网络流以及一些题目

1. 网络流初步介绍
2. 最大权闭合子图
3. 平面图与对偶图
4. 费用递增网络流
5. 距离限制网络流
6. 区间选择网络流
7. 有上下界网络流
8. 二分图与费用流

绵阳南山中学 曾国洋

网络流初步介绍

网络流初步介绍

- 介绍
  - ✓ 网络流
  - 网络
  - 网络的流
  - 性质
- 算法
- 题目

什么是网络流：

图论中的一种理论与方法，研究网络上的一类最优化问题。

——百度百科

概括的说，网络流就是一个与图论有关的东西。

网络流初步介绍

- 介绍
  - 网络流
  - ✓ 网络
  - 网络的流
  - 性质
- 算法
- 题目

什么是网络：

假设G(V,E)是一个有限的有向图，它的每条边(u,v)∈E都有一个非负值实数的容量c(u,v)。如果(u,v)不属于E，我们假设c(u,v)=0。我们区别两个顶点：一个源点s和一个汇点t。

——百度百科

网络流初步介绍

- 介绍
  - 网络流
  - ✓ 网络
  - 网络的流
  - 性质
- 算法
- 题目

什么是网络：

1. 有向图
2. 图中每条边存在容量
   1. 容量非负
   2. 容量可以是实数
3. 存在源点S、汇点T

网络流初步介绍

- 介绍
  - 网络流
  - 网络
  - ✓ 网络的流
  - 性质
- 算法
- 题目

什么是网络的流：

网络上的流就是由起点流向终点的可行流，这是定义在网络上的非负函数，它一方面受到容量的限制，另一方面除去起点和终点以外，在所有中途点要求保持流入量和流出量是平衡的。

——百度百科

我们能推出的一些性质：

1. 流过一条边的流量不能超过它的容量
2. 边(u,v)的流量一定和边(v,u)的流量为相反数
3. 除了源点汇点，每一个点流入的流量总和等于流出的流量总和

其中第三条又称为流量平衡

通常我们要求解的网络流问题有三类

1. 最大流
2. 最小割
3. 最小费用最大流

其中，最大流的值等于最小割的值

但是求出最小割所有可行割集是一个困难的问题

所以并不是所有最小割问题都能输出方案

Ford-Fulkerson方法：

Ford-Fulkerson方法也叫增广路方法，是很多网络流方法的基础。

找到一条从S到T的路径，增加路径上的流量并调整残留网络，不断调整直到没有增广路为止。

增广路定理：

网络达到最大流当且仅当残留网络中没有增广路。

显然最多增广O(VE)次即可得到最大流

- 介绍
- 算法
  - Ford-Fulkerson
  - ✓ EK
  - ISAP
  - Dinic
  - 最高标号预流推进
  - 最小费用最大流
- 题目

EK：

复杂度$O(m^2n)$

这个复杂度比较高。

但是我们考虑一种网络，这个网络的最大流比较小，也就是需要增广的次数不多的时候，这个算法的复杂度还是挺优秀的。

ISAP：

复杂度$O(n^2m)$

这是一个不错的复杂度。

实测速度很快，通常达不到复杂度上界

代码短！

Dinic：

复杂度$O(n^2m)$

这是一个不错的复杂度。

实测速度很快，通常达不到复杂度上界

代码不长，对于单位流量图，能达到更好的复杂度。

存在几个优化：

1. 逆向宽搜
2. 当前弧优化

网络流初步介绍

- 介绍
- 算法
  - Ford-Fulkerson
  - EK
  - ISAP
  - Dinic
  - ✓ 最高标号预流推进
  - 最小费用最大流
- 题目

最高标号预流推进：

复杂度$O(n^2m^{1/2})$

没有使用通常的增广路思想

这个复杂度很不错，实践中通常会达到上界

以至于比ISAP和Dinic慢

代码很长

最小费用最大流：

将找增广路的过程换成spfa

复杂度O(mystery)

理论上复杂度很高，实际上运行效率是玄学问题，一般尽力简化模型就能跑过。（实际上存在着一种叫做单纯形法的东西）

注意：

1. 最小费用最大流是在最大流的前提下，费用最小
2. 最小费用最大流的网络中可能存在费用负环但是仍然存在解
3. 不能将spfa换成dijkstra因为图中一定会存在负边（增广算法的反向边），但是可以使用一个叫做Johnson的算法

网络流初步介绍

- 介绍
- 算法
- ✓ 题目

题目：

- 题库 10398 最大流
- 题库 10399 最小费用流
- 题库 10477 最大流加强
- 题库 10732 最大流输路径

最大权闭合子图

什么是闭合图：

在一个图中，我们选取一些点构成集合，记为V，且集合中的出边(即集合中的点的向外连出的弧)，所指向的终点(弧头)也在V中，则我们称V为闭合图。

——《最小割模型在信息学竞赛中的应用》胡伯涛

简单的说，如果一个图的中所有点的出边指向的点都在这个图中，那么这个图就是一个闭合图。

最大权闭合子图

- 介绍
  - 闭合图
  - ✓ 最大权闭合子图
  - 模型
  - 求解
- 例题
- 题目
- 拓展

什么是最大权闭合子图：

就是对于一个顶点标有权值的图，在图中找到一个闭合子图使得子图中顶点权值和最大。

注意，顶点的权值可以为负数，甚至实数。图是有向图，可以存在环。

最大权闭合子图

- 介绍
  - 闭合图
  - 最大权闭合子图
  - ✓ 模型
  - 求解
- 例题
- 题目
- 拓展

模型：

我们可以通过将问题建模为最大权闭合子图，再通过网络流来求解这一问题。

通常，一类带有依赖性的最优化问题可以建模成这样。

从问题建模为最大权闭合子图：

存在一些物品A，每个物品都有自己的收益，可正可负

物品之间还有一些依赖关系，A_i依赖于A_j表示，如果要选择A_i就必须选择A_j。

一个物品可以依赖于多个物品，也可以被多个物品依赖。

我们想选出一些符合条件物品使得收益和尽可能的大

我们可以这样建模：

构造一张有向图，图中点i的权值为物品Ai的收益

对于A_i依赖于A_j，我们可以从i向j建一条有向边，可以发现，如果在闭合子图中选择了i那么j必须被包含在闭合子图中。

对这样的图求最大权闭合子图就是原问题的最优解

最大权闭合子图

- 介绍
  - 闭合图
  - 最大权闭合子图
  - 模型
  - ✓ 求解
- 例题
- 题目
- 拓展

求解：

我们可以将建立好的最大权闭合子图模型进一步向网络流模型转换，并使用最大流算法求解

将最大权闭合子图建模为网络流：

对于一个顶点带有权值的有向图，我们将所有正权值的点，从S向它连边，将所有负权值的点向T连边，边的容量为代价的绝对值。

对于原来有向图上的边(u, v)，我们在新图上建立一条u->v的边，容量为 + ∞

对这个图求S到T的最大流，假设最小割为Mincut

那么最大权闭合子图的权值就等于：

S连出所有边的容量和 - Mincut

证明：

略。

最大权闭合子图

- 介绍
- 例题
  - ✓ 题面
  - 建模
  - 方案
- 题目
- 拓展

航天计划问题：

有n个实验$\{E_1, \ldots, E_n\}$，其中第i个实验E_i的收益为P_i。

有m个仪器$\{I_1, \ldots, I_m\}$，其中第k个仪器需要花C_k的代价购买

每个实验需要依赖一些仪器，一个仪器只用购买一次

求如何才能收益最大，输出方案

最大权闭合子图

- 介绍
- 例题
  - 题面
  - ✓ 建模
  - 方案
- 题目
- 拓展

建模：

将所有实验建模为点，其中点E_i的权值为P_i

将所有仪器建模为点，其中点I_k的权值为C_k

如果E_i依赖于I_k则从E_i向I_k建一条有向边

我们对这个图求最大权闭合子图，就是原问题的最优解。

为了求解这一问题，我们需要进一步将问题建模为网络流。

对于点E_i，我们建立从S到E_i的边，容量为P_i

对于点I_k，我们建立从I_k到T的边，容量为C_k

对于E_i依赖I_k，我们建立从E_i到I_k的边，容量为$+\infty$

最终的答案 $= \Sigma P_i$ - Mincut

方案：

最终我们如何输出方案呢？

考虑网络流模型中，所有S连出的边(S, E_i)

假如这条边满流，我们就不进行实验E_i，否则就进行实验Ei并购买所有相关仪器

最大权闭合子图

- 介绍
- 例题
- 题目
  - CEOI2008 Order
    - ✓ 题面
    - 做法
  - CF#185 div1.E
- 拓展

Order：

有n个项目，完成了某个项目就可以获得相应的收益。

但是每个项目又依赖于一些机器，每个机器可以买，也可以租，价格不同。

买一个机器可以供多个项目使用，而租一个机器只能供一个项目使用。

询问最大收益是多少？

做法：

如果机器不能租，那么这个题目就是一个传统的最大权闭合子图问题。

现在问题就是，如何处理租借？

我们先不考虑租借，将问题建模为网络流模型

我们可以发现，一个合法的流会有三个部分：

① S -> 项目

② 项目 -> 机器

③ 机器 -> T

因此这个图的最小割，一定会割掉三部分中的一部分，假如割掉①是放弃项目，割掉③是购买机器，那么我们可以令割掉②为租借机器，这样最小割就一定会在三者中选择其中一个割掉。

所以我们把②部分的容量改为租借费用即可

最大权闭合子图

- 介绍
- 例题
- 题目
  - CEOI2008 Order
  - CF#185 div1.E
    - ✓ 题面
    - 做法
- 拓展

题面：

有n个点，每个可以是白色或者黑色。可以花v_i的代价改变第i个点的颜色。

有m条件，每个条件都是要求某一些点都是某种颜色。如果满足了第i个

条件可以得到g_i的收益，没有满足则须付出l_i的代价。

求最大收益。

做法：

这个题就是一个普通的最大权闭合子图，我们可以先考虑如何将问题建模为最大权闭合子图问题。

换句话说，我们要考虑如何去构造这个依赖性

具体做法可以自己去思考

二元费用问题：

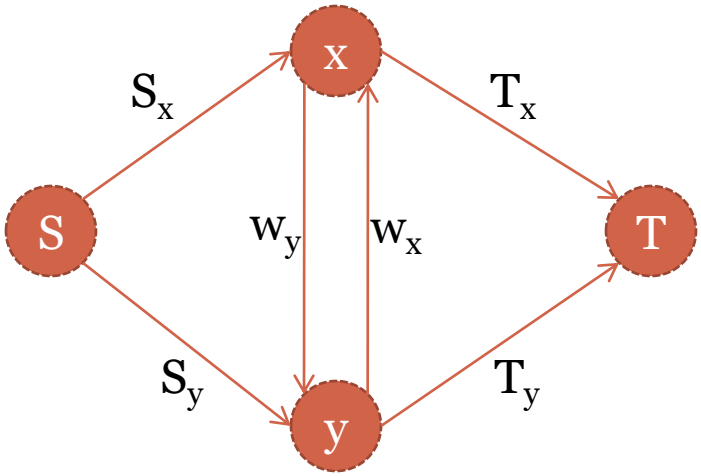
有n个点，选和不选有相应的收益

有m个条件，分为三类：

- 如果x和y都选了，获得额外W_1的收益
- 如果x和y都没选，获得额外W_2的收益
- 如果x和y都一个选了一个没选，付出额外W_3的代价

求最大收益。

二元费用问题：



我们考虑一个形如这样的模型

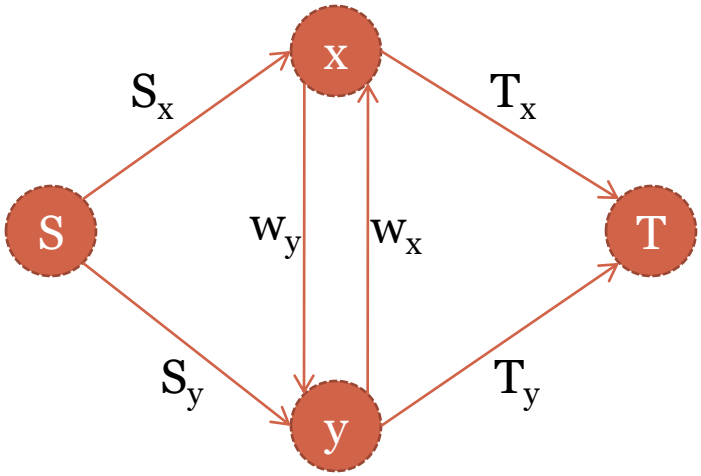
这个图割有很多个，但是可能成为最小割的只有4种情况

1. $\{S_x, S_y\}$
2. $\{S_x, T_y, w_y\}$
3. $\{S_y, T_x, w_x\}$
4. $\{T_x, T_y\}$

最大权闭合子图

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 二元费用问题

二元费用问题：



考虑每种情况：

- x和y都选：此时被割的边为$S_x + S_y$
- x和y都没选：此时被割的边为$T_x + T_y$
- x选y没选：此时被割的边为$S_x + T_y + w_x$
- x没选y选：此时被割的边为$T_x + S_y + w_y$

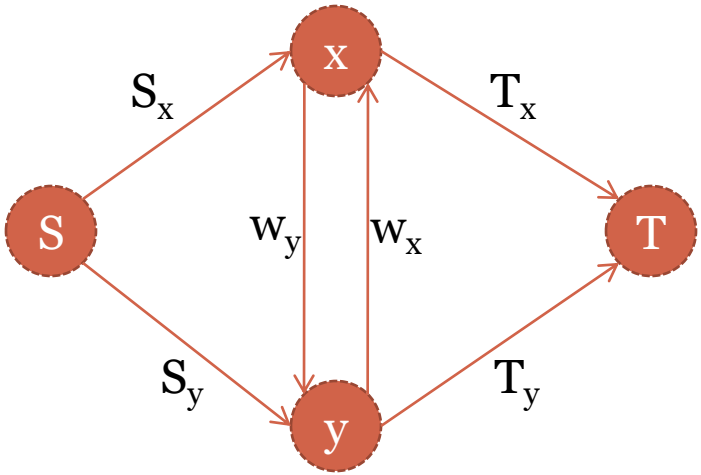
假如我们将被割的边看做损失掉的代价

那么我们可以列出方程

最大权闭合子图

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 二元费用问题

二元费用问题：



$$S_x + S_y = VB_x + VB_y + EB$$
$$T_x + T_y = VA_x + VA_y + EA$$
$$S_x + T_y + w_x = VB_x + VA_y + EA + EB + EC$$
$$T_x + S_y + w_y = VA_x + VB_y + EA + EB + EC$$

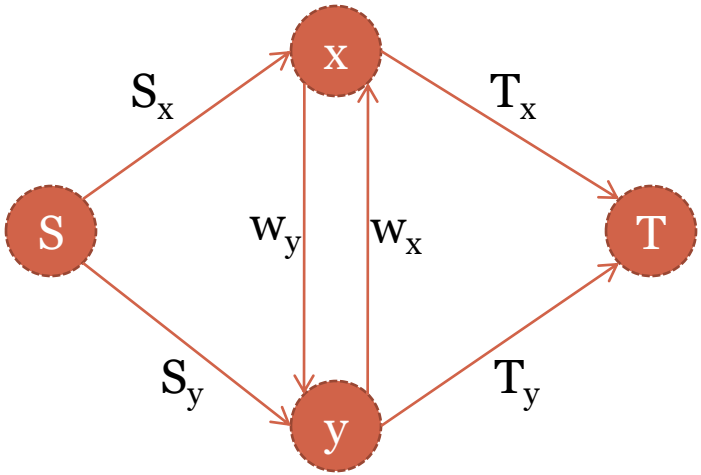
其中

- VA_x表示选x的收益，VB_x为不选x的收益
- x，y都选的收益为EA
- x，y都不选的收益为EB
- x，y一个选一个不选的代价为EC

最大权闭合子图

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 二元费用问题

二元费用问题：



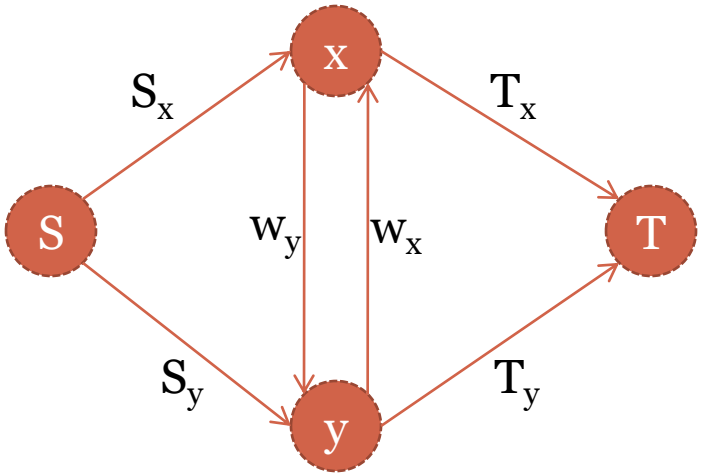
解得：

$$S_x = VB_x + \frac{EB}{2}$$

$$T_x = VA_x + \frac{EA}{2}$$

$$w_x = w_y = \frac{EA + EB}{2} + EC$$

二元费用问题：



假设建成的图的最小割为Mincut

最终的答案为：

$$ans = \sum (VA + VB + EA + EB) - Mincut$$

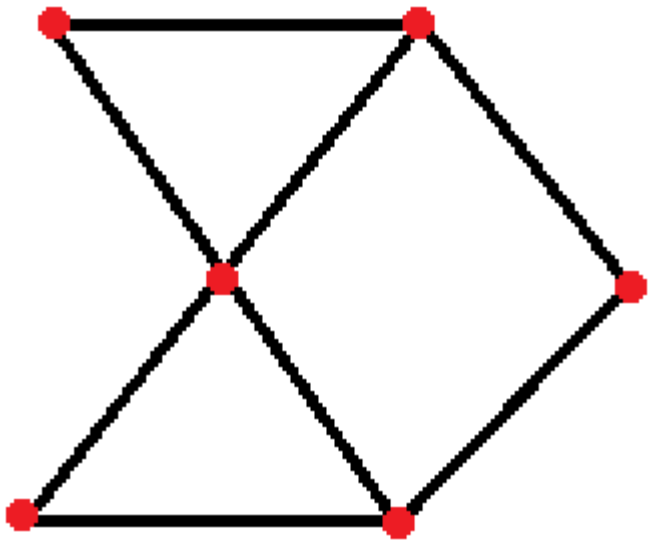
平面图与对偶图

什么是平面图：

平面图是可以画在平面上并且使得不同的边可以互不交叠的图。

——百度百科

举个例子：



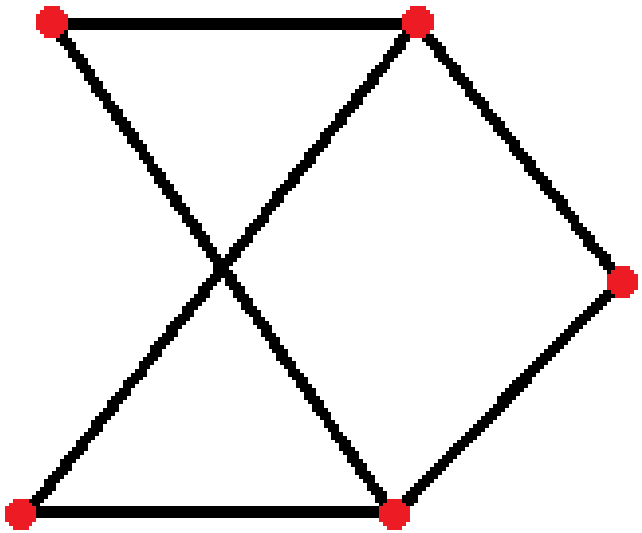
这是一个平面图

什么是平面图：

平面图是可以画在平面上并且使得不同的边可以互不交叠的图。

——百度百科

这个呢？

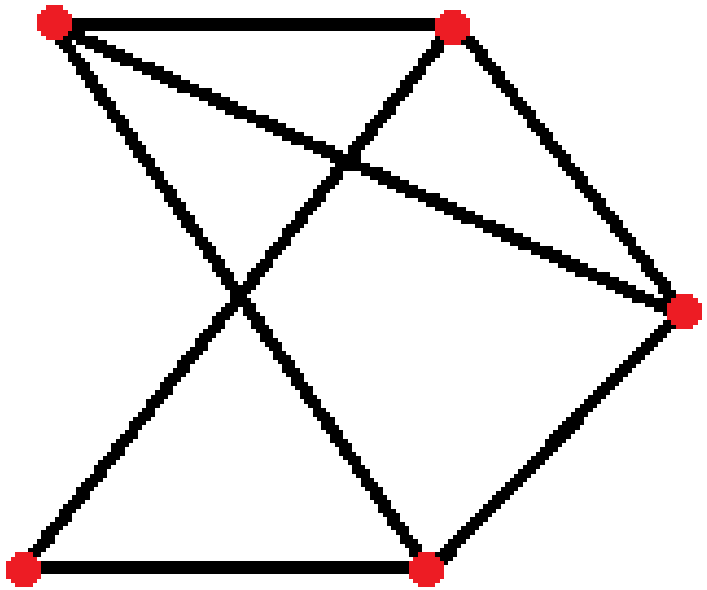


什么是平面图：

平面图是可以画在平面上并且使得不同的边可以互不交叠的图。

——百度百科

这个呢？

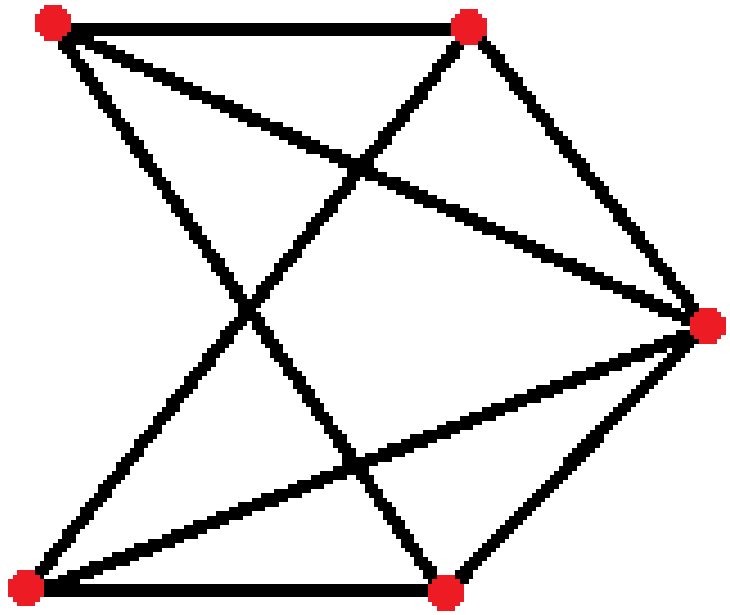


什么是平面图：

平面图是可以画在平面上并且使得不同的边可以互不交叠的图。

——百度百科

这个呢？

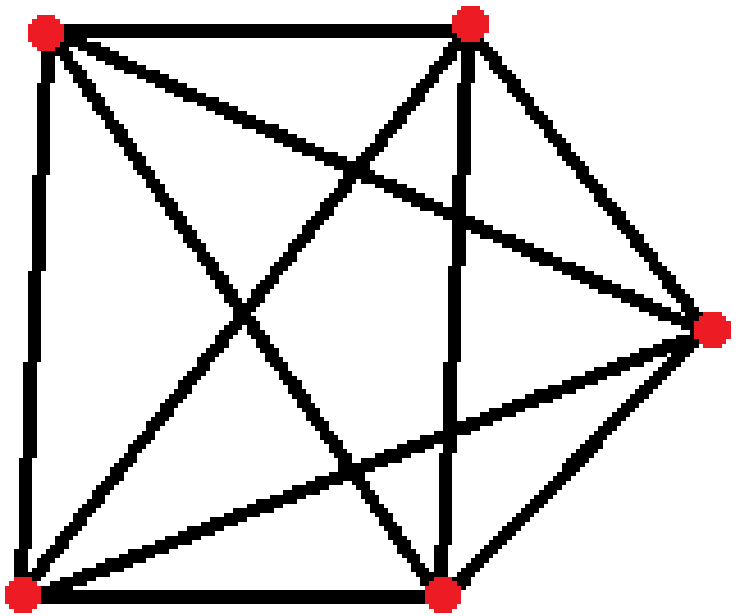


什么是平面图：

平面图是可以画在平面上并且使得不同的边可以互不交叠的图。

——百度百科

这个呢？

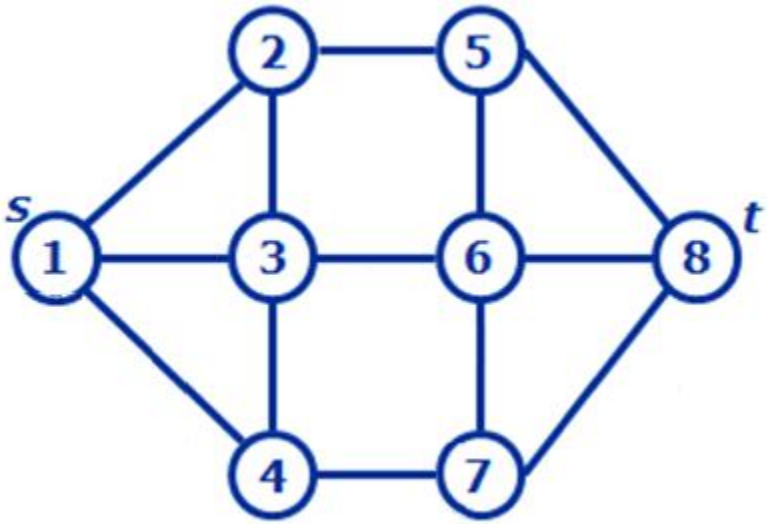


平面图与对偶图

- 介绍
  - 什么是平面图
  - ✓ 什么是对偶图
- 例题
- 题目

什么是对偶图：

对偶图



假如我们要对这样一张图求S到T的最大流

我们会发现：

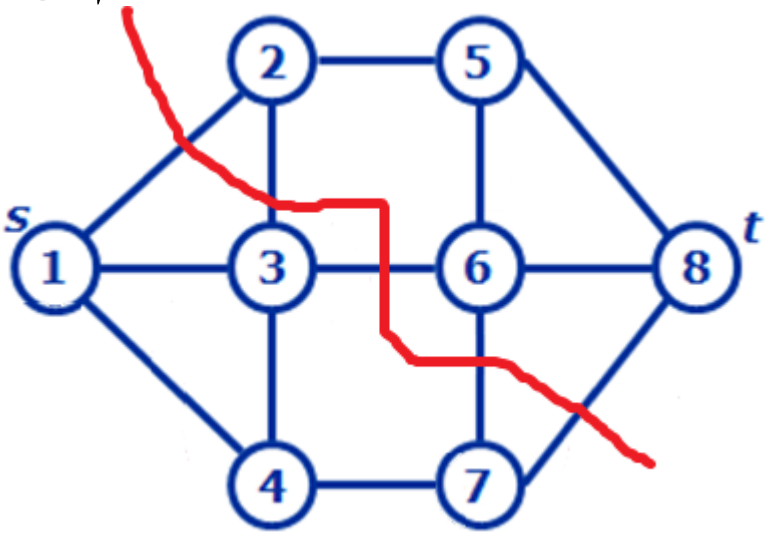
因为最大流的值等于最小割，所以我们可以求这个图的最小割。能够很直观的发现，我们只用在平面上画一条直线将ST，这条线穿过的边就是一种可行割

平面图与对偶图

- 介绍
  - 什么是平面图
  - ✓ 什么是对偶图
- 例题
- 题目

什么是对偶图：

例如这样：

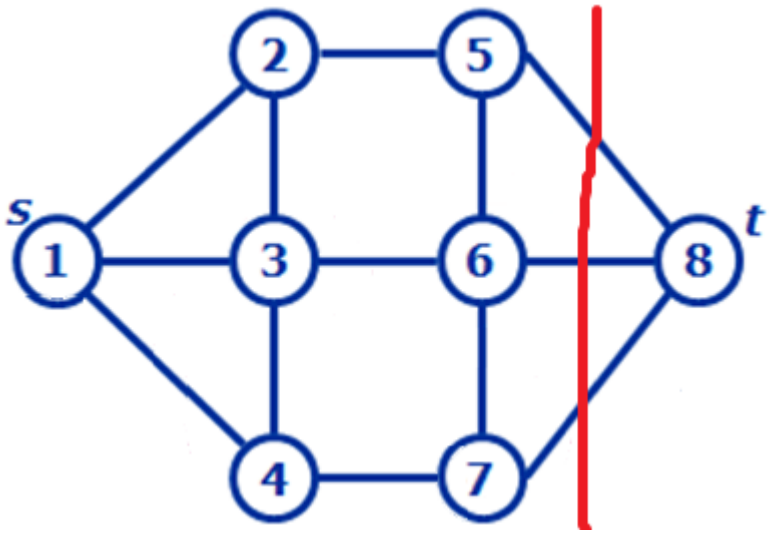


平面图与对偶图

- 介绍
  - 什么是平面图
  - ✓ 什么是对偶图
- 例题
- 题目

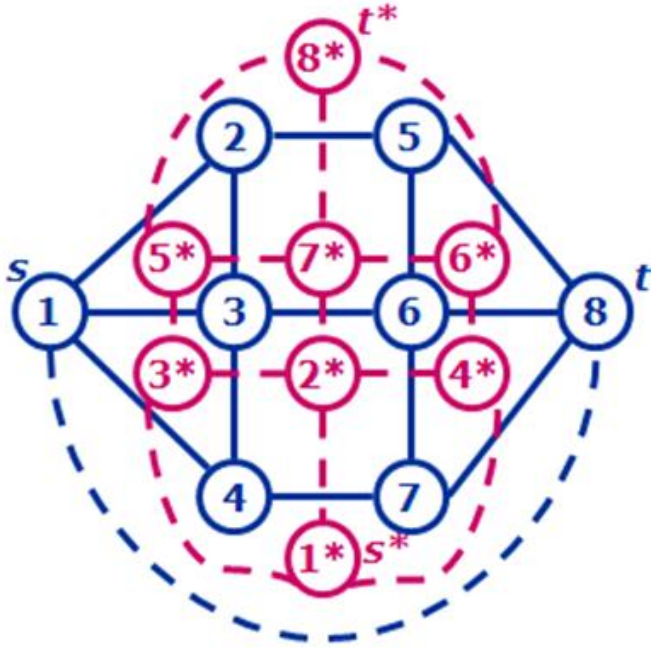
什么是对偶图：

或者这样：



什么是对偶图：

因为我们要割尽量小，所以我们希望这条线穿过的边的和尽量小，当然这就是最短路。



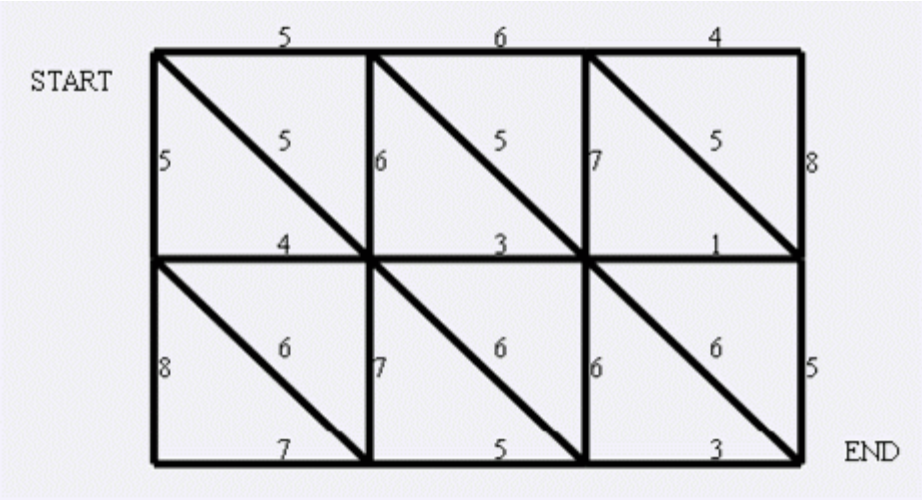
当然，我们能建出这样一个图来，这样的图就叫做对偶图。将面看做点，并在相邻面之间连边。

对对偶图求最短路，相当于对原图求最小割

平面图与对偶图

- 介绍
- 例题
  - ✓ 题目
  - 建模
  - 解法
- 题目

BeiJing2006 狼抓兔子：



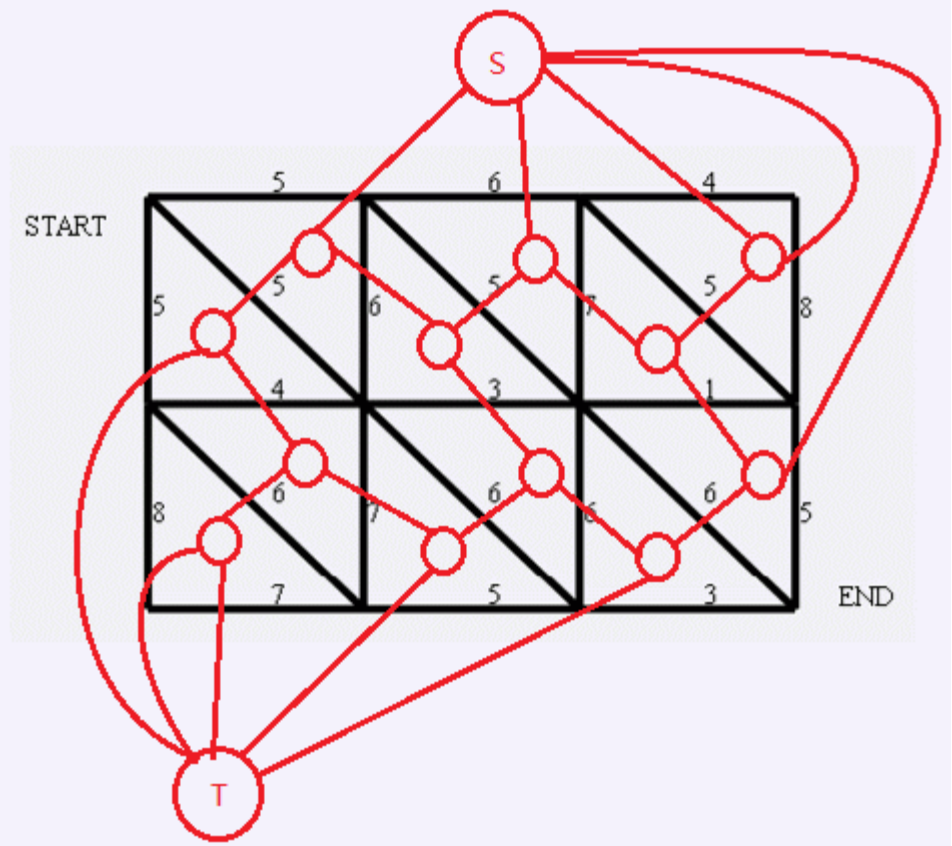
给一张形如上图的网络，求START到END的最小割

数据范围能卡掉网络流算法。

平面图与对偶图

- 介绍
- 例题
  - 题目
  - ✓ 建模
  - 解法
- 题目

建模：



都Too Simple（太简单）

解法：

对我们建出的对偶图求最短路

然后就没了

我相信求解平面图的网络流，对大家没有任何难度……才怪呢

求一个任意平面图的对偶图是一个很麻烦的问题，通常我们会考虑问题的特殊性质来构建对偶图。

当然在这个过程中我们可能用到“平面嵌入”

建立平面图的对偶图时如何确定边的方向？

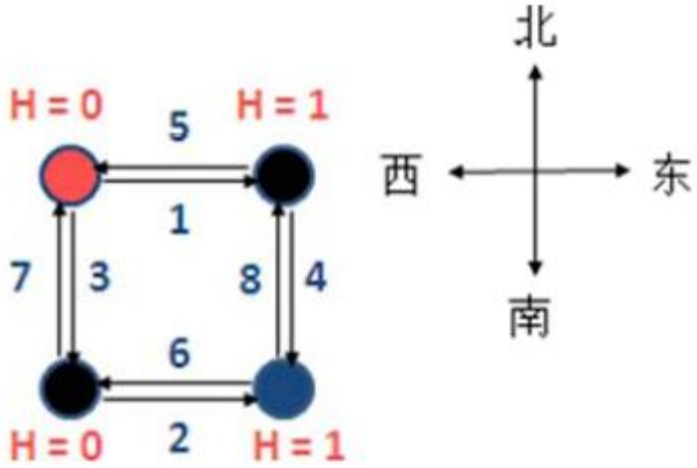
NOI2010海拔：

n×n的网格图，每条边双向通行。已知每条边两个方向的人流量，你需要给每个点x设定一个海拔高度h_x

左上角的高度为0，右下角的高度为1。对于x到y的流量为w的边，代价为$w \times \max(0, h_y - h_x)$

求最小代价。

$n \le 500$。



做法：

显然，最优解中所有点的高度是0或者1

为什么呢？

因为我们总能通过调整使得某个高度为实数的点变成整数且答案不变差

再进一步思考，可以发现，最优解中所有高度为0的点与左上角在一个联通块中，所有高度为1的点与右下角在一个联通块中。

然后我们需要找到这条线将左上角与右下角所述的联通块分开，且使得代价尽量小。

这个问题是一个典型的最小割模型，我们可以直接建模后发现这是一个平面图，然后就可以通过对偶图最短路求解了。

平面图与对偶图

- 介绍
- 例题
- 题目
  - NOI2010海拔
  - CJ #2 2014 C
    - ✓ 题面
    - 做法

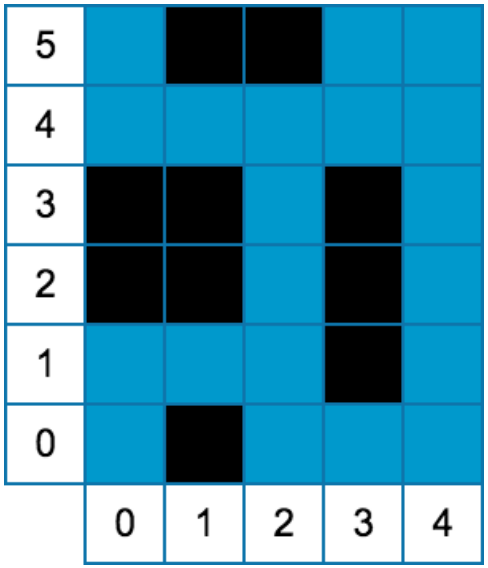
Codejam Round 2 2014 C：

n×m的网格图，底部的每个格子接受1的入流量，顶部的每个格子可以流出1的流量，四连通的格子之间有容量为1的双向边

图中有k个长方形区域是不能走流量的，这些长方形互不相交，但可以有公共边。

求图的最大流

$m \le 1000$，$n \le 10^8$，$k \le 1000$

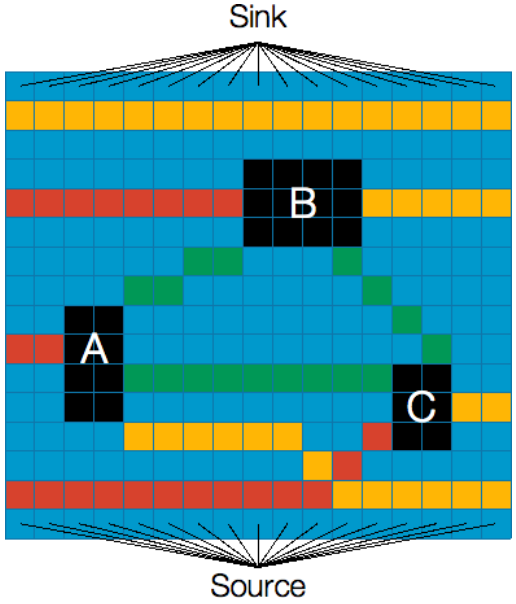


做法：

最大流的值等于最小割的值，通常题目的最大流比较好求，而这个题的最大流并不好求了

但是这道题的最小割好像比较好求

脑补一下，选出尽量少的格子使得顶部和底部分开。



注意这也是平面图

然后就随便做做就行了

费用递增网络流

什么是费用递增的网络流?

通常的费用流，一条边的费用等于：

流量 × 单位流量所需费用

然而有时候有些问题比较奇怪，一条边的费用可能是：

① 流量^2

② 流量^{流量}

③ e^{流量}

④ 斐波那契数列F(流量)

这时候我们应该怎么办呢?

费用递增网络流

- 介绍
  - 什么是费用递增
  - ✓ 解决方法
- 例题
- 题目

解决办法：

假如所有边容量整数的时候，我们有一种很暴力的方法：

> 因为所有边流量都是整数，那么一定存在最优解中所有边的流量一定都是整数，我们将一条容量为c的边拆成c条容量为1的边。由于费用递增，所以最小费用最大流一定会优先选择费用小的边，最终求出的解就是最优解

这样，当所有边容量都是整数的时候，问题就被解决了。

当容量为浮点数的时候怎么办呢？

自己想

费用递增网络流

- 介绍
- 例题
  - ✓ 题面
  - 解法
- 题目

JSOI2009：

有n支球队，有些球队之间已经打了一些比赛了，现给出每个球队的数据win、$lose$、C和D，分别表示已胜场数、已负场数，以及计算收益的两个系数。

一支球队的收益为$Cw^2 + Dl^2$，其中w和l是最后胜负的场数。

接下来还有m场比赛。给出接下来m场比赛的对阵情况，求出n支球队收益和的最小值。

接下来m场比赛的胜负是你可以决定的。

注意是总收益和<u>最小</u>！

费用递增网络流

- 介绍
- 例题
  - 题面
  - ✓ 解法
- 题目

解法：

先假设后来m场比赛双方都输了，然后我们现在的问题就变成了如何将这m场比赛的胜利分配给相应的球队。

当一支球队多胜利一场，代价的变化是：

$$(C(w+1)^2 + D(l-1)^2) - (Cw^2 + Dl^2) = 2wC - 2lD + C + D$$

很显然，这个代价是递增的。因为胜利一场后w会增加，l会减少

所以这是费用递增的网络流，使用先前的方法就可以解决

费用递增网络流

- 介绍
- 例题
- 题目
  - SCOI2007修车
    - ✓ 题面
    - 做法
  - WC2007剪刀石头布

SCOI2007 修车：

有n辆车和m名修理工，一名修理工在一个时刻只能修一辆车，并且在修完一辆之后才能开始修下一辆。

已知每位修理工修每辆车的时间，求顾客的最小平均等待时间。

顾客的等待时间为他的车被修好的时间。

换句话说就是求最小的等待时间总和

费用递增网络流

- 介绍
- 例题
- 题目
  - SCOI2007修车
    - 题面
    - ✓ 做法
  - WC2007剪刀石头布

做法：

我们发现，对于每一个修理工，越后面修的车等待时间越长，这是一个费用递增。

但是这个费用递增我们并不能很好的运用，因为当前等待的时间需要知道在此之前修理了哪些车。当前选择存在一个后效性。

我们换一个思路考虑。

用C_i表示修理工倒数第i个修理的车花费的时间

我们考虑每一辆车对答案的贡献，显然修理工修理的最后一辆车对答案的贡献为C_1，倒数第二辆车的贡献为$2C_2$，倒数第i辆车的代价为iC_i

我们发现，如果我们倒着考虑，那么这个代价是递增的，也就是说费用是递增的

然后问题就相当简单了，暴力建模就行了

WC2007石头剪刀布：

有n个人，两两之间进行一次比赛。有些比赛已经进行了，有些还没有。

我们用一个竞赛图来表示输赢情况，一场比赛的有向边从输家连向赢家。

你可以决定尚未进行的比赛的输赢情况，使得下面这种，三元组的数量最多：

(a, b, c)表示一个无序三元组

满足

有向边$a \to b$、$b \to c$、$c \to a$

需要输出方案

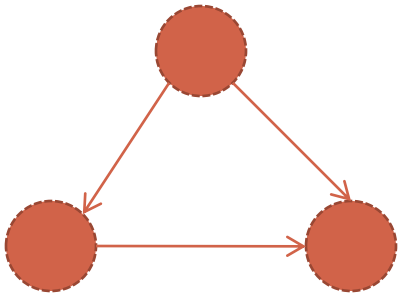
$n \le 100$

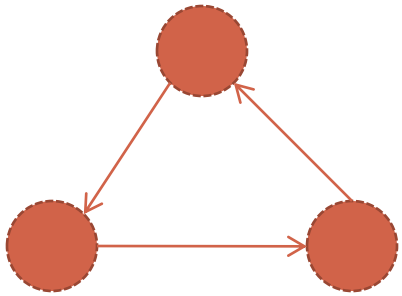
做法：

我们发现我们要最大化的东西比较奇怪，是满足某一条件的三元组。

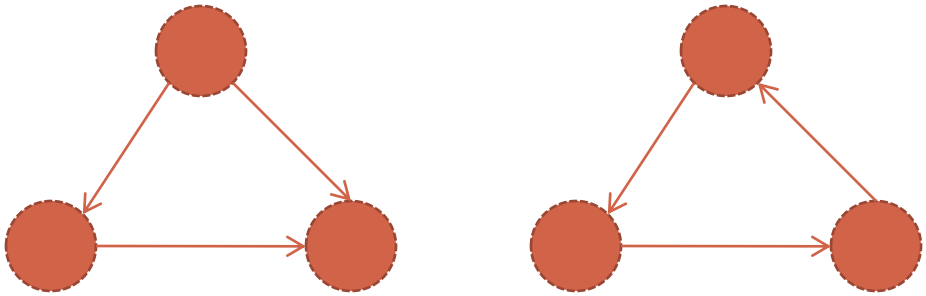
但是这道题的问题具有特殊性，注意这个图中任意两点之间都有边。

我们考虑任意三元组的连边情况，其实只有两种：





做法：



可以发现，不合法的三元组一定存在一个点的出度为2

图中三元组一共有$\binom{n}{3}$个，我们用d_i表示点i的出度，那么不合法的三元组有$\sum\binom{d_i}{2}$个

所以合法三元组的个数为：

$$ans=\binom{n}{3}-\sum_i\binom{d_i}{2}$$

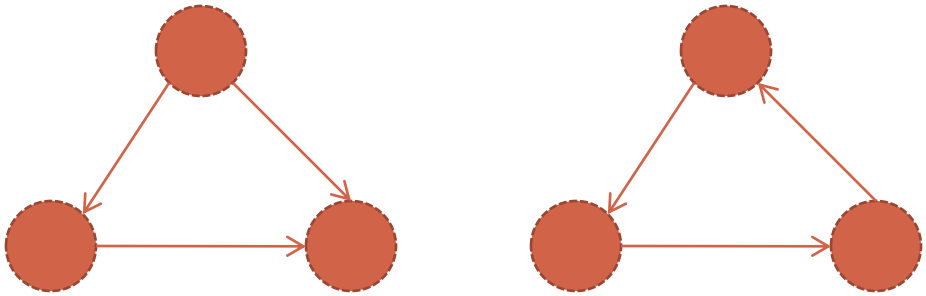
$$ans=\binom{n}{3}-\frac{\sum\left(d_i^2-d_i\right)}{2}$$

$$ans=\binom{n}{3}+\frac{1}{2}\sum d_i-\frac{1}{2}\sum d_i^2$$

费用递增网络流

- 介绍
- 例题
- 题目
  - SCOI2007修车
  - WC2007剪刀石头布
    - 题面
    - ✓ 做法

做法：



$$ans = \binom{n}{3} - \sum_i \binom{d_i}{2}$$

$$ans = \binom{n}{3} - \frac{\sum \left(d_i^2 - d_i\right)}{2}$$

$$ans = \binom{n}{3} + \frac{1}{2}\sum d_i - \frac{1}{2}\sum d_i^2$$

我们发现Σd_i是确定的，因此我们只需要最小化后面的Σd_i^2就行了，而这个问题就是经典的费用递增网络流

距离限制网络流

限制距离网络流

- 介绍
  - ✓ 什么是距离限制
  - 如何建模
- 例题
- 题目

什么是距离限制：

存在一些变量$\{x_1, \ldots, x_n\}$

第i个变量存在一些取值$\{A_{i,1}, \ldots, A_{i,m}\}$

我们用C_i表示第i个变量取值为$A_{i,Ci}$。

用$f(i, Ci)$表示第i个变量取C_i时的代价

现在有一些约束条件

$$Ci - Cj \le d$$

我们希望在满足约束条件的情况下，给变量赋值使得代价和尽量小.

这样的问题就叫做距离限制

限制距离网络流

- 介绍
  - 什么是距离限制
  - ✓ 如何建模
- 例题
- 题目

建模：

用下面的方法建模：

对于每一个变量i

我们将每一个变量拆成m个点对应这个变量的m个取值

我们建立边：

$S \rightarrow A_{i,1}$　　容量 $A_{i,1}$

$A_{i,j} \rightarrow A_{i,j+1}$　　容量$A_{i,j+1}$

$A_{i,m} \rightarrow T$　　容量$+\infty$

对于限制

$$C_i - C_j \le d$$

我们建边

$A_{i,k} \rightarrow A_{j,k-d}$　　容量$+\infty$

建模：

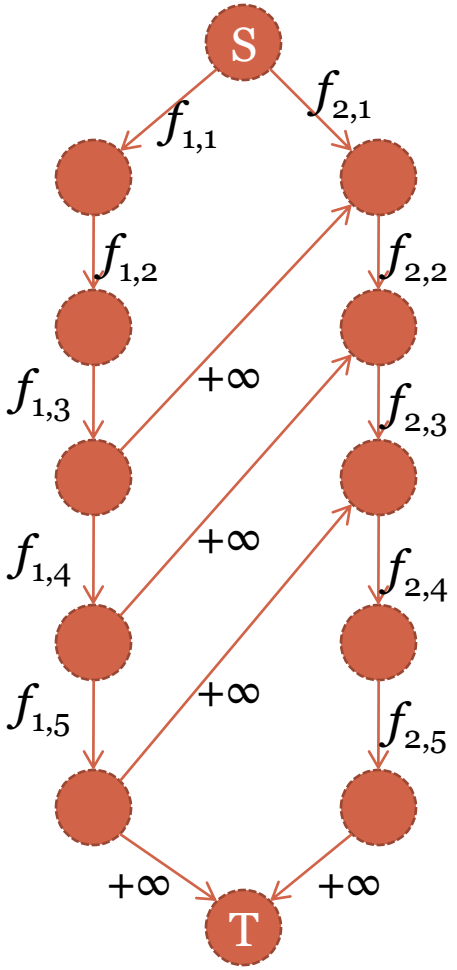
假设当前有两个点，d = 2，我们建的图如下：

我们可以通过观察发现这张图的割

首先最小割不会包含中间那些+∞的边

那么所有可行的割就都满足了限制条件

我们在所有可行割中找到权值最小的，就是最终的答案



限制距离网络流

- 介绍
- 例题
  - ✓ 题面
  - 做法
- 题目

HNOI2013切糕：

$p \times q$的网格，每个位置都有r个取值的选择，每个选择有各自的代价

要求四连通相邻的两个位置的值相差不超过d

问最小代价和

$n, q, r \leq 40$

限制距离网络流

- 介绍
- 例题
  - 题面
  - ✓ 做法
- 题目

做法：

我们将棋盘上每一个格子看做一个变量

每一个格子有r个取值

题目中的条件：相邻两个位置上的数字之差看作约束

这样我们就能根据前面所说的建模方法建出模型了，然后通过网络流求解

看来这道题并没有什么难度

但是并不代表此类方法的题就很简单

限制距离网络流

- 介绍
- 例题
- 题目
  - TC SRM590 D1L3
    - ✓ 题面
    - 做法
  - CTSC移民站迷址

TC SRM590 D1L3 FoxAndCity：

给定n个点的无向图，图上已经有一些边，边权均为1

每个点有一个属性wi

现在可以在图中任意加边

记加边后每个点到1号点的距离为di

最小化

$$\sum (w_i - d_i)^2$$

做法：

这道题的做法并不显然，需要我们对问题的特殊性进行一些挖掘

我们会发现，无论如何添加边，新的图一定满足以下性质：

1. $d_1 = 0$
2. 对于原图中的边(u，v)，新图中也一定有这样的关系：$|d_u - d_v| \le 1$

进一步思考，我们可以证明，满足以上两个条件的图也一定是合法的新图

经过这样的转化，我们可以发现，问题就变成了距离限制的网络流。

对于每一个限制 $|d_u - d_v| \le 1$ 可以拆分成两部分即 $d_u - d_v \le 1$ 和 $d_v - d_u \le 1$，这样就可以做了

限制距离网络流

- 介绍
- 例题
- 题目
  - TC SRM590 D1L3
  - CTSC移民站选址
    - ✓ 题面
    - 做法

CTSC2009 移民站选址：

已有n个移民站，第i个坐标为(ui, vi)

还需建立m个新的移民站，记坐标为(xi, yi)

移民站之间需要传输数据，第i个旧站和第j个新站之间要传a_{ij}的数据，第i个新站和第j个新站之间要传b_{ij}的数据

传输代价是两个站之间传输的数据量乘上两个站之间的曼哈顿距离

求最小代价和

虽然这是提交答案，但是可以当成普通题做

做法：

这个题并不是传统的距离限制模型，但是这个题的做法和距离限制很相似

不过通过题目，我们可以发现很重要的一点：这是个二维的问题，但是两个维度可以拆开考虑

现在我们只用考虑一维

不过只考虑一维也有很多情况，不如只考虑一维时两个点的情况

做法：

在这里，我们考虑这样一个模型：

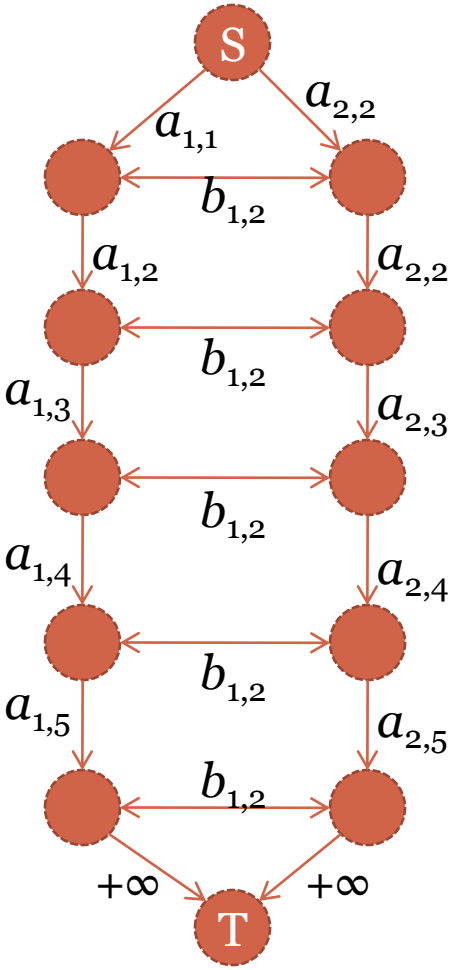
其中$a_{i,j}$表示第i个新站与第j个旧站重合时与所有旧站的代价和

看了这个模型，我们再考虑一下它的最小割

可以发现这个最小割与我们所需要求的问题有关

能不能将这个模型拓展到n个点的情况呢？

自行思考



区间选择网络流

区间选择网络流

- 介绍
  - ✓ 什么是区间选择
  - 模型
- 例题
- 题目

什么是区间选择：

有些时候，我们需要在一个序列上进行抉择，甚至是树上或者图上。

不过这里只讨论序列上的问题，因为序列上的解决方法，通常能够比较容易的推广到树上或者图上（当然也有可能不能推广）

我们如何在网络流中表示选择了一些区间呢？

区间选择网络流

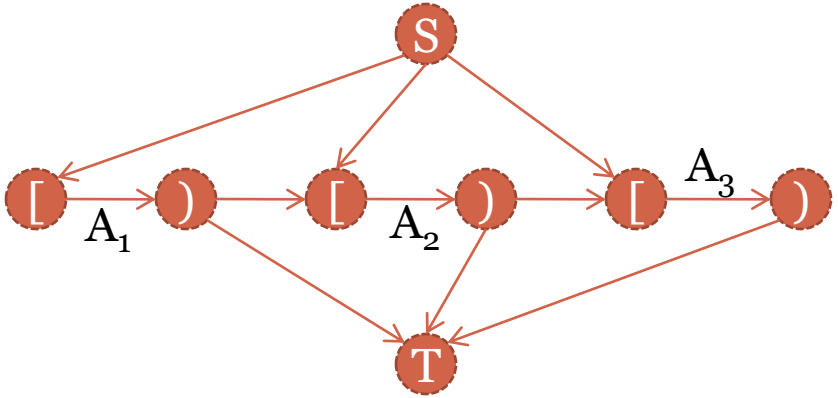
- 介绍
  - 什么是区间选择
  - ✓ 模型
- 例题
- 题目

模型：

假设这里有一个长度为n的序列

在数学中，我们可以使用[L, R)来表示一段左闭右开的区间，当然我们也能在网络流中这样做

我可以用一张图来形象的表示：



我们发现，网络中的每一条流，都是从S出发从一个“[”进入序列，再从一个“)”出去，到达T。

因此，我们就表示出了这个区间

区间选择网络流

- 介绍
- 例题
  - ✓ 题面
  - 做法
- 题目

TCO09 Championship Round D1L2 ArrayTransformations：

给定序列a，记a的元素个数为n

称一次变换为：

选定区间$[i, j]$，满足$1 \le i \le j \le n$，对序列a在区间中的每个数减1，如果已经为0则不操作。这样一次变换的代价为$j-i+1$

对于给定序列a，最大变换次数k和最大代价和m，求：在使用不超过k次变换，总变换代价不超过m的前提下，a中最大元素的最小值是多少。

$n \le 250$

做法：

首先，对于最大值最小问题，我们很容易的会想到使用二分将问题转化为判定性问题

现在我们要解决的就是，给定k次操作和m的最大代价，能否将所有值都变化到p以下，其中p是我们二分的值

我们还可以进一步转化

对于数列上的数字a_i，它需要被操作的次数是已知的，即$max(0, a_i - p)$

那么问题也就变成了：

给定k次操作和m的最大代价，能否找到一种操作方法，使得对于任意一个位置i都被覆盖了至少$max(0, a_i - p)$次

区间选择网络流

- 介绍
- 例题
  - 题面
  - ✓ 做法
- 题目

做法：

我们直接套用刚才的模型，将数列上的每个元素拆成两个点，第i个元素拆为$In[i]$和$Out[i]$，分别对应“[”和“)”

建立点S′和T′

从S向S′连一条边，容量为k

从T′向T连一条边，容量为k

从所有S′向$In[i]$连一条容量为1的边

从所有$Out[i]$向T′连一条容量为1的边

从所有$In[i]$到$Out[i]$连一条容量为+∞，下界为$max(0, a_i - p)$，费用为1的边

从所有$Out[i]$到$In[i+1]$建一条容量为+∞的边

这时，我们求此网络有流量下界的最小费用最大流，再将费用与p比较即可

这样这道题就被解决了，虽然你可能暂时不会带流量下界的网络流

区间选择网络流

- 介绍
- 例题
- 题目
  - CF#172 (Div. 1) D
    - ✓ 题面
    - 做法

CF#172 (Div. 1)D k-Maximum：

给定一个长度为n的序列，有q次操作。

操作有2种：

1. 修改一个元素的值
2. 查询一个区间中的最大k-子段和

什么叫最大k-子段和呢？

就是从区间中选出k个子段，使得k个子段中所有元素之和最大，并且这k个子段不能重合

当$k=1$时就是连续最大和

$n, m \le 10^5$

$k \le 20$

区间选择网络流

- 介绍
- 例题
- 题目
  - CF#172 (Div. 1) D
    - 题面
    - ✓ 做法

做法：

我们会发现，这个题根本就不是一个网络流的题，而是一个数据结构题2333333

是不是因为我放错题了呢？当然不是

这道题的正解需要用到本部分的思想

我们可以考虑对于单次询问如何用网络流来建模处理。

我们能大概建立一个网络流模型

仔细观察费用流每一次增广时，对网络流模型做了哪些改变，我们可以使用数据结构在序列上进行相应的模拟。

最终我们能得到一个单次操作$O(klogn)$的算法

但是因为这里空间太小，留给大家思考

有上下界网络流

有上下界网络流

- 介绍
  - ✓ 什么是上下界
  - 模型
  - 最大流与最小流
- 例题
- 题目
- 拓展

什么是有上下界网络流：

通常我们网络流模型中，对于每一条边上流量的现在都是 流量 ≤ 容量

如果我们模型中对流量下界有限制怎么办呢？

即我们希望 下界 ≤ 流量 ≤ 容量

如何求这样网络的最大流与最小流呢？

接下来，将介绍解决这一类问题的方法，但是方法正确性的证明，我们放到最后

有上下界网络流

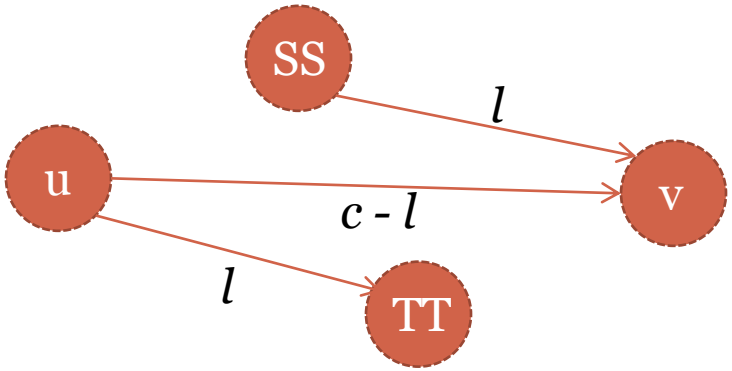
- 介绍
  - 什么是上下界
  - ✓ 模型
  - 最大流与最小流
- 例题
- 题目
- 拓展

模型：

对于有容量下界的网络流，我们通常的做法是建立附加源汇，重建有流量下界的边

考虑有一条边(u, v)，下界为l，容量为c

我们建立附加源和汇 SS和TT



其中边上的数值表示这条边在新图中的容量

通过对图的重建，我们得到的新图是不存在流量下界的

对于这张新图求SS到TT的最大流，如果所有从SS出发的边满流，说明原图存在无源无汇的可行流

即存在一些环形的流使得网络中所有边的约束得到满足

最大流与最小流：

当对于一张图，我们能够求出是否存在可行的环流的时候，我们如何去求一个有源有汇有上下界网络的最大流和最小流呢？

对于一个这样有源有汇有上下界的网络，我们要求其最大流或者最小流需要加如一条边

当我们要求其最大流时，我们为新加入的边设置一个下界l

当我们要求其最小流时，我们为新加入的边设置一个容量c

这样我们就可以二分我们设置的这个值，再使用前面说的方法判断是否存在可行流，这样就能得到最大流和最小流了

事实上存在更优秀的方法，这里不予介绍

Budget：

有一个$n \times m$的方阵，里面的数字未知，但是我们知道如下约束条件：

- 每一行的数字的和为r_i
- 每一列的数字的和为c_i
- 某些格子有特殊的大小约束$A_{i,j}$，用大于号，小于号和等于号表示

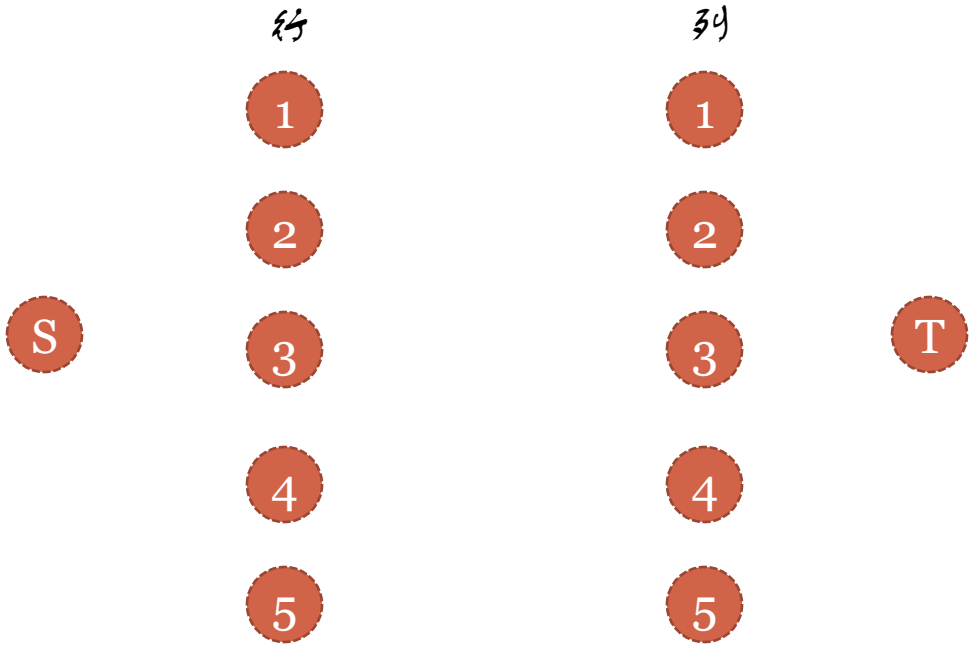
询问是否存在用正数填充这个方阵的方案满足所有的约束，如果有就输出方案否则输出*Impossible*

有上下界网络流

- 介绍
- 例题
  - 题面
  - ✓ 做法
- 题目
- 拓展

做法：

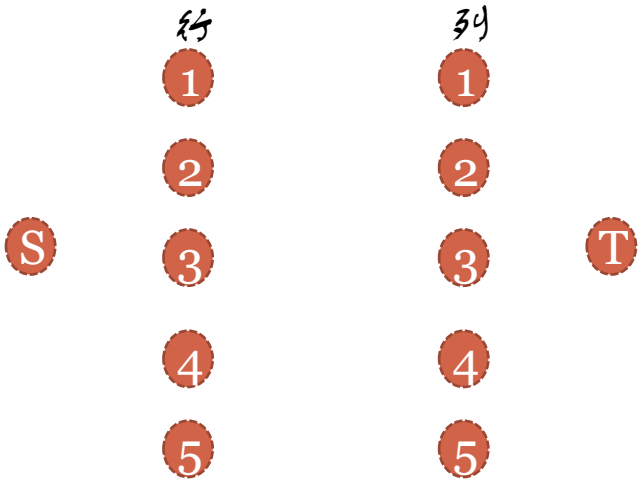
这道题有一个很大的特点，就是棋盘是二维的也就说，我们可以像棋盘二分图一样的建边方法：



有上下界网络流

- 介绍
- 例题
  - 题面
  - ✓ 做法
- 题目
- 拓展

做法：



从S向所有行建边，容量与下界均为r_i

从所有列向T建边，容量与下界均为c_i

从所有行向所有列建边，其中第i行向第j列建的边满足方正中第i行j列的约束

这样我们能建立出一个带有容量下界的网络流模型，求解这一模型就能得到原问题的解

有上下界网络流

- 介绍
- 例题
- ✓ 题目
- 拓展

题目：

这里只列举了一些基础题目

sgu194 Reactor Cooling

zoj3229 Shoot the Bullet

sgu176 Flow construction

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 根据流量平衡建模
  - NOI08志愿者招募
  - 上下界网络流证明

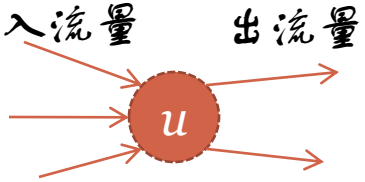
流量平衡建模：

在最开始的时候网络流的部分，介绍了网络流的一些性质。

流量平衡就是其中之一。

在这里，专门提出以下利用流量平衡来建模

我们再回顾一下流量平衡：



所谓的流量平衡，指的就是：

入流量 = 出流量

当然这一点是很显然的

但是如果我们在等式上进行一些操作会怎么样呢？我们如何才能满足以下的条件？

入流量 + Z = 出流量

其中Z是一个常数

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 根据流量平衡建模
  - NOI08志愿者招募
  - 上下界网络流证明

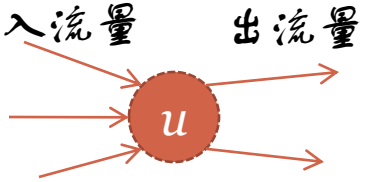
流量平衡建模：

在最开始的时候网络流的部分，介绍了网络流的一些性质。

流量平衡就是其中之一。

在这里，专门提出以下利用流量平衡来建模

我们再回顾一下流量平衡：



所谓的流量平衡，指的就是：

入流量 = 出流量

当然这一点是很显然的

但是如果我们在等式上进行一些操作会怎么样呢？我们如何才能满足以下的条件？

入流量 + Z = 出流量

其中Z是一个常数

有上下界网络流

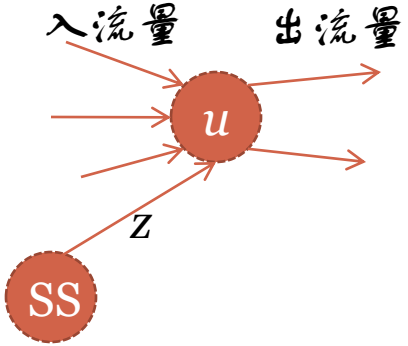
- 介绍
- 例题
- 题目
- 拓展
  - ✓ 根据流量平衡建模
  - NOI08志愿者招募
  - 上下界网络流证明

流量平衡建模：

既然

$$入流量 + z = 出流量$$

那么



我们这样搞一搞不就解决了？

同样的，如果“入流量 = 出流量 + z”

我们也可以类似的处理一下，从u向TT连一条容量为z的边即可

这样我们求SS到TT的最大流，如果所有加入的附加边都满流了，那么说明找到了一个可行解

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - ✓ 根据流量平衡建模
  - NOI08志愿者招募
  - 上下界网络流证明

流量平衡建模：

如果最终存在有附加边没有满流，说明该网络不存在可行解，证明比较显然。

我们发现，如果这样做，是不是意味着我们可以用网络流来解方程？

答案是否定的，存在两个比较明显的问题：

1. 不是所有方程组都可以建模成网络流
2. 网络流求出的解中，所有变量都非负

正因为这样的问题，所以网络流并不能解所有方程组。但是对于有些特殊的问题，要求所有解非负。使用网络流解这类问题就非常的方便

当然我们也可以将问题转化为满足条件的等式这样就能够通过流量平衡建模了

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - 根据流量平衡建模
  - ✓ NOI08志愿者招募
  - 上下界网络流证明

志愿者招募：

有n天每天需要a_i个志愿者

有m种志愿者每种可已从第x_i天干到第y_i天花费为c_i

问最小花费

这里讲通过举这道题的例子，展示如何利用流量平衡建模

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - 根据流量平衡建模
  - ✓ NOI08志愿者招募
  - 上下界网络流证明

做法：

这里引用byvoid博客上的做法

例如一共需要4天，四天需要的人数依次是4,2,5,3。有5类志愿者，如下表所示：

种类	1	2	3	4	5
时间	1-2	1-1	2-3	3-3	3-4
费用	3	4	3	5	6

设雇佣第i类志愿者的人数为$X[i]$，每个志愿者的费用为$V[i]$，第j天雇佣的人数为$P[j]$，则每天的雇佣人数应满足一个不等式，如上表所述，可以列出

- $P[1] = X[1] + X[2] \geq 4$
- $P[2] = X[1] + X[3] \geq 2$
- $P[3] = X[3] + X[4] + X[5] \geq 5$
- $P[4] = X[5] \geq 3$

对于第i个不等式，添加辅助变量$Y[i]$ ($Y[i] \geq 0$)，可以使其变为等式（这是线性规划的常用做法）：

做法：

种类	1	2	3	4	5
时间	1-2	1-1	2-3	3-3	3-4
费用	3	4	3	5	6

- $P[1] = X[1] + X[2] \geq 4$
- $P[2] = X[1] + X[3] \geq 2$
- $P[3] = X[3] + X[4] + X[5] \geq 5$
- $P[4] = X[5] \geq 3$

对于第i个不等式，添加辅助变量$Y[i]$ ($Y[i] \geq 0$)，可以使其变为等式（这是线性规划的常用做法）：

- $P[1] = X[1] + X[2] - Y[1] = 4$
- $P[2] = X[1] + X[3] - Y[2] = 2$
- $P[3] = X[3] + X[4] + X[5] - Y[3] = 5$
- $P[4] = X[5] - Y[4] = 3$

在上述四个等式上下添加P[0]=0,P[5]=0，每次用下边的式子减去上边的式子，得出：

做法：

种类	1	2	3	4	5
时间	1-2	1-1	2-3	3-3	3-4
费用	3	4	3	5	6

- $P[1] = X[1] + X[2] - Y[1] = 4$
- $P[2] = X[1] + X[3] - Y[2] = 2$
- $P[3] = X[3] + X[4] + X[5] - Y[3] = 5$
- $P[4] = X[5] - Y[4] = 3$

在上述四个等式上下添加P[0]=0,P[5]=0，每次用下边的式子减去上边的式子，得出：

① $P[1] - P[0] = X[1] + X[2] - Y[1] = 4$

② $P[2] - P[1] = X[3] - X[2] - Y[2] + Y[1] = -2$

③ $P[3] - P[2] = X[4] + X[5] - X[1] - Y[3] + Y[2] = 3$

④ $P[4] - P[3] = - X[3] - X[4] + Y[3] - Y[4] = -2$

⑤ $P[5] - P[4] = - X[5] + Y[4] = -3$

有上下界网络流

- 介绍
- 例题
- 题目
- 拓展
  - 根据流量平衡建模
  - ✓ NOI08志愿者招募
  - 上下界网络流证明

做法：

种类	1	2	3	4	5
时间	1-2	1-1	2-3	3-3	3-4
费用	3	4	3	5	6

① $P[1] - P[0] = X[1] + X[2] - Y[1] = 4$

② $P[2] - P[1] = X[3] - X[2] - Y[2] + Y[1] = -2$

③ $P[3] - P[2] = X[4] + X[5] - X[1] - Y[3] + Y[2] = 3$

④ $P[4] - P[3] = - X[3] - X[4] + Y[3] - Y[4] = -2$

⑤ $P[5] - P[4] = - X[5] + Y[4] = -3$

将这些等式进行一些变换可以得到

① $X[1] + X[2] = Y[1] + 4$

② $X[3] + Y[1] + 2 = X[2] + Y[2]$

③ $X[4] + X[5] + Y[2] = X[1] + Y[3] + 3$

④ $Y[3] + 2 = X[3] + X[4] + Y[4]$

⑤ $Y[4] + 3 = X[5]$

我们发现，每个变量只出现在了两个等式中，因此我们可以将每个变量看成一条边。满足类似条件等式可以很方便的用刚刚所说的方法建模成网络流。

做法：

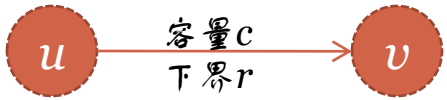
通过刚刚所说的那些推导，我们就能对任意数据建立出模型，然后使用最小费用最大流的算法求解这一问题了。

实际上，我们发现，一个变量最多只出现了两次是一个很重要的条件。

当我们的问题满足这样的条件时，就可以考虑从流量平衡的方向入手了。

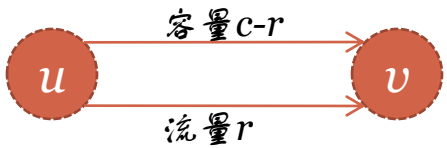
证明：

我们考虑简单的情况



我们用f_0表示这条边的流量

对于这样一条边，我们可以拆成两部分



其中一条边流量确定为常数r，另一条边我们用f_1表示它的流量

可以列出等式：

对于u $f_0 = f_1 + r$ 其中f_0为入流量的总和

对于v $f_1 + r = f_0$ 其中f_0为出流量的总和

有上下界网络流

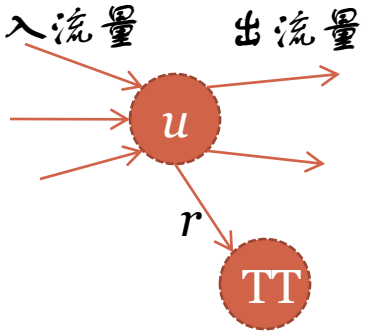
- 介绍
- 例题
- 题目
- 拓展
  - 根据流量平衡建模
  - NOI08志愿者招募
  - ✓ 上下界网络流证明

证明：

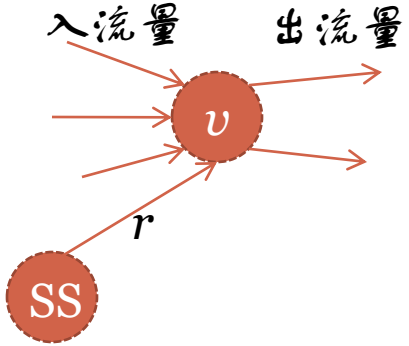
对于u　　$f_o = f_1 + r$ 其中f_o为入流量的总和

对于v　　$f_1 + r = f_o$ 其中f_o为出流量的总和

所以根据前面所说的有：

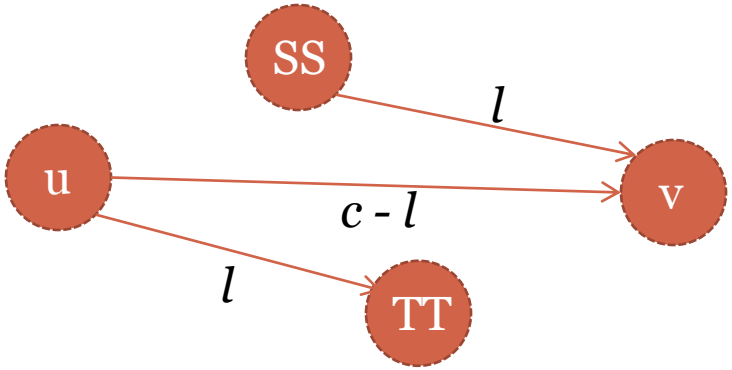


同理，对于V



证明：

到这里，就证明得差不多了。



根据前面的推导，我们得到了这个最终的模型

在实际建模中，直接使用有上下界的网络流，能给我们带来很大的便利。

二分图与费用流

绵阳南山中学 曾国洋

二分图与费用流

- ✓ 介绍
- □ 证明
- □ 应用

二分图与费用流：

通常二分图最大权匹配我们会使用效率较高的KM算法。

然而在有些时候，使用费用流求解二分图最大权匹配时，会存在一些特殊的性质，这一部分会粗略的讲解其中一个性质

使用费用流计算二分图最大权匹配，考虑我们只增广一条最短路径（所以二分图上边权取负）。

我们会发现每次增广，二分图中匹配边边权总和会增加ΔL

其中ΔL=费用流增广时求出的最短路长度

用ΔL_i表示第i次增广时匹配边边权和的变化量

我们会发现其满足以下性质$\Delta L_1 \leq \Delta L_2 \leq \cdots \leq \Delta L_n$

即每次增广时最短路长度是

二分图与费用流

- 介绍
- ✓ 证明
- 应用

证明：

我们考虑相邻两次增广ΔL_x和ΔL_{x+1}

因为在第一次增广后需要建立反向边，所以第二次增广可能会经过第一次增广建立的反向边。

我们分情况讨论：

1. 如果此次增广没有经过上次增广所建立的反向边，很显然$\Delta L_x \leq \Delta L_{x+1}$
2. 如果此次增广经过了上次增广时所建立的反相边，则是以下情况：

二分图与费用流

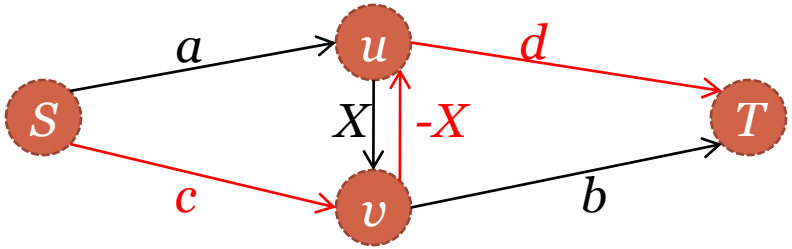
- 介绍
- ✓ 证明
- 应用

证明：

我们假设上次增广路长度为 $a+X+b$

此次增广路长度为 $c-X+d$

其中X是此次增广时经过上次增广时某条反向边的长度



图中，黑边为上次增广时的路径，红边为本次增广时的路径。

我们可以列出下列不等式

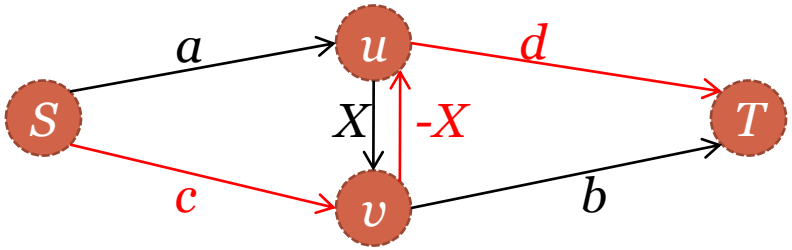
- $c+b>a+X+b$
- $a+d>a+X+b$

因为增广时选择的是最短路，上次增广之所以选择了$a+X+b$就是因为它是其中最短的

二分图与费用流

- 介绍
- ✓ 证明
- 应用

证明：



- $c+b>a+X+b$
- $a+d>a+X+b$

然后我们对不等式进行一些变换：

- $c>a+X$
- $d>X+b$

通过不等式两边减去一个变量

我们再将两个不等式相加：

$$c+d>a+X+b+X$$

$$c-X+d>a+X+b$$

这样就得以证明了

但是此结论能否推广到一般图？请自行思考

二分图与费用流

- 介绍
- 证明
- ✓ 应用

应用：

这里提供一道题目，需要用到此思想

Codechef Annual Parade