网络流建模汇总

by Edelweiss

目　录

最大流 4

《POJ 1149　PIGS》 4
《POJ 1637　Sightseeing tour》 8
《POJ 2391　Ombrophobic Bovines》 9
《POJ 2699　The Maximum Number of Strong Kings》 10
《POJ 3281　Dining》 11
《JOJ 2453　Candy》 11
《ZOJ 2760　How Many Shortest Path》 12
《WOJ 1124　Football Coach》 12
《SGU 326　Perspective》 13
《SGU 438　The Glorious Karlutka River =)》 14
《SPOJ 287　Smart Network Administrator》 14
《SPOJ 962　Intergalactic Map》 15
《小结》 15

最小割 15

《HOJ 2634　How to earn more》 16
《HOJ 2713　Matrix1》 16
《POJ 1815　Friendship》 17
《ZOJ 2532　Internship》 17
《Ural 1277　Cops and Thieves》 18
《SPOJ 839　Optimal Marks》 18
《SPOJ 1693　Coconuts》 19
《Beijing Regional 2008 Problem A　Destroying the bus stations》 19
《小结》 20

有上下界 21

《POJ 2396　Budget》 21
《小结》 22

最小费用流 22

《HOJ 2543　Stone IV》 22
《HOJ 2715　Matrix3》 23
《HOJ 2739　The Chinese Postman Problem》 23
《POJ 3680　Intervals》 24
《SPOJ 371　Boxes》 24
《BASHU 2445　餐巾问题》 25

《剪刀石头布》.....25
《小结》.....26

最大流

《POJ 1149　PIGS》

【题目大意】
有 M 个猪圈，每个猪圈里初始时有若干头猪。一开始所有猪圈都是关闭的。依次来了 N 个顾客，每个顾客分别会打开指定的几个猪圈，从中买若干头猪。每个顾客分别都有他能够买的数量的上限。每个顾客走后，他打开的那些猪圈中的猪，都可以被任意地调换到其它开着的猪圈里，然后所有猪圈重新关上。问总共最多能卖出多少头猪。（1 <= N <= 100, 1 <= M <= 1000）

举个例子来说。有 3 个猪圈，初始时分别有 3、1 和 10 头猪。依次来了 3 个顾客，第一个打开 1 号和 2 号猪圈，最多买 2 头；第二个打开 1 号和 3 号猪圈，最多买 3 头；第三个打开 2 号猪圈，最多买 6 头。那么，最好的可能性之一就是第一个顾客从 1 号圈买 2 头，然后把 1 号圈剩下的 1 头放到 2 号圈；第二个顾客从 3 号圈买 3 头；第三个顾客从 2 号圈买 2 头。总共卖出 2+3+2=7 头。

【建模方法】
不难想象，这个问题的网络模型可以很直观地构造出来。就拿上面的例子来说，可以构造出图 1 所示的模型（图中凡是没有标数字的边，容量都是∞）：

- 三个顾客，就有三轮交易，每一轮分别都有 3 个猪圈和 1 个顾客的结点。
- 从源点到第一轮的各个猪圈各有一条边，容量就是各个猪圈里的猪的初始数量。
- 从各个顾客到汇点各有一条边，容量就是各个顾客能买的数量上限。
- 在某一轮中，从该顾客打开的所有猪圈都有一条边连向该顾客，容量都是∞。
- 最后一轮除外，从每一轮的 i 号猪圈都有一条边连向下一轮的 i 号猪圈，容量都是∞，表示这一轮剩下的猪可以留到下一轮。
- 最后一轮除外，从每一轮被打开的所有猪圈，到下一轮的同样这些猪圈，两两之间都要连一条边，表示它们之间可以任意流通。

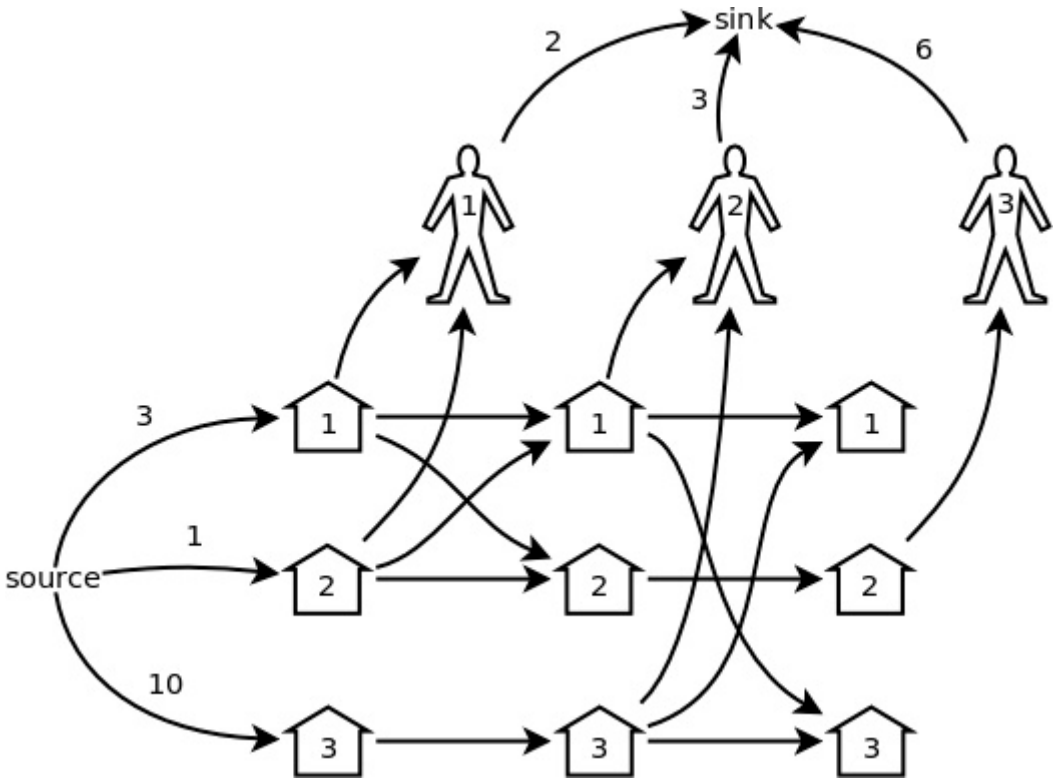


图 1

这个网络模型的最大流量就是最多能卖出的数量。图中最多有 2+N+M×N≈100,000 个结点。这个模型虽然很直观，但是结点数太多了，计算速度肯定会很慢。其实不用再想别的算法，就让我们继续上面的例子，用合并的方法来简化这个网络模型。

首先，最后一轮中没有打开的猪圈就可以从图中删掉了，也就是图 2 中红色的部分，显然它们对整个网络的流量没有任何影响。

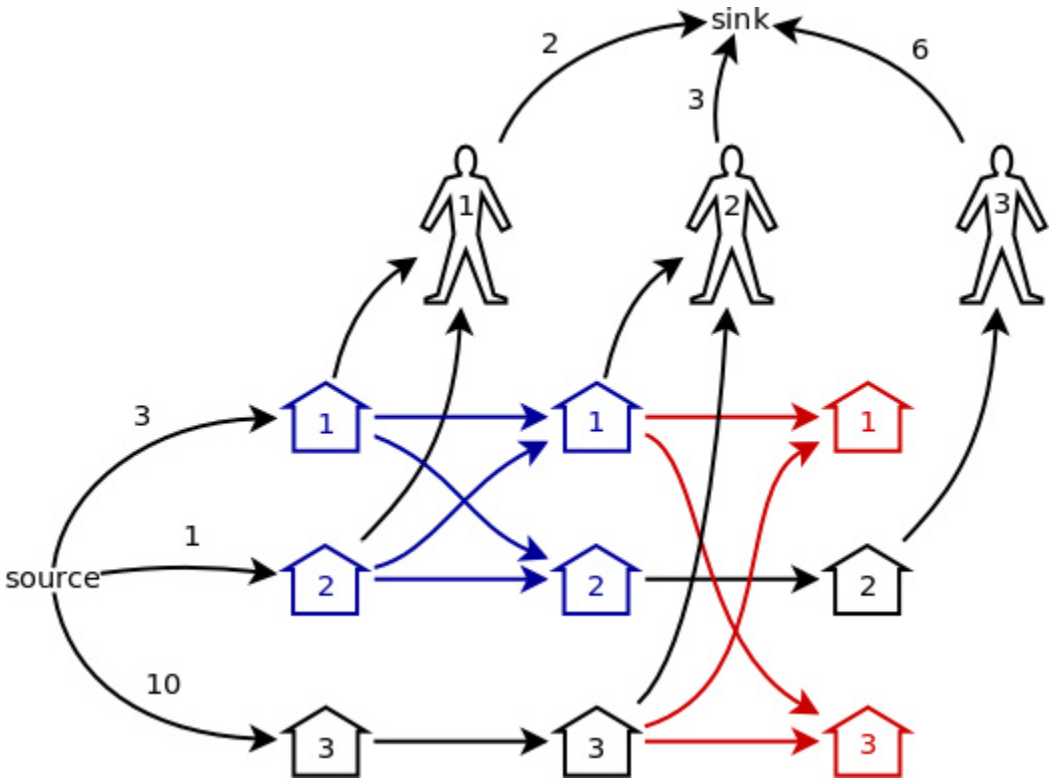


图 2

接着，看图 2 中蓝色的部分。根据我总结出的以下几个规律，可以把这 4 个点合并成一个：

规律 1. 如果几个结点的流量的来源完全相同，则可以把它们合并成一个。

规律 2. 如果几个结点的流量的去向完全相同，则可以把它们合并成一个。

规律 3. 如果从点 u 到点 v 有一条容量为∞的边，并且点 v 除了点 u 以外没有别的流量来源，则可以把这两个结点合并成一个。

根据规律 1，可以把蓝色部分右边的 1、2 号结点合并成一个；根据规律 2，可以把蓝色部分左边的 1、2 号结点合并成一个；最后，根据规律 3，可以把蓝色部分的左边和右边（已经分别合并成了一个结点）合并成一个结点。于是，图 2 被简化成了图 3 的样子。也就是说，最后一轮除外，每一轮被打开的猪圈和下一轮的同样这些猪圈都可以被合并成一个点。

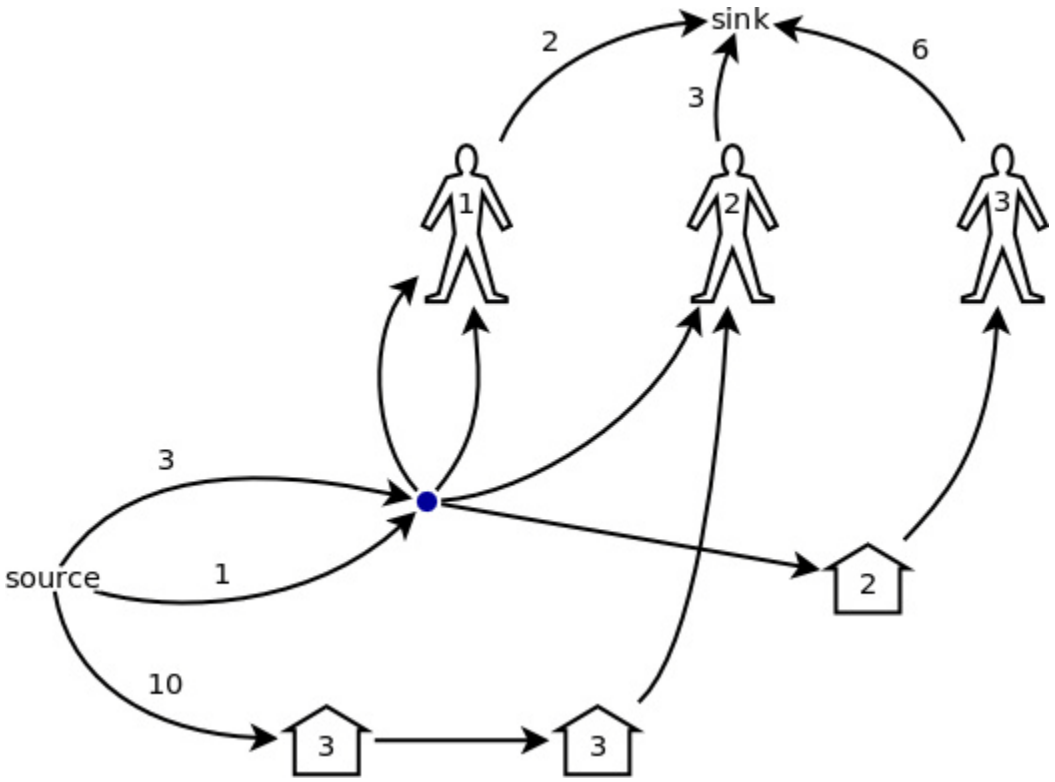


图 3

接着，根据规律 3，图 3 中的蓝色结点、2 号猪圈和 1 号顾客这三点可以合并成一个；图 3 中的两个 3 号猪圈和 2 号顾客也可以合并成一个点。当然，如果两点之间有多条同向的边，则这些边可以合并成一条，容量相加，这个道理很简单，就不用我多说了。最终，上例中的网络模型被简化成了图 4 的样子。

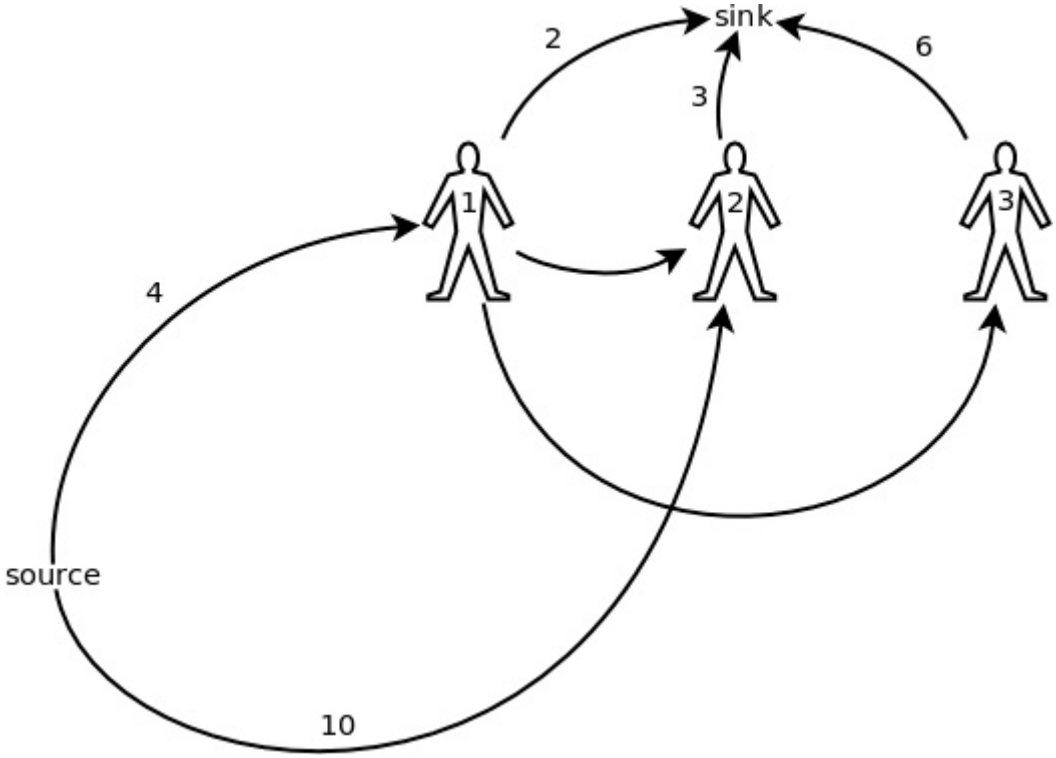


图 4

让我们从图 4 中重新总结一下构造这个网络模型的规则：

- 每个顾客分别用一个结点来表示。
- 对于每个猪圈的第一个顾客，从源点向他连一条边，容量就是该猪圈里的猪的初始数量。如果从源点到一名顾客有多条边，则可以把它们合并成一条，容量相加。
- 对于每个猪圈，假设有 n 个顾客打开过它，则对所有整数 i∈[1, n)，从该猪圈的第 i 个顾客向第 i + 1 个顾客连一条边，容量为∞。
- 从各个顾客到汇点各有一条边，容量是各个顾客能买的数量上限。
-

拿我们前面一直在讲的例子来说：1 号猪圈的第一个顾客是 1 号顾客，所以从源点到 1 号顾客有一条容量为 3 的边；1 号猪圈的第二个顾客是 2 号顾客，因此从 1 号顾客到 2 号顾客有一条容量为∞的边；2 号猪圈的第一个顾客也是 1 号顾客，所以从源点到 1 号顾客有一条容量为 1 的边，和之前已有的一条边合并起来，容量变成 4；2 号猪圈的第二个顾客是 3 号顾客，因此从 1 号顾客到 3 号顾客有一条容量为∞的边；3 号猪圈的第一个顾客是 2 号顾客，所以从源点到 2 号顾客有一条容量为 10 的边。

新的网络模型中最多只有 2 + N = 102 个结点，计算速度就可以相当快了。可以这样理解这个新的网络模型：对于某一个顾客，如果他打开了猪圈 h，则在他走后，他打开的所有猪圈里剩下的猪都有可能被换到 h 中，因而这些猪都有可能被 h 的下一个顾客买走。所以对于一个顾客打开的所有猪圈，从该顾客到各猪圈的下一个顾客，都要连一条容量为∞的边。

在面对网络流问题时，如果一时想不出很好的构图方法，不如先构造一个最直观，或者说最“硬来”的模型，然后再用合并结点和边的方法来简化这个模型。经过简化以后，好的构图思路自然就会涌现出来了。这是解决网络流问题的一个好方法。

《POJ 1637　Sightseeing tour》

【题目大意】

混合图欧拉回路。（1 <= N <= 200, 1 <= M <= 1000）

【建模方法】

把该图的无向边随便定向，计算每个点的入度和出度。如果有某个点出入度之差为奇数，那么肯定不存在欧拉回路。因为欧拉回路要求每点入度 = 出度，也就是总度数为偶数，存在奇数度点必不能有欧拉回路。

好了，现在每个点入度和出度之差均为偶数。那么将这个偶数除以 2，得 x。也就是说，对于每一个点，只要将 x 条边改变方向（入>出就是变入，出>入就是

变出），就能保证出=入。如果每个点都是出=入，那么很明显，该图就存在欧拉回路。

现在的问题就变成了：我该改变哪些边，可以让每个点出=入？构造网络流模型。首先，有向边是不能改变方向的，要之无用，删。一开始不是把无向边定向了吗？定的是什么向，就把网络构建成什么样，边长容量上限 1。另新建 s 和 t。对于入>出的点 u，连接边(u, t)、容量为 x，对于出>入的点 v，连接边(s, v)，容量为 x（注意对不同的点 x 不同）。之后，察看是否有满流的分配。有就是能有欧拉回路，没有就是没有。欧拉回路是哪个？察看流值分配，将所有流量非 0（上限是 1，流值不是 0 就是 1）的边反向，就能得到每点入度=出度的欧拉图。

由于是满流，所以每个入>出的点，都有 x 条边进来，将这些进来的边反向，OK，入=出了。对于出>入的点亦然。那么，没和 s、t 连接的点怎么办？和 s 连接的条件是出>入，和 t 连接的条件是入>出，那么这个既没和 s 也没和 t 连接的点，自然早在开始就已经满足入=出了。那么在网络流过程中，这些点属于“中间点”。我们知道中间点流量不允许有累积的，这样，进去多少就出来多少，反向之后，自然仍保持平衡。

所以，就这样，混合图欧拉回路问题，解了。

《POJ 2391　Ombrophobic Bovines》

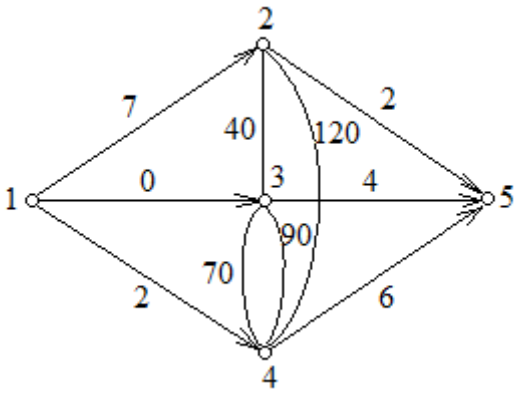
【题目大意】

给定一个无向图，点 i 处有 Ai 头牛，点 i 处的牛棚能容纳 Bi 头牛，求一个最短时间 T 使得在 T 时间内所有的牛都能进到某一牛棚里去。（1 <= N <= 200, 1 <= M <= 1500, 0 <= Ai <= 1000, 0 <= Bi <= 1000, 1 <= Dij <= 1,000,000,000）

【建模方法】

将每个点 i 拆成两个点 i’, i’’，连边(s, i’, Ai), (i’’, t, Bi)。二分最短时间 T，若 d[i][j]<=T（d[i][j]表示点 i, j 之间的最短距离）则加边(i’, j’’, ∞)。每次根据最大流调整二分的上下界即可。

一种错误的建图方法，即不拆点，见下图：



其中每条无向边表示两条方向相反的有向边，容量均为∞。

当二分到 T = 70 的时候，显然我们只加入了(2, 3)和(3, 4)两条无向边，因为只有这两对点间的最短距离小于等于 70。但是从图中也可以看出，由于没有拆点，点 2 也可以通过这两条边到达点 4，而实际上这是不允许的。也就是说我们所加的限制条件没有起到作用。由此可见，只有拆点才是正确的做法。

《POJ 2699　The Maximum Number of Strong Kings》

【题目大意】
一场联赛可以表示成一个完全图，点表示参赛选手，任意两点 u, v 之间有且仅有一条有向边(u, v)或(v, u)，表示 u 打败 v 或 v 打败 u。一个选手的得分等于被他打败的选手总数。一个选手被称为“strong king”当且仅当他打败了所有比他分高的选手。分数最高的选手也是 strong king。现在给出某场联赛所有选手的得分序列，由低到高，问合理安排每场比赛的结果后最多能有几个 strong king。已知选手总数不超过 10 个。

【建模方法】
此题乍看之下无从下手，但是注意到 strong king 只能在分数最高的若干人当中产生，而总人数只有 10 个，故我们可以枚举 strong king 的个数 C，那么分数最高的 C 个便是 strong king，接下来按下列方法构图：每场比赛 i 作为一个点（总共 tot=n*(n-1)/2 个），加边(s, i, 1)；每名选手 j 作为一个点，加边(j, t, score[j])，并再拉出一个点 j’，如果 j 是 strong king，则加边(j’, j, score[j]-Max[j])（注意此处 score[j] - Max[j] 有可能为负，所以要在构图之前判一下。设当前分数最小的 strong king 为 mj，若 score[mj] < Max[mj]则直接停止枚举，因为 mj 每场都赢也不够把所有比他分高的选手赢个遍），其中 Max[i]表示比选手 i 分数高的选手总数；否则加边(j’, j, score[j])。对每场比赛 i，设参赛的两名选手为 Ui, Vi 且 score[Ui] <= score[Vi]，若 Ui 是 strong king 且 score[Ui] < score[Vi]，则加边 (i, Ui, 1)；否则加边(i, Ui’, 1), (i, Vi’, 1)。当某次最大流小于 tot 时停止枚举即可。
对每名选手 j，所有连到 j 的边(i, j)表示比赛 i 必须让 j 赢。若 j 是 strong king，则(j’, j, score[j]-Max[j])的目的是除去必须让 j 赢的那 Max[j]场以外，剩下的 score[j] – Max[j]场可以随意赢剩下的 N – Max[j]名选手。对于非 strong king 的选手也是类似的作用。
这里提一下我的一种错误建模方法：
每名选手 i 用三个点 i’, i’’, i’’’表示，加边(s, i’, score[i]), (i’’’, t, N-1-score[i])，而 i’’则充当那个被拉出来的点。其他细节不再提。这样构图之后每一条 s-t 流表示一场比赛，看似合理但其实是错误的，因为它没有考虑到任何两名选手之间只能进行一场比赛，而这个网络中可能存在两条 s-t 流，它们关联的是同样的两名选手，只不过胜负对调。故以后在构图时要时刻关注它的严谨性和正确性。

《POJ 3281　Dining》

【题目大意】
有 F 种食物和 D 种饮料，每种食物或饮料只能供一头牛享用，且每头牛只享用一种食物和一种饮料。现在有 N 头牛，每头牛都有自己喜欢的食物种类列表和饮料种类列表，问最多能使几头牛同时享用到自己喜欢的食物和饮料 （1 <= F <= 100, 1 <= D <= 100, 1 <= N <= 100）

【建模方法】
此题的建模方法比较有开创性。以往一般都是左边一个点集表示供应并与源相连，右边一个点集表示需求并与汇相连。现在不同了，供应有两种资源，需求仍只有一个群体，怎么办？其实只要仔细思考一下最大流的建模原理，此题的构图也不是那么难想。最大流的正确性依赖于它的每一条 s-t 流都与一种实际方案一一对应。那么此题也需要用 s-t 流将一头牛和它喜欢的食物和饮料“串”起来，而食物和饮料之间没有直接的关系，自然就想到把牛放在中间，两边是食物和饮料，由 s, t 将它们串起来构成一种分配方案。至此建模的方法也就很明显了：每种食物 i 作为一个点并连边(s, i, 1)，每种饮料 j 作为一个点并连边(j, t, 1)，将每头牛 k 拆成两个点 k’, k’’并连边(k’, k’’, 1), (i, k’, 1), (k’’, j, 1)，其中 i, j 均是牛 k 喜欢的食物或饮料。求一次最大流即为结果。

《JOJ 2453　Candy》

【题目大意】
有 N 颗糖果和 M 个小孩，老师现在要把这 N 颗糖分给这 M 个小孩。每个小孩 i 对每颗糖 j 都有一个偏爱度 Aij，如果他喜欢这颗糖，Aij = 2，否则 Aij = 1。小孩 i 觉得高兴当且仅当∑Cij×Aij >= Bi，j=1,2,…,N，若他分得了糖 j，Cij = 1，否则 Cij = 0。问能否合理分配这 N 颗糖，使得每个小孩都觉得高兴。（1 <= N <= 100,000, 1 <= M <= 10, 0 <= Bi <= 1,000,000,000）

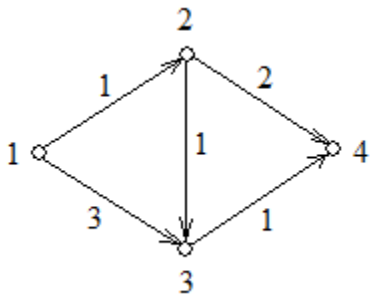
【建模方法】
一种最直观的想法就是每颗糖 i 作为一个点并连边(s, i, ?)，每个小孩 j 作为一个点并连边(j, t, Bj)。若小孩 j 喜欢糖果 i 则连边(i, j, 2)，否则连边(i, j, 1)，然后求一次最大流看是否等于∑Bj。但是问题也很明显，我们还没有给与源点关联的边确定容量。实际上我们无法确定它们的容量，因为最大流无法实现这样一种控制：一个点有若干条出边，容量不尽相同，现在要求经过该点的流可以任选一条出边流走，且一旦选定之后就只能从这条边流而不能再进入其他的出边。因此我们无法确定与源关联的边的容量，因为经过每颗糖 i 的流无法在出边容量有 1 又有 2 的情况下作出正确的选择。
那么是否就没有办法了呢？虽然流无法在容量有 1 又有 2 的情况下作出正确的选

择，但却可以在容量有 1 又有 0 的情况下最自然地作出正确的选择，流过去就表示选择了那条出边，且因为容量为 1，不会再有流进入其他的出边。那么此题的构图方法也就出来了：每颗糖 i 作为一个点并连边(s, i, 1)，每个小孩 j 作为一个点并连边(j, t, floor(Bj/2))，若小孩 j 喜欢糖果 i 则连边(i, j, 1)，否则连边(i, j, 0)或者干脆不连边，效果一样。设最大流为 ans，若 ans+N >= ΣBj 则可以满足要求。为什么？因为每颗糖迟早都要分给某个小孩，它一定会为总权值贡献 1，只不过如果它分给了喜欢它的小孩就再额外贡献 1。现在我只考虑这额外的 1 单位贡献究竟能送出去多少，最后加上基值 N 并与ΣBj 比较即可。

《ZOJ 2760　How Many Shortest Path》

【题目大意】
给定一个带权有向图 G=(V, E)和源点 s、汇点 t，问 s-t 边不相交最短路最多有几条。（1 <= N <= 100）

【建模方法】
分别从源点和汇点作一次 Dijkstra，然后为建图方便仍保留所有的点，但只加入满足 ds[u]+w[u][v]+dt[v]==dst 的边(u, v)（这样便保证网络中的任意一条 s-t 路都是最短路），容量为 1。求一次最大流即为结果。
注意不能按以下方法建图，因为这样会引进一些不在 s-t 最短路上的边：
保留所有满足 ds[u]+dt[u]==dst 的点 u，在这些点的导出子图中求最大流。
反例：如下图



上述方法会将(1, 3)引入，导致形成错误流 1-3-4，而第一种方法则不会。其实只要将第二种方法也加上对边的限制条件而不是将导出子图中的所有边都加上，同样也是正确的。

《WOJ 1124　Football Coach》

【题目大意】
有 N 支球队，互相之间已经进行了一些比赛，还剩下 M 场没有比。现在给出各支球队目前的总分以及还剩下哪 M 场没有比，问能否合理安排这 M 场比赛的结果，使得第 N 支球队最后的总分大于其他任何一支球队的总分。已知每场比赛胜者得 2 分，败者 0 分，平局则各得 1 分。（1 <= N <= 100, 0 <= M <= 1000）

【建模方法】
首先，所有跟球队 N 相关的比赛都要让球队 N 赢。如果此时仍有某支球队的总分大于等于球队 N 的总分，则已经不可能满足要求；否则按如下方法建图：每场比赛 i（不包括与球队 N 相关的比赛）作为一个点并加边(s, i, 2)，每支球队 j（不包括球队 N）作为一个点并加边(j, t, score[N]-score[j]-1)，每场比赛向与其关联的两支球队 u, v 连边(i, u, 2), (i, v, 2)。若最大流等于 2*比赛场数（不包括与球队 N 相关的比赛）则可以满足要求。

发散思维：
(1) 如果不允许平局存在，该如何构图？
(s, i, 2) ---> (s, i, 1)
(j, t, score[N]-score[j]-1) ---> (j, t, floor((score[N]-score[j]-1)/2))
(i, u, 2), (i, v, 2) ---> (i, u, 1), (i, v, 1)

《SGU 326 Perspective》

【题目大意】
NBA 某小组内有 N 支球队，小组内以及小组间已经进行了若干场比赛。现在给出这 N 支球队目前胜利的场数、还剩多少场没有比（包括小组内和小组间）以及小组内任意两支球队之间还剩多少场没有比，问能否合理安排剩下的所有比赛，使得球队 1 最后胜利的场数至少和小组内任何一支其他球队一样。（2 <= N <= 20, 0 <= x <= 10000, x 表示其他任何输入）

【建模方法】
此题和上一题非常相似。同样，所有和球队 1 相关的比赛全让球队 1 赢，如果此时仍有某支球队胜利的场数大于球队 1，则已经不可能满足要求。按如下方法建图：所有小组内的比赛 i（不包括与球队 1 相关的比赛）作为一个点并加边(s, i, num[i])，每支球队（不包括球队 1）作为一个点并加边(j, t, wins[1]-wins[i])，每场比赛向与其关联的两支球队 u, v 连边(i, u, ∞), (i, v, ∞)。至于其他球队小组间的比赛，直接让他们输掉就好，不用管。若最大流等于 Σnum[i]则可以满足要求。

《SGU 438　The Glorious Karlutka River =)》

【题目大意】
有一条东西向流淌的河，宽为 W，河中有 N 块石头，每块石头的坐标(Xi, Yi)和最大承受人数 Ci 已知。现在有 M 个游客在河的南岸，他们想穿越这条河流，但是每个人每次最远只能跳 D 米，每跳一次耗时 1 秒。问他们能否全部穿越这条河流，如果能，最少需要多长时间。（0 <= N <= 50, 0 < M <= 50, 0 <= D <= 1000, 0 < W <= 1000, 0 < Xi < 1000, 0 < Yi < W, 0 <= Ci <= 1000）

【建模方法】
经典的**动态流问题**，在流量限制的基础上加入了时间限制。首先将南岸作为源点，北岸作为汇点，将每块石头拆成两个点(i', i'')并连边(i', i'', Ci)。接下来我们再按时间将每块石头拆点，形成一个分层网络 time-expanded network，每块石头在不同的时间点都有一个点表示。所有能从南岸跳到的石头，从源点向其各时间点处连边，容量为∞。所有能跳到北岸的石头，从其各时间点处向汇点连边，容量为∞。任意两块距离小于等于 D 的石头，互相从 t 到 t+1 连边，容量为∞。枚举 t，并不断地往网络中加点表示当前时刻的石头，直到最大流等于总人数为止，此时的 t 即为结果。

《SPOJ 287　Smart Network Administrator》

【题目大意】
一座村庄有 N 户人家。只有第一家可以连上互联网，其他人家要想上网必须拉一根缆线通过若干条街道连到第一家。每一根完整的缆线只能有一种颜色。网管有一个要求，各条街道内不同人家的缆线必须不同色，且总的不同颜色种数最小。求在指定的 K 户人家都能上网的前提下，最小的不同颜色种数。（1 <= N <= 500）

【建模方法】
此题乍一看不太好考虑，不知道如何去控制同一街道内的缆线颜色各不相同。我们不妨先尝试着建立一个网络模型出来：以第一家作为汇点 t，K 户人家中的每一户 i 作为一个点并连边(s, i, 1)，对每条街道(u, v)，连边(u, v, c), (v, u, c)，c=∞。这样求完最大流后每一条 s-t 流均对应一户人家的缆线，而各条街道内的流量表示有多少户人家的缆线同时穿过该街道，那么这个流量就是只考虑该条街道的时候最少的不同颜色种数。那么答案是否就是所有街道的不同颜色种数的最大值呢？当然不是！最大流只保证总流量最大，而不会去管每条流具体该往哪儿流，所以这么做不一定能得到最优解。我们只要稍微修改这个模型就一定能保证得到最优解：强制 c 等于某个值 limit，再对网络求最大流，如果等于 K，说明用 c 种不同的颜色已经足够了；如果小于 K，说明 c 种还不够，还需要往高了调。同时此处的单调性也很明显：c 越高越容易满足要求。思路一下就豁然开朗了：二分 c 的

值，如果足够了就往下调，否则往上调，直到找到足够和不足够的临界值为止。

《SPOJ 962　Intergalactic Map》

【题目大意】
在一个无向图中，一个人要从 A 点赶往 B 点，之后再赶往 C 点，且要求中途不能多次经过同一个点。问是否存在这样的路线。（3 <= N <= 30011, 1 <= M <= 50011）

【建模方法】
由于每个点只能走一次，似乎最短路之类的算法不能用，只有往网络流上靠。将每个点 i 拆成两个点 i', i''并加边(i', i'', 1)就能轻易达到这个目的。起初我一直以 A 为源点思考，却怎么也想不出如何处理先后经过两个汇点的问题，直到灵光一现，想到可以以 B 为源点，A、C 为汇点，看能否增广两次。这个转化看似简单，但却蕴含着一个深刻的道理：条条大路通罗马！正所谓“天涯何处无芳草，何必单恋一枝花”，这样行不通那就换一种方法，说不定一下就妥了！

《小结》

一般来讲，最大流构图最直观最容易理解，但变化也是相当之多，需要多多 A 题多多体会。常用的构图方法有以下几种：

(1) 用 s-t 流表示方案
这是最大流最常用的构图法，每一条 s-t 流都实实在在地对应着实际问题中的一种操作方案，很好理解。

(2) 待补充。

最小割

《HOJ 2634　How to earn more》

【题目大意】
有 M 个项目和 N 个员工。做项目 i 可以获得 Ai 元，但是**必须**雇用若干个指定的员工。雇用员工 j 需要花费 Bj 元，且一旦雇用，员工 j 可以参加多个项目的开发。问经过合理的项目取舍，最多能挣多少钱。（1 <= M, N <= 100）

【建模方法】
注意到题目中加粗的两个字“必须”，此题是典型的“**蕴含式最大获利问题**”，套用解决最大权闭合子图的建模方法即可解决。每个项目 i 作为一个点并连边(s, i, Ai)，每名员工 j 作为一个点并连边(j, t, Bj)，若项目 i 需要雇用员工 j 则连边(i, j, ∞)。设最小割为 ans，那么 ΣAi-ans 即为结果。

发散思维：
(1) 若进一步要求找出做了哪些项目以及雇用了哪些员工，该如何做？
从源点开始作一次 DFS 并对扫描过的点进行标记，那么被标记的项目和员工即为所求。
(2) 若再进一步要求项目数和员工数的总和最小，该如何做？
待解决。
(3) 如果因做项目 i 而雇用员工 j，则员工 j 只能参加项目 i 的开发，若要做另一个需要雇用员工 j 的项目 k，则需要重新付给员工 j 费用 Bj。该如何求解？
由于这样一来各个项目之间在雇用员工的意义上是相互独立的，直接贪心即可。

《HOJ 2713　Matrix1》

【题目大意】
一个 N*M 的网格，每个单元都有一块价值 Cij 的宝石。问最多能取多少价值的宝石且任意两块宝石不相邻。（1 <= N, M <= 50, 0 <= Cij <= 40000）

【建模方法】
经典的**最大点权独立集问题**。转化为**最小点权覆盖集**：先将网格黑白染色，从源点到每个黑点有一条边，从每个白点到汇点有一条边，容量均为相应宝石的价值。每个黑点向与其相邻的四个白点连边，容量为∞。设最小割为 ans，结果即为 ΣCij – ans。

《POJ 1815　Friendship》

【题目大意】
现代社会人们都靠电话通信。A 与 B 能通信当且仅当 A 知道 B 的电话号或者 A 知道 C 的电话号且 C 与 B 能通信。若 A 知道 B 的电话号，那么 B 也知道 A 的电话号。然而不好的事情总是会发生在某些人身上，比如他的电话本丢了，同时他又换了电话号，导致他跟所有人失去了联系。现在给定 N 个人之间的通信关系以及特定的两个人 S 和 T，问最少几个人发生不好的事情可以导致 S 与 T 无法通信并输出这些人。如果存在多组解，输出字典序最小的一组。（2 <= N <= 200）

【建模方法】
此题是求一个最小点割集，但是要求输出方案且要字典序最小。首先仍是拆点构图，求一次最小割记为 ans。之后我们采取贪心的策略，从 1 至 N 枚举删点，如果最小割不变，说明该点不可能在最小割中，我们再把该点加入到网络中；否则最小割一定变小了，用这个较小值更新 ans，记录该点是一个解并不再将其放回。重复这个过程，把所有点都扫描一遍后结果就出来了。

《ZOJ 2532　Internship》

【题目大意】
有 N 个城市，M 个中转站以及 L 条有向边(u, v, c)，表示可以从 u 向 v 传送信息，带宽为 c。每个城市都在向 CIA 总部发送无穷大的信息量，但是目前总部实际接收带宽已经不能满足要求。CIA 决定要增大某条边的带宽以增大总部的接收带宽，请找出哪些边带宽的增加能导致总部接收带宽的增加。（1 <= N+M <= 100, 1 <= L <= 1000）

【建模方法】
此题要求找出这样一条边，增加它的容量可以导致最大流的增加。最直观的做法是先求一次最大流记为 ans，然后枚举每条边的容量加 1 再求最大流，看是否大于 ans。但是这样做复杂度太高。有没有更好的办法？如果我们换个角度，考虑求完最大流后残量网络 R 中的割，那么问题便迎刃而解。最大流的含义是我们现在在 R 中找不到一条从 s 走到 t 的增广路。我们只要找到这样的割边(u, v)，使得 R 中从 s 能走到 u，从 v 能走到 t，那么当我增加这条割边的容量后，它就会再次出现在 R 中，架起一条增广路 s-u-v-t，从而增加最大流的流量。再正式描述一下算法流程：求一次最大流，然后在残量网络 R 中以正向弧对 s、以正向弧的逆对 t 作 DFS，并用 from[i], to[i]标记点 i 能否从 s 可达，能否可达 t。扫描每一条正向弧(u, v)，若它残留容量为 0 且 from[u]且 to[v]，则(u, v)为一个解。

发散思维：

(1) 如何判定网络的最小割是否唯一？
同样求一次最大流后在残量网络 R 中以正向弧对 s、以正向弧的逆对 t 作 DFS，只不过这次只用一个 flag[i]标记点 i 是否在两次 DFS 中被探访过。最后扫一遍所有点，如果存在没有被探访过的，则说明最小割不唯一。

《Ural 1277　Cops and Thieves》

【题目大意】
一个犯罪团伙打算去偷一家美术馆。警察决定派 K 个人堵住所有从匪窝通向美术馆的道路，不过他们只能驻守在沿途顶点处而不能在匪窝或美术馆，且每个点都有一个需要警察驻守的最低人数 Ri。问警察能否完成任务。（2 < N <= 100, 1 < M <= 10000）

【建模方法】
此题是**无向图点带权的点连通度问题**。将每个点 i 拆成两个点 i', i''，除匪窝 s 和美术馆 t 外加边(i', i'', Ri)，将每条无向边(i, j)分解为(i'', j'), (j'', i')。令 s''为源，t'为汇，求一次最小割即为结果。（注意此题还需要特判匪窝和美术馆在同一点的情况）

发散思维：
(1) 如果警察只能驻守在边上，该如何做？
直接以原图为网络求最小割即可。
(3) 如何求**无向图的边连通度**？
任选一点固定为源，枚举汇点求最小割，取最小值即为所求。
(3) 如何求**无向图的点连通度**？
令本题中的 Ri = 1，以度最小的顶点为源，枚举汇点求最小割，取一个最小值即为所求。

《SPOJ 839　Optimal Marks》

【题目大意】
给出一个无向图，每个点有一个标号 mark[i]，不同点可能有相同的标号。对于一条边(u, v)，它的权值定义为 mark[u] xor mark[v]。现在一些点的标号已定，请决定剩下点的标号，使得总的边权和最小。（0 < N <= 500, 0 <= M <= 3000, 0 <= mark[i] <= 2^31-1）

【建模方法】

先规范化问题，不妨设所有点一定和已知标号的点连通。否则，令那些不与已知标号点连通的点的标号为 0，这些点间的边权也不必计入目标函数（边权都为 0）。在问题的优化式子中，发现异或操作不好处理，难以转化为一些基本运算。深入分析异或操作的本质我们会发现，各个二进制位间是互不影响的。所以我们可以分别处理各个二进制位，最后再汇总即可。处理每个二进制位时构图如下：保留原始标号，将每个点的最优标号置 0；对每个已经有原始标号的点 i，若 i 的该位是 1 则连边(s, i, ∞)，否则连边(i, t, ∞)；对原图中的每条边(i, j)，连边(i, j, 1), (j, i, 1)。求一次最小割后，对源点 DFS，所有被探访到的点，将其最优标号中的该位置 1 即可。

《SPOJ 1693　Coconuts》

【题目大意】
N 个城堡守卫正在就非洲的燕子能否搬运椰子而进行投票（他们怎么这么无聊-_-b）。每个人都有自己的看法，但是为了避免跟自己的朋友持相反意见，他们时常会投相反的票。现在给出每个人的初始看法以及朋友关系，求在某种投票方案下，违背自己意愿的票数与持不同意见的朋友对数的总和最小。（2 <= N <= 300, 1 <= M <= N(N-1)/2）

【建模方法】
此题是经典的“二者取其一式问题”。每名守卫 i 作为一个点，若他投赞成票，则加边(s, i, 1), (i, t, 0)，否则加边(s, i, 0), (i, t, 1)（设最小割中与源 s 连通的点表示赞同）；若 i 跟 j 是朋友，则加边(i, j, 1), (j, i, 1)。求一次最小割即为结果。

发散思维：
(1) 如何求这个函数的最小值？
E(x1, x2, …, xn) = ∑(1-xi)*|pi-v0| + ∑xi*|pi-v1| + ∑|xi-xj|*|pi-pj|, pi, v0, v1 均为常数，xi∈{0, 1}
且不管这个函数表达什么意思，光是分析它的结构我们就能发现这是一个“二者取其一式问题”：把前两项合起来考虑就会发现，要么 xi=0 并获得一个|pi-v0|，要么 xi=1 并获得一个|pi-v1|；考虑第三项，如果 xi, xj 取不同值时获得一个|pi-pj|。与这个模型惊人的相似！构图思路和之前一模一样，求一次最小割即为结果。

《Beijing Regional 2008 Problem A　Destroying the bus stations》

【题目大意】
给定一个无权有向图 G=(V, E)和源点 s、汇点 t，问最少去掉几个点（连同与其关

联的所有边）使得不存在长度小于等于 K 的 s-t 路 （0 < N <= 50, 0 < M <= 4000, 0 < K < 1000）

【建模方法】
此题与《ZOJ 2760　How Many Shortest Path》非常相似，区别在于边不带权且求的是最小点割。同样，分别从源点和汇点作一次 BFS，为了阻断所有长度小于等于 K 的 s-t 路，我们必须将所有满足 ds[u]+dt[u]<=K 的点都考虑在内而不用去管 ds[u]+dt[u]>K 的点，因为只要某一条 s-t 路经过了该点，它的长度就一定大于 K，不删这个点这条路照样满足要求。那么建模的方法也就呼之欲出了：在网络中加入满足 ds[u]+dt[u]<=K 的点 u，将其拆成两个点(u', u'')，　加边(u', u'', 1)。若原图中有边(u, v)且 u, v 均在网络中，则加边(u'', v', ∞)。求一次最小割即为结果。

《小结》

最小割是最大流的对偶问题，但在实际建模过程中，它不如最大流来的直观，模型也往往隐蔽得很深，不容易找到构图方法。通过做大量的题，我总结出以下几种构图方法：

(1) 用 s-t 割集表示方案
这是最小割最基本的构图法，每一个 s-t 割集都实实在在地对应着实际问题中的一种操作方案，比其他应用好理解。

(2) 蕴含式最大获利问题
个人认为这是最小割最 NB 的应用。这类建模方法有一个明显的特点，就是所谓的“蕴含式 A --> B”，表示“要做 A 必须也得做 B”。就是这么一个简单的蕴含关系，造就了最小割这个 NB 的应用。

(3) 二者取其一式问题
个人认为这是最小割次 NB 的应用。这类建模方法也有一个明显的特点，就是每个点都可以有两种方案供选择，每种方案都有一个花费，必须且只能选择其中一种；另外如果某两个点选择了不同的方案，还要在这两个点之间增加额外的费用。这种应用其实也可算作第一种构图法，只不过它的特点更明显，所以我把它单列出来。

(4) 待补充。

有上下界

《POJ 2396　Budget》

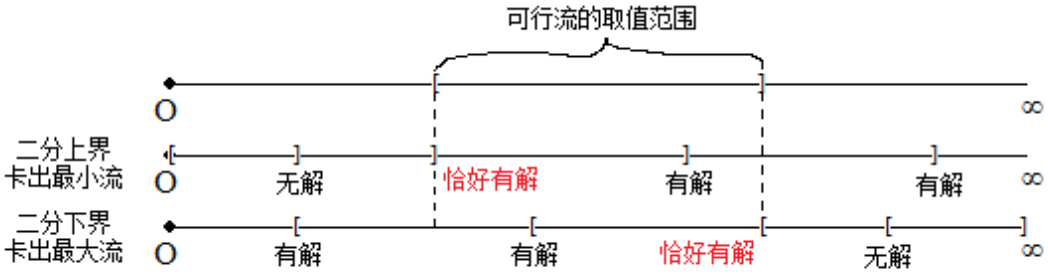
【题目大意】
一个 M*N 的矩阵，给定每行、每列的元素之和 Ri、Cj 以及 c 个满足下列形式的约束条件：i j {< = >} k（i, j 表示第 i 行第 j 列的元素，为 0 表示整列或整行），问是否存在满足所有约束条件的矩阵。（1 <= M <= 200, 1 <= N <= 20, 0 <= c < 1000）

【建模方法】
经典的**有源汇有上下界网络的可行流问题**。

第一步：建立无源汇有上下界的网络模型
每行 i 作为一个点并连边(s, i, Ri, Ri)，每列 j 作为一个点并连边(j, t, Cj, Cj)，设 Uij, Lij 分别表示第 i 行第 j 列元素的上下界，初始时设 Uij=∞, Lij=0。按照给定的约束条件不断调整 Uij, Lij，若出现 Lij > Uij 的情况则已经不存在合法解。对所有元素加边(i, j, Lij, Uij)。另添加边(t, s, 0, ∞)消去原网络的源汇。

第二步：转化为最大流模型
新建源 s'和汇 t'，对每条下界大于 0 的边(i, j, Lij, Uij)，加边(i, t', 0, Lij), (s', j, 0, Lij)。若新网络中最大流等于所有下界之和，则原网络存在可行流，即存在满足所有约束条件的矩阵。

发散思维：
(1) 怎样求无源汇有上下界网络的可行流？
由于有源汇的网络我们先要转化成无源汇，所以本来就无源汇的网络不用再作特殊处理，其他操作同第二步。
(2) 怎样求无源汇有上下界网络的最大流、最小流？
一种简易的方法是采用二分的思想，不断通过可行流的存在与否对(t, s)边的上下界 U, L 进行调整。求最大流时令 U = ∞并二分 L；求最小流时令 L = 0 并二分 U。道理很简单，因为可行流的取值范围是一段连续的区间，我们只要通过二分找到有解和无解的分界线即可。见下图：



《小结》

有上下界网络的可行流问题并不多，目前也只找到不超过 5 道题。它的构图法比较简单直观，也比较单一，故考察不多。前面已经详细讨论了做法，这里不再列举。

最小费用流

《HOJ 2543　Stone IV》

【题目大意】
在无向图 G 中，wywcgs 要从源点 s 购买一些石头并运到汇点 t。每块石头单价是 P 元。每条边 i 有一个初始容量 Ci，当容量超过 Ci 时，每增加一单位容量要额外花费 Ei 元。wywcgs 现在手头只有 C 元，问他最多能买多少块石头并成功运到目的地。（2 <= N <= 1000, 1 <= M <= 10000, 1 <= C <= 100,000,000, 1 <= P <= 10000, 0 <= Ci, Ei <= 10000）

【建模方法】
此题是吴教主出的经典**凸费用流问题**，费用随流的增加呈分段线性状态，可以通过加边的手段转化为最小费用流进行求解。只需将原图中的每条边(u, v)拆成两条：(u, v, Ci, 0), (u, v, ∞, Ei)即可。利用连续最短路算法，每次找一条最小费用路并尽可能多的增广，直到剩余钱数不够再运一个石头为止。

发散思维：

(1) 如果费用函数不是分段线性的，该如何做？

枚举流量为 0, 1, 2, …，计算对应的费用 cost[0], cost[1], cost[2], …，然后依次加入容量为 1，费用为 cost[0], cost[1]-cost[0], cost[2]-cost[1]-cost[0], …的边。最小费用流以及费用函数的凸性保证了流一定会先充满费用较小的边，再流向费用较大的边，而这些费用的和正好等于流量为 x 时的总费用，是正确的流。具体可以参考《剪刀石头布》这道题。

《HOJ 2715　Matrix3》

【题目大意】

一个 N*N 的网格，每个单元都有一个价值 Vi 的宝物和一个高度 Hi。现在 ZhouGuyue 要作至多 K 次旅行，每次旅行如下：他可以借助 bin3 的直升机飞到任意一个单元，之后他每次只能向相邻的且高度比当前所在格子低的格子移动。当他移动到一个边界的格子上时，他可以跳出这个网格并完成一次旅行。旅行中所到之处的宝物他可以全部拿走，一旦拿走原来的格子里就没有宝物了。问他最多能拿走价值多少的宝物。（1 <= N <= 50, 0 <= K <= 50, 0 <= Vi <= 10000）

【建模方法】

将每个格子 i 拆成两个点 i’, i’’并加边(i’, i’’, 1, -Vi), (i’, i’’, ∞, 0), (s, i’, ∞, 0)；对相邻的四个格子 j，若 Hi > Hj 则加边(i’’, j’, ∞, 0)；若格子 i 在边界上则加边(i’’, t, ∞, 0)。限制增广次数小于等于 K 求最小费用流即可。

《HOJ 2739　The Chinese Postman Problem》

【题目大意】

带权有向图上的中国邮路问题：一名邮递员需要经过每条有向边至少一次，最后回到出发点，一条边多次经过权值要累加，问最小总权值是多少。（2 <= N <= 100, 1 <= M <= 2000）

【建模方法】

若原图的基图不连通，或者存在某个点的入度或出度为 0 则无解。统计所有点的入度出度之差 Di，对于 Di > 0 的点，加边(s, i, Di, 0)；对于 Di < 0 的点，加边(i, t, -Di, 0)；对原图中的每条边(i, j)，在网络中加边(i, j, ∞, Dij)，其中 Dij 为边(i, j)的权值。求一次最小费用流，费用加上原图所有边权和即为结果。

若进一步要求输出最小权值回路，则对所有流量 fij > 0 的边(i, j)，在原图中复制

fij 份，这样原图便成为欧拉图，求一次欧拉回路即可。

发散思维：

(1) 如何求**带权无向图上的中国邮路问题**？

若原图不连通则无解。若所有点的度均为偶数，则存在欧拉回路，所有边权加和即为结果。否则必有偶数个奇度点，将这些奇点拉出来构建新图，任意两点之间的边权为原图中两点之间的最短距离，对新图求一次一般图的最小权完美匹配，加上原图所有边权和即为最终结果。

若进一步要求输出最小权值回路，则将每条匹配边在原图中对应的最短路上的每条边都额外增加一条，这样原图便成为欧拉图，求一次欧拉回路即可。

《POJ 3680　Intervals》

【题目大意】

给定 N 个带权的开区间，第 i 个区间覆盖(ai, bi)，权为 wi。现在要你挑出一些区间使得总权值最大，并且满足实轴上任意一个点被覆盖不超过 K 次 （1 <= K <= N <= 200, 1 <= ai < bi <= 100,000, 1 <= wi <= 100,000）

【建模方法】

经典构图题。先将所有区间端点离散化到整数 1..M，另加源 s=0，汇 t=M+1；对每个点 i (0 <= i <= M)加边(i, i+1, K, 0)；对每个区间(ai, bi)加边(ai', bi', 1, -wi)，其中 ai', bi'分别表示 ai, bi 离散化后对应的数值。求一次最小费用流再取反即为结果。

《SPOJ 371　Boxes》

【题目大意】

N 个盒子围成一圈。第 i 个盒子初始时有 Ai 个小球（∑Ai <= N）。每次可以把一个小球从一个盒子移到相邻的两个盒子之一。问最少需要移动多少次，使得每个盒子中小球的个数不超过 1。（1 <= N <= 1000）

【建模方法】

一个比较直观的想法是每个盒子 i 作为一个点。若 Ai > 1 则连边(s, i, Ai-1, 0)；若 Ai = 0 则连边(i, t, 1, 0)。对任意两个盒子 i, j，若 Ai > 1 并且 Aj = 0，连边(i, j, ∞, min(|i - j|, n - |i - j|))。求一次最小费用流即为结果。但是这样构图复杂度会很高，边数会达到 O(N^2)，不够聪明。更加简洁的方法是直接由每个盒子向与其相邻的两个盒子连边(i, j, ∞, 1)，总共也才 2N 条，将边数降到了 O(N)，由 TLE 变成 AC。

《BASHU 2445　餐巾问题》

【题目大意】
一家餐厅在相继的 N 天里，每天要用 ri 块餐巾。餐厅可以选择购买新的餐巾，每块费用为 p 分；也可以选择把旧餐巾送到快洗部，洗一块需 m 天，花费 f 分；或者送到慢洗部，洗一块需 n(n>m)天，费用为 s 分。求合理餐巾使用计划下可能的最小总花费。（1 <= N <= 1000, 1 <= m, f, n, s <= 60）

【建模方法】
经典构图题。将每一天拆成两个点 i, i’，加如下 6 条边：
(s, i, ri, p)——在第 i 天可以买至多 ri 个餐巾，每块 p 分；
(i, t, ri, 0)——第 i 天要用 ri 块餐巾；
(s, i’, ri, 0)——第 i 天用剩的 ri 块旧餐巾；
(i’, i+m, ∞, f)——第 i 天的旧餐巾送到快洗部，每块 f 分；
(i’, i+n, ∞, s)——第 i 天的旧餐巾送到慢洗部，每块 s 分；
(i’, i’+1, ∞, 0)——第 i 天的旧餐巾可以留到第 i+1 天再处理；
求一次最小费用流即为结果。

《剪刀石头布》

——很棒的一道题！

【题目大意】
竞赛图中一些边已经给定，另一些边需要你去定向。问合理安排这些边的方向后，长为 3 的环最多能有多少（一个长为 3 的环叫做“剪刀石头布”现象）。（3 <= N <= 100）

【建模方法】
此题运用了“补集转化思想”。考虑集合{a, b, c}不构成剪刀石头布的情况，那么三个点中一个入度为 2，一个出度为 2，一个入度出度均为 1。不妨以入度为 2 的点作思考，那么总的非剪刀石头布的情况数便为∑C(indegree[i], 2)，剪刀石头布的情况数便为 S = C(N, 3) - ∑C(indegree[i], 2) = N(N-1)(N-2)/6 – M/2 – (∑indegree[i]^2)/2。注意到 N(N-1)(N-2)/6 – M/2 是常数，那么要想让 S 最大，只要(∑indegree[i]^2)/2 最小就可以了。这里产生了凸费用函数 f(x) = x^2，如何转化为最小费用流模型呢？每条待定向的边 i 作为一个点并加边(s, i, 1, 0)；每个点 j 作为一个点并加边(j, t, 1, 1), (j, t, 1, 3), (j, t, 1, 5), …，边数上界视情况而定，但不会超过 N - 1 条；每条边 i 向其两个顶点 u, v 连边(i, u, 1, 0)。求一次最小费用流即可。

《小结》

费用流较最大流更加灵活多变，更需要多多 A 题多多体会。由于变化太多，这里不作过多讨论。