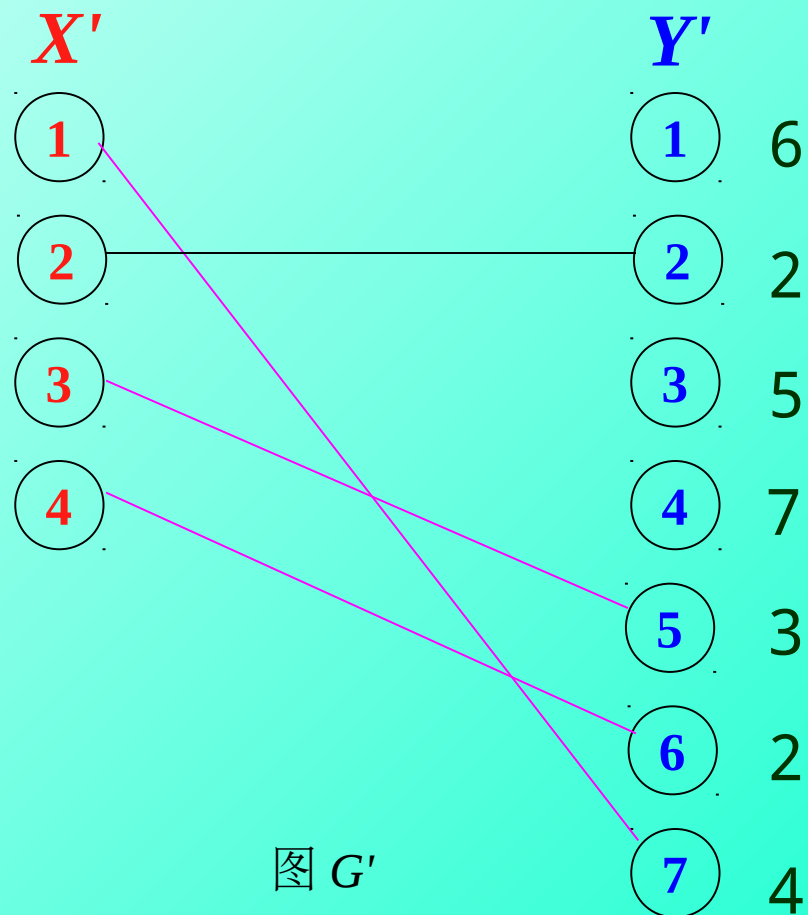


图 G'

有匹配边 (1,7) 与其对应
 有匹配边 (2,2) 与其对应
 有匹配边 (3,5) 与其对应
 有匹配边 (4,6) 与其对应

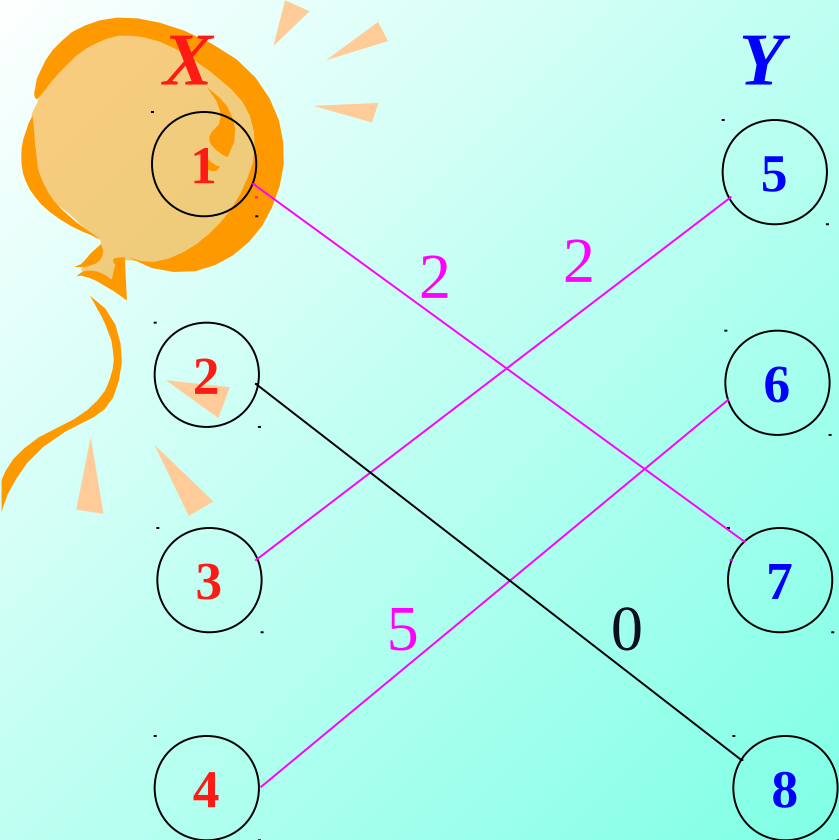


图 G

图 G 中这个完备匹配 M 为：
 $(1,7), (2,8), (3,5), (4,6)$

$$S_M = 2 + 0 + 2 + 5 = 9$$

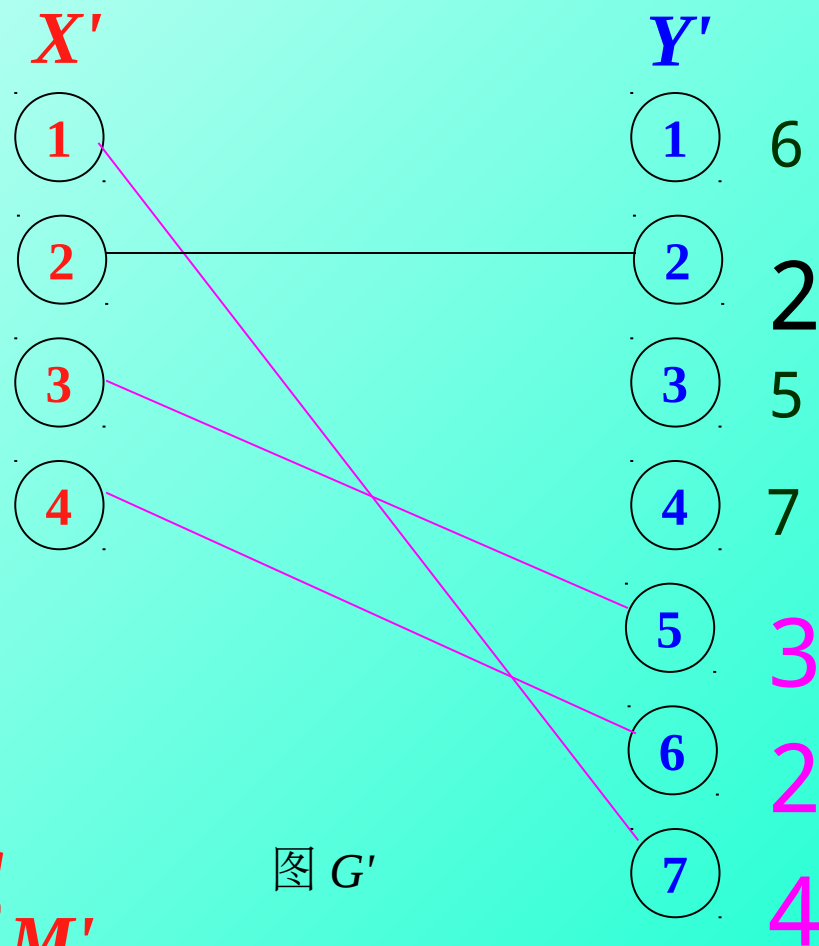


图 G'

图 G' 中对应的完备匹配 M' 为：
 $(1,7), (2,2), (3,5), (4,6)$

$$S_{M'} = 4 + 2 + 3 + 2 = 11$$

$$\mu = 6 + 2 + 5 + 7 = 20$$

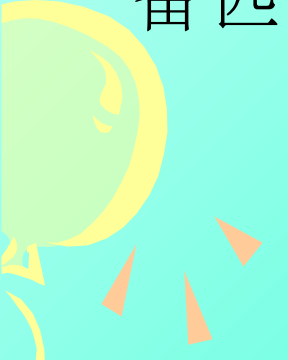
$$S_M = \mu - S_{M'}$$



算法分析

因为 $S_M + S_{M'} = \mu$ 。所以当 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' 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。






算法分析

由于图 G' 的权值都集中在 Y' 结点上，所以 $S_{M'}$ 的值只与 Y' 结点中哪些被匹配到有关。

那么可以将所有的 Y' 结点按照权值大小非降序排列，然后每个 X' 结点都尽量找到权值较小的 Y' 结点匹配。






算法分析

用 R 来记录可匹配点，如果 X' 结点 $i \in R$ ，则表示 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 Y' 结点 j 的邻结点集合， Y' 结点 j 能找到匹配当且仅当存在点 i ， $i \in B_j$ 且 $i \in R$ 。



下面给出算法的流程:

将 Y' 结点非降序排列

初始化 M' , P 和 $Path$

$j \leftarrow 1$

$q \leftarrow Y'$ 的第 j 个结点

Y

存在 q 的某个邻结点 p 为可匹配点

N

更新 M' , R 和 $Path$

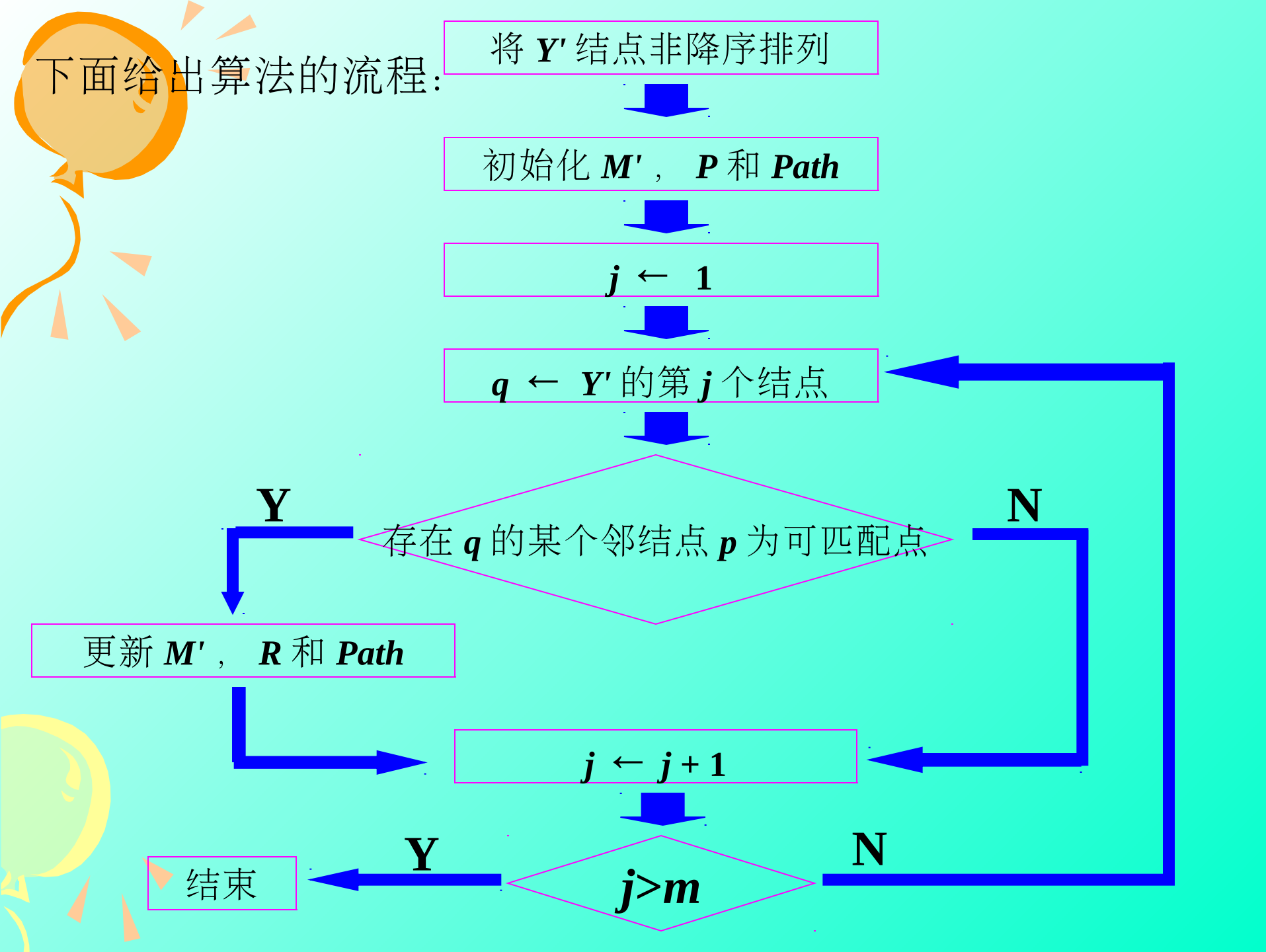
$j \leftarrow j + 1$

Y

结束

N

$j > m$



复杂度分析

下面来分析一下该算法的时间复杂度。

算法中执行了如下操作：

1> 将所有 Y' 结点按权值大小非降序排列；

$$O(m \log_2 m) = O(n^2 \log_2 n)$$

2> 询问是否存在 q 的某个邻结点 p 为可匹配点；

3> ~~$O(mn)$~~ $O(n^3)$;
更新 M ;

$$O(n)$$

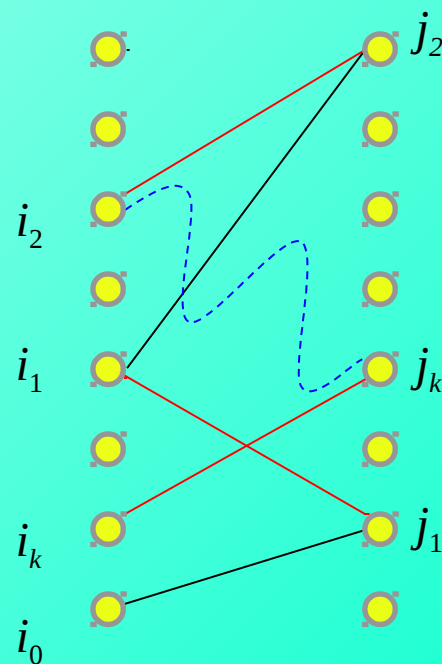
4> 更新 R 以及 $Path$;

$$O(n^3)$$

复杂度分析

前三个操作复杂度都显而易见，下面讨论操作 **4** 的时间复杂度。

如果某个点为可匹配点，则它的路径必然为
 $i_0 \rightarrow j_1 \rightarrow i_1 \rightarrow j_2 \rightarrow i_2 \rightarrow \dots \rightarrow j_k \rightarrow i_k$
($k \geq 0$), 其中 i_0 为未匹配点而且
 $(j_t, i_t)(t \in [1, k])$ 为匹配边。






复杂度分析

所以 Y' 结点中的未匹配点是不可能出现在某个 X' 结点 i 的 $Path_i$ 中的。

也就是说我们在更新 R 和 $Path$ 时只需要处理 X' 结点和已匹配的 Y' 结点以及它们之间的边构成的子二分图。



显然任意时刻图 G' 的匹配边数都不超过 $n-1$ ，所以该子图的点数为 $O(n)$ ，边数为 $O(n^2)$ 。所以该操作执行一次的复杂度即为 $O(n^2)$ ，最多执行 n 次，所以其复杂度为 $O(n^3)$ 。

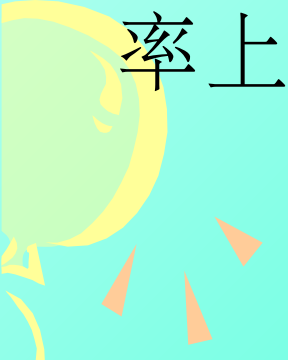


复杂度分析

那么算法总的时间复杂度为：


$$O(n^2 \log_2 n) + O(n^3) + O(n) + O(n^3) = O(n^3)$$

因为 $O(m) = O(n^2)$ ，所以该算法相对于算法一 $O(m^3) = O(n^6)$ 的复杂度，在效率上有了巨大的飞跃。





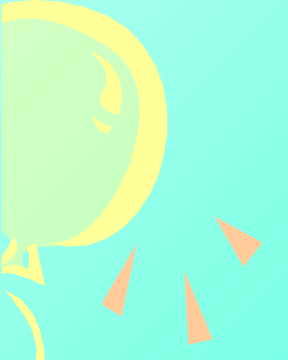
回顾

- 通过对最小生成树性质的分析得到一组不等式 $D_v \leq D_u$ 。
 - 将不等式变形后，通过对其观察，联想到了解决二分图最佳匹配经典的 **KM** 算法，即得到了算法一。
 - 正是通过猜想将权值由图中的边转移到顶点上，重新构造二分图，才得到了更为优秀的算法二！
- 



总结

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，设计优秀的算法解决问题。





结语

仔细观察

认真分析

大胆猜想



