Matrix 解题报告

李晓潇

福州第一中学

高三五班

lixx1991@sina.com

目录

题目大意

给出一个 $N*N$ 的非负矩阵 $B=(b_{ij})$，和一个 $1*N$ 的非负矩阵 $C=(c_{ij})$。$A=(a_{ij})$ 是一个 $1*N$ 的 0，1 矩阵，令矩阵 $D=(d_{ij})=(A*B-C)*A^T$，根据矩阵运算的规则可知，D 是一个 1*1 的矩阵。要求构造 A 矩阵最大化 D 矩阵的元素值，输出得到的 D 矩阵。

问题分析

算法一

在乍看之下没有很好想法的情况下，搜索往往就成为一种万能的解法。看到题目 30% 的数据 $N \le 20$，可以采用搜索解决。

搜索的方法比较简单，直接 *0*，*1* 枚举矩阵 A 的每一位，并计算矩阵 D 更新最优解。但是即使是如此简单的搜索也要注意在常数项上的优化。假如我们每次搜索出一个 A 矩阵都重新计算 $(A*B-C)*A^T$，那么计算次数将会达到 2^N*N^2，对于 $N=20$ 的数据难以通过，所以优化势在必行。

当 $N=5$ 时，我们观察搜索过程中可能出现的两种状态：

$$A_1\begin{pmatrix}1 & 0 & 1 & 1 & 0\end{pmatrix}$$

$$A_2\begin{pmatrix}1 & 0 & 1 & 1 & 1\end{pmatrix}$$

我们发现这两个矩阵的前 4 位都是相同的，可是在最后的计算的时候，我们对前缀重复计算。为了避免冗余的计算，我们采用一边搜索一边计算的策略，即我们在搜索的过程中，每确定一位 A 矩阵的元素，同时计算 $A*B$ 的值。

令矩阵 $E=A*B$，搜索过程为代码如下：

SEARCH(I)

if ($i > N$) **then**

$$ans \leftarrow \max(ans, (E-C)*A^T)$$

else

$a[i] \leftarrow 0$

SEARCH (i+1);

$a[i] \leftarrow 1$

for $j \leftarrow 1$ to N

$e[j] \leftarrow e[j] + b[i][j]$

SEARCH (i+1);

for $j \leftarrow 1$ to N

$e[j] \leftarrow e[j] - b[i][j]$

经过优化，避免对相同前缀的重复计算，大大提高效率，可以轻松的通过 30%的数据。

时间复杂度 ： $O(2^N N^2)$

空间复杂度： $O(N^2)$

期望得分： 20～30 分

算法二

我们尝试从计算矩阵 D 的公式寻找本题的突破点，试着对这个式子进行化简：

$$D = (A * B - C) * A^T$$

根据矩阵乘法的分配律不难得到：

$$D = A * B * A^T - C * A^T$$

我们把减号前后两个部分拆开分别计算：

$$A * B * A^T$$

$$= (a_1 \quad a_2 \quad \dots \quad a_N) * \begin{pmatrix} b_{11} & \cdots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NN} \end{pmatrix} * \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}$$

$$= \left(\sum_{i=1}^{N} a_i * b_{i1} \quad \sum_{i=1}^{N} a_i * b_{i2} \quad \dots \quad \sum_{i=1}^{N} a_i * b_{iN} \right) * \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}$$

$$= \left(\sum_{\substack{i=1 \\ a_i=1}}^{N} \sum_{\substack{j=1 \\ a_j=1}}^{N} a_i * b_{ij} * a_j \right)$$

由于 A 矩阵是一个 0，1 矩阵，因此上面这个结果也可以表示为：

$$\left(\sum_{\substack{i=1 \\ a_i=1}}^{N} \sum_{\substack{j=1 \\ a_j=1}}^{N} b_{ij} \right)$$

我们再来算后半段，我们同样将其写成矩阵的形式：

$$C * A^T$$

$$= (c_1 \quad c_2 \quad \dots \quad c_N) * \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}$$

$$= \left(\sum_{\substack{i=1 \\ a_i=1}}^{N} c_i * a_i \right)$$

同样因为 A 矩阵是一个 0，1 矩阵，上式还可以写做：

$$\left(\sum_{\substack{i=1 \\ a_i=1}}^{N} c_i \right)$$

我们将前后两部分合并，则有：

$$D = \left(\sum_{\substack{i=1 \\ a_i=1}}^{N} \sum_{\substack{j=1 \\ a_j=1}}^{N} b_{ij} \right) - \left(\sum_{\substack{i=1 \\ a_i=1}}^{N} c_i \right)$$

$$D = \left(\sum_{\substack{i=1 \\ a_i=1}}^{N} \sum_{\substack{j=1 \\ a_j=1}}^{N} b_{ij} - \sum_{\substack{i=1 \\ a_i=1}}^{N} c_i \right)$$

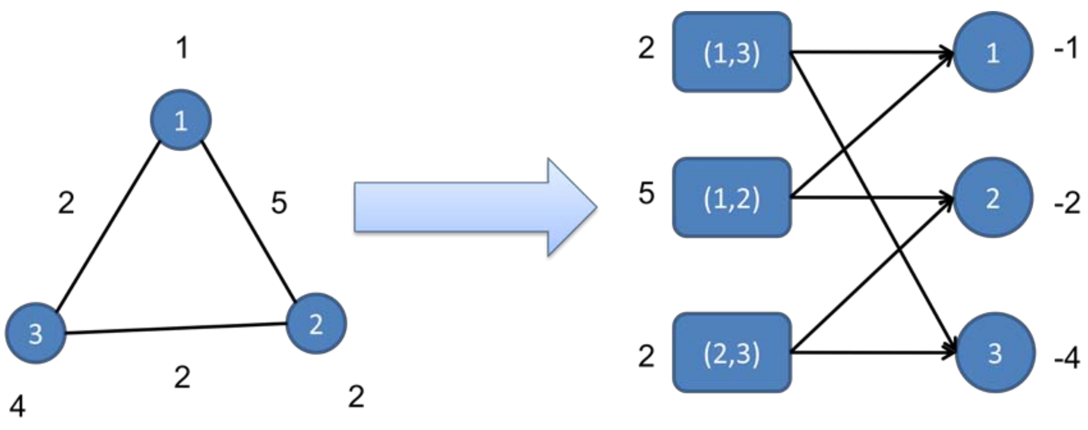
我们的目标就是要最大化 D 矩阵元素值，观察上式，发现当我们将一个 $a_i=1$ 时，c_i 的权值就会从最后的答案中扣除，当 $a_i=1$ 且 $a_j=1$ 时，b_{ij} 就会加入最后的答案。我们发现问题中 B 矩阵十分类似用于描述图的邻接矩阵，因此我们联想到可以尝试用图论模型来描述这个问题。

首先，对于 B 矩阵的元素 b_{ij} 是否加入答案与 a_i 和 a_j 两个元素有关，因此我们可以令其为 i，j 之间连边的边权；对于 C 矩阵的元素 c_i 是否从答案中扣除只与 a_i 一个元素有关，所以我们可以令其为点 i 的点权；那么 A 矩阵中 a_i 为 0 或 1 表示最后方案中对 i 点的决策。据此，我们可以将原问题表述为以下的图论问题：

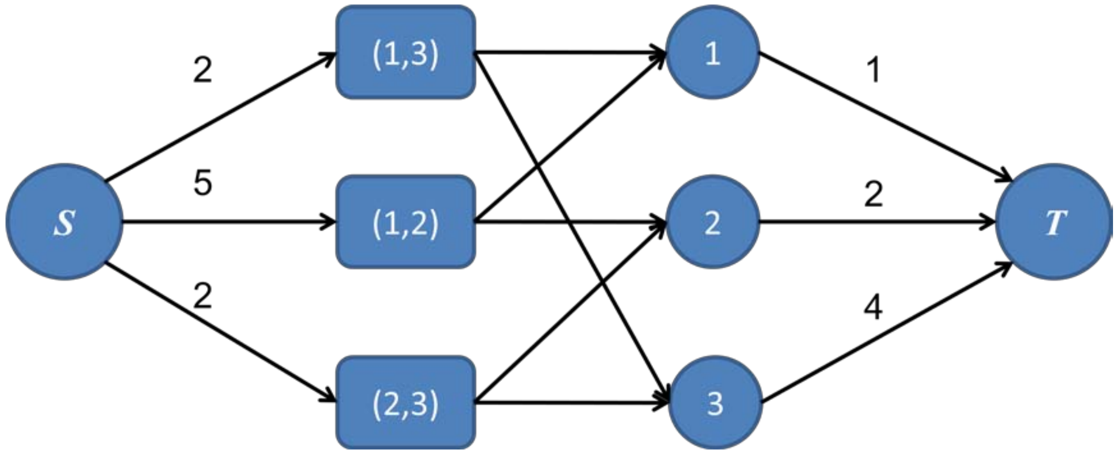
给出一个 N 个点的带权无向完全图，每个点的权值为 $p_i=c_i-b_{ii}$,图中点 i 和点 $j(i\neq j)$之间所连接的无向边边权为 $w_{ij}=b_{ij}+b_{ji}$；

` 要求你寻找一个子图，使得子图内边权之和减去点权之和最大。

这个图论的模型让人联想起 NOI2006 《最大获利》问题，的确我们可以套用最大权闭合子图的模型解题，如图：



我们可以重新构建一个二分图，将原图中的每条边独立成点，作为二分图的左部，左部点的权值为原图的边权；右部有 N 个点，代表图中的 N 个点，点权值为原图中点权的相反数。二分左部每个点向右部它所代表原图的边的两个端点连有向边。那么，问题就成功的被转化为经典的最大权闭合子图问题，可以采用网络流解决。



如图，对于最大权闭合子图问题的一般解法：我们可以新建源汇 S，T，源 S 向图中所有正权点连边，容量为点权，所有负权点向汇 T 连边，容量为权值的绝对值。原图中的有向边保留，容量为正无穷。在这个网络上做最大流，那么答案就是所有正权点权值之和，减去最大流的值。

具体证明可以参见 2006 年年鉴中周源的解题报告。

注意到，这样构建网络，网络中点将达到 N^2 级别的，当 $N=600$ 时，图中点的个数将会有 179700 个，1s 的时限该算法明显力不从心，因此算法还需改进。

时间复杂度 ： $O(Maxflow(n^2,n^2))$

空间复杂度： $O(N^2)$

期望得分： 60～70 分

算法三

算法二低效的原因是没有很好的抓住这道题的特点。本题的图中虽然点只有 600 个，但是，是一张完全图，所以边将会达到 10 万的级别，用点来代表边的算法不能取得很好的效果，我们需要另辟蹊径，我们直接讨论图论模型：

给出有 N 个点的无向完全图，点 i 和点 j 之间的边的边权为 $w_{ij}=b_{ij}+b_{ji}$，点 i 的点权为 $p_i=c_{1i}-b_{ii}$，现在要你找出一个导出子图，使得子图内的边权之和减去点权之和的值最大。

令原图为$G=(V,E)$，我们最后选择的子图为$G'=(V',E')$，$\overline{V'}$为V的补集：$\overline{V'}=V-V'$，$\overline{E'}$为E的补集：$\overline{E'}=E-E'$，那么我们的目标就是最大化：

$$|E'|-|V'|$$

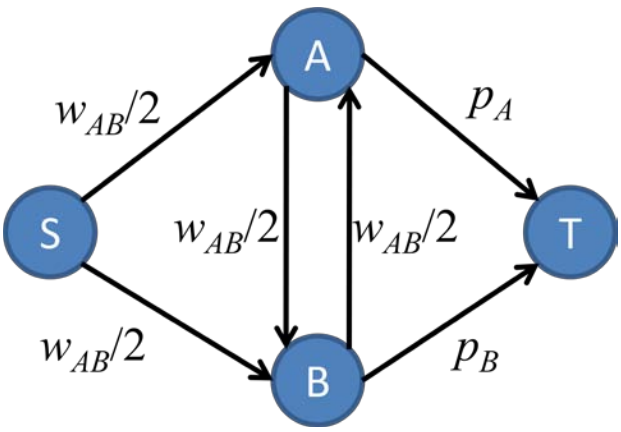
其中，$|E'|$和$|V'|$代表E'集合和V'的权值和。我们希望继续用网络流的方法来解题，根据最大流最小割定理，我们可以知道通过最大流求出的是最小割，使用最小割解决方法一般是将最小化问题与割对应，通过最小割求解最小化问题。但本题是一个最大化的问题，所以我们思考问题的反面，将问题转化为最小化问题。

$$\begin{aligned}&|E'|-|V'|\\=&\left|E-\overline{E'}\right|-|V'|\\=&|E|-\left|\overline{E'}\right|-|V'|\\=&|E|-\left(\left|\overline{E'}\right|+|V'|\right)\end{aligned}$$

由于$|E|$是一个定值，所以我们的目标变成最小化$\left|\overline{E'}\right|+|V'|$，问题成功转化。

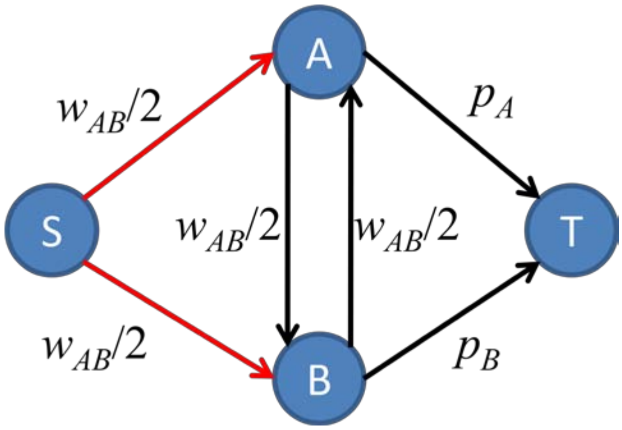
接下来我们所要做的工作就是构建一个网络，使图中的割和$\left|\overline{E'}\right|+|V'|$对应，通过求最小割来解决问题。

为了方便思考，我们先假设$N=2$，即原图中只有两个点A和B，这两个点的点权分别为p_A和p_B，两点连边的边权值为$|W|$。根据题意这两个点有 4 种选择方法，同时会产生 4 种的$\left|\overline{E'}\right|+|V'|$，我们的目标就是构建一张网络，使得网络的割的大小与 4 种$\left|\overline{E'}\right|+|V'|$的值一一对应，我们构图如下：

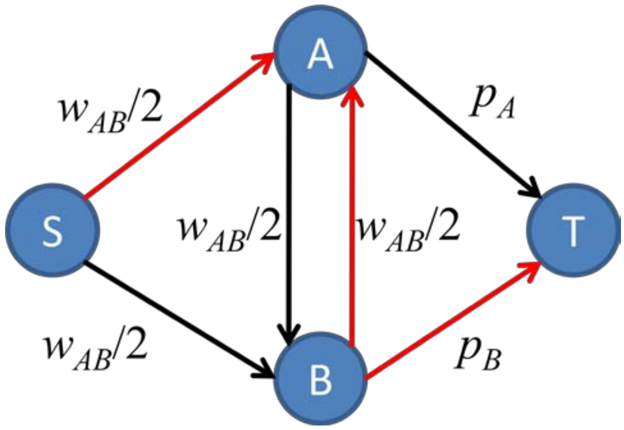


最后，在割中属于S集合的点属于V'，在割中属于T集合的点属于$\overline{V'}$，这样四种选择方法刚好就与图的四种割联系起来（红色的边为割边）：

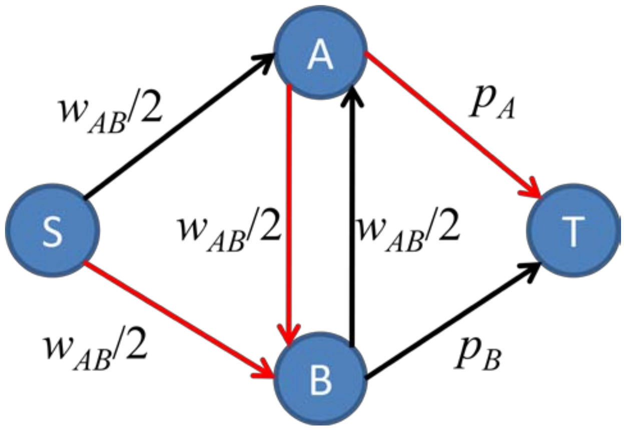
1. $A\in\overline{V}'$且$B\in\overline{V}'$，此时$\left|\overline{E}'\right|+\left|V'\right|=W$，对应的割如图，割大小也为$W$：



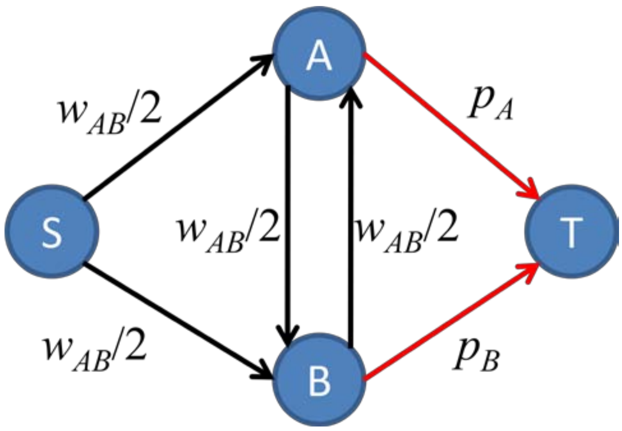
2. $A\in V'$且$B\in\overline{V}'$，此时$\left|\overline{E}'\right|+\left|V'\right|=W+p_A$，对应的割如图，割大小也为$W+p_A$：



3. $A\in\overline{V}'$且$B\in V'$，此时$\left|\overline{E}'\right|+\left|V'\right|=W+p_B$，对应的割如图，割大小也为$W+p_B$：



4. $A\in V'$且$B\in V'$，此时$\left|\overline{E}'\right|+\left|V'\right|=p_A+p_B$，对应的割如图，割大小也为$p_A+p_B$：



更一般的对于 N 个点的图，我们可以这样构图 ：新建源 S，汇 T，从源 S 向 N 个点各连一条有向边，与第 i 个点连边的容量为$\sum_{\substack{j=1 \\ j\neq i}}^{N}\frac{w_{ij}}{2}$，$N$ 个点各向汇 T 连一条有向边，第 i 个点连出的边的容量为 p_i，N 之间相互连边，对于 i, j 两个点 $(i \neq j)$，连边的容量为 $\frac{w_{ij}}{2}$。

定理：图的割与 V' 和 $\overline{V'}$ 的划分一一对应，并且割的大小与 $|E'|+|V'|$ 的值对应相等。

证明：我们令割中属于 S 集合的点在划分时属于 V' 集合，割中属于 T 集合的点划分时属于 $\overline{V'}$ 集合，那么我们就做到割与 V' 和 $\overline{V'}$ 的划分一一对应，此时割的大小为:

$$\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in S}}^{N}p_i$$

$$=\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\neq i \\ j\in T}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in S}}^{N}p_i$$

$$=\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\neq i \\ j\in T}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}w_{ij}+\sum_{\substack{i=1 \\ i\in S}}^{N}p_i$$

$$=\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\neq i \\ j\in T}}^{N}\frac{w_{ij}}{2}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}w_{ij}+\sum_{\substack{i=1 \\ i\in S}}^{N}p_i$$

$$=\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j>i \\ j\in T}}^{N}w_{ij}+\sum_{\substack{i=1 \\ i\in T}}^{N}\sum_{\substack{j=1 \\ j\in S}}^{N}w_{ij}+\sum_{\substack{i=1 \\ i\in S}}^{N}p_i$$

$$=|\overline{E'}|+|V'|$$

根据以上定理，最小割对应着就是最小的$\left|\overline{E'}\right|+\left|V'\right|$，这样我们就能得到最大的$\left|E\right|-(\left|\overline{E'}\right|+\left|V'\right|)$，即最大的$D$矩阵的元素值。

在上面的讨论中，我们都是当点权是非负数，注意到$p_i=c_{1,i}-b_{ii}$有可能是负数，需要我们的特别处理，根据贪心思想，当一个点的权值是负数的时候，最优解中，他一定属于V'。我们可以用反证法来证明：假设在某个最优解中，这个负权点不在V'中，那么我们把这个点放入V'，$\left|\overline{E'}\right|$不可能增大，$\left|V'\right|$将会减少，最后的答案$\left|E\right|-(\left|\overline{E'}\right|+\left|V'\right|)$将会变大，将会得到一个更优解，矛盾。具体的做法可以在一开始构图前对于所有权值小于 0 的点，将其权值的绝对值直接加入最后的答案中，然后把该点的点权改为 0。

使用以上方法，构出的网络点数降到了$O(N)$级别，但是规模还是比较大的，所以本题还要注意网络流的优化，在此类边较为稠密的图中，当前弧优化往往会有很好的效果。笔者所用的就是 sap+当前弧优化，即使是极限数据，也可以在 0.5s 以内出解。

时间复杂度 ： $O(Maxflow(n,n^2))$

空间复杂度： $O(N^2)$

期望得分： 80～100 分

算法四

算法二、三都是在把问题转化为图论模型基础上解决的，我们也可以直接在矩阵模型上构思，观察在算法二中，我们得到：

$$D=\left(\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{\substack{j=1\\a_j=1}}^{N}b_{ij}-\sum_{\substack{i=1\\a_i=1}}^{N}c_i\right)$$

继续往下：

$$=\left(\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{j=1}^{N}b_{ij}-\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{\substack{j=1\\a_j=0}}^{N}b_{ij}-\sum_{\substack{i=1\\a_i=1}}^{N}c_i\right)$$

$$=\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{N}b_{ij}-\sum_{\substack{i=1\\a_i=0}}^{N}\sum_{j=1}^{N}b_{ij}\right)-\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{\substack{j=1\\a_j=0}}^{N}b_{ij}-\sum_{\substack{i=1\\a_i=1}}^{N}c_i\right)$$

$$=\left(\sum_{i=1}^{N}\sum_{j=1}^{N}b_{ij}-\left(\underline{\sum_{\substack{i=1\\a_i=0}}^{N}\sum_{j=1}^{N}b_{ij}+\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{\substack{j=1\\a_j=0}}^{N}b_{ij}+\sum_{\substack{i=1\\a_i=1}}^{N}c_i}\right)\right)$$

因为$\sum_{i=1}^{N}\sum_{j=1}^{N}b_{ij}$是一个定值，所以我们将问题转化成一个最小化问题，目标最小化划线部分的值。这个问题可以使用最小割来解决问题（其实，这里本质上和算法 3 的转化方法是相同的），这里提供一个与算法三不同的构图：

我们新建源S汇T和N个点，从源S向N个点各连一条有向边，向i点连边容量为$\sum_{j=1}^{N}b_{ji}$；在N个点中，对于任意一对有序数对$(i,j)(i\neq j)$，点i连向点j的边的容量为b_{ij}；N个点向汇T连有向边，i点连出去的边的容量为c_i。

这样我们可以将图的割与$\sum_{\substack{i=1\\a_i=0}}^{N}\sum_{j=1}^{N}b_{ij}+\sum_{\substack{i=1\\a_i=1}}^{N}\sum_{\substack{j=1\\a_j=0}}^{N}b_{ij}+\sum_{\substack{i=1\\a_i=1}}^{N}c_i$的值一一对应起来，通过求最小割，来最大化$D$的值。具体证明方法类似算法三，这里就不再赘述。

时间复杂度 ： $O(Maxflow(n,n^2))$

空间复杂度： $O(N^2)$

期望得分： 80～100 分

思路总结

总结本题的分析思路，首先对于一个矩阵问题，我们先尝试使用搜索算法解决，但是由于时间复杂度太高而宣告失败；经过矩阵运算的性质分析，我们将其转化为图论模型，套用经典模板，但是构建网络规模太大，不能很好处理问题，于是我们思考将最大化问题通过补集转化思想变为最小化问题，构建网络使最小化问题的方案与网络的割一一对应，通过求解最小割来求最小化问题，返回求解最大化问题，终于得到解决。

将最小化问题通过构建网络，使其与网络的割对应，通过求解最小割来寻求最小化问题的解，这正反映了用最小割来解决问题的一般思路。

参考文献

[1] 《最大获利》解题报告 周源

[2] 《最小割模型在信息学竞赛中的应用》 胡伯涛

感谢

感谢我的指导老师——福建省福州一中学的陈颖老师

感谢福建省福州一中信息组的全体同学对我的帮助

感谢陈健飞，李新野，赖陆航等集训队队员对我的帮助

感谢福建省福州第三中学王君行同学对我的帮助

附录

知识点

矩阵运算：考察选手是否能熟练运用矩阵运算，将题目给出的公式化简；

模型转换：是否能通过推导出的公式联想到图论模型，使问题更加直观；

最小割：是否能熟练的运用最小割解决问题，是否掌握用割解决问题的一般思路。

数据生成

众所周知，网络流算法虽然理论时间复杂度高，但是实际运行速度往往很快，因此要想出数据卡住最大权闭合子图的算法确实有难度。并且本题的数据规模较大，如果纯随机的生成矩阵 B 和 C，往往会因为过于平均而造成矩阵 A 的最优解全是 0，或者全是 1。笔者为了避免这种情况的出现，在生成数据的时候希望矩阵元素之间的差距尽量大。对于每个格子以一定的概率生成一个很小的随机数，以一定的概率生成一个很大的随机数，其他情况生成一个正常随机值，至于三种概率各是多少，要针对 N 大小的不同而调整。

同时 B，C 矩阵的元素值的比例也需要考虑，由于 C 矩阵代表点权，B 矩阵代表边权，因此 C 矩阵元素的值与 B 矩阵元素值的比例应该大约为 N 的级别，具体数值要根据生成数据的规模不断的调整。

笔者在生成测试数据时，根据数据规模的不同，写了 3 个数据生成器，并且写了 8 程序对数据进行检测。首先保证了数据的正确性；其次，各种网络流实现的原理各不相同，多种构图加多种网络流实现的程序进行比较，保证数据足够强大。

预计分数分布

测试数据从小到大分布，前 30%的数据是小规模数据，通过简单的搜索优化就可以解决，是为初涉 NOI 的选手准备的，应该大多数选手都可以拿到这部分分数。中间 40%的数据规模中等，为了考察选手是否能从题目的矩阵运算转化为图论模型，并且灵活运用经典算法，有一定的思维难度，但是这也是一个合格 NOI 选手必须拥有的能力，因此我估计大部分选手可以拿到这部分的分数。最后 30%的数据都是极限大数据，要求选手能熟练使用最小割来解

决问题，能够突破传统算法，有创新的思维。考虑到这部分难度较大，预计能拿到这部分分数的选手人数应该比较少。