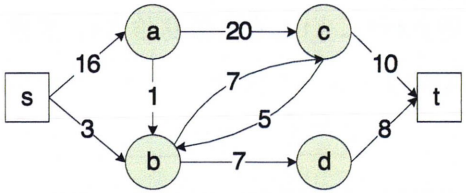
网络流

人大附中
梁霄

定义

流网络 $G=(V,E)$ 是一个有向图，记 $n=|V|, m=|E|$，图中的每条有向弧 $(u,v)\in E$ 有一个非负的容量限制 $c(u,v)\geq 0$。

流网络的源点记为 s，汇点记为 t。对于每个结点 $v\in V$，流网络都存在一条路径 $s\rightsquigarrow v\rightsquigarrow t$。因此流网络是连通的，且不允许有自循环。



流网络

在流网络的所有顶点中，有两个特殊的点：源点s和汇点t。

为了方便起见，假定每个节点都在从源结点到汇点的某条路径上。也就是说对于每个结点v，流网络都包含一条路径s ⇝ v ⇝ t。

流网络图是连通的。在流网络中不允许自循环。

性质

设G(V,E)为一个流网络，其容量函数为c。设s为源点，t为汇点。G中的流是一个实值函数f：$V \times V \to R$，满足下面的两条性质：

- 容量限制：对于所有的结点u，$v \in V$，有$0 \le f(u,v) \le c(u,v)$；
- 流量平衡：对于除了源点s和汇点t之外的任意结点u，要求

$\sum_{v \in V} f(v,u) = \sum_{v \in V} f(u,v)$

最大流问题

一个流f的值定义如下：

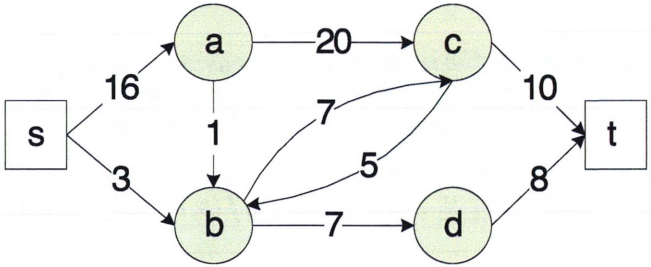
$$|f| = \sum_{u \in V} f(s,u) - \sum_{v \in V} f(v,s)$$

即流f的值是从源点留出的总流量减去流入源点的总流量。通常一个流网络不会有任何进入源点的边，所以上式的后面的求和项是0。

最大流问题就是最大化$|f| = \sum_u f(s,u) = \sum_v f(v,t)$，$|f|$表示从源点s流出的净流量，也等于流入汇点t的净流量。

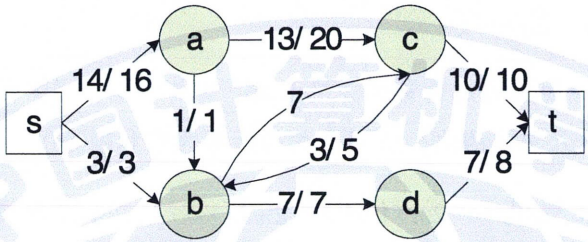
例子

这个流网络的最大流是多少？



例子

最大流：17



流的性质

流f满足以下三个性质：

- 容量限制：f(u,v)<=c(u,v)；
- 流量对称性：f(u,v)=-f(v,u)；
- 流量平衡：对于除了源点s和汇点t之外的任意节点u，满足流量平恒 $\sum_v f(u,v)=0$。即对于顶点u，流进的流量和流出的流量必须完全相同。流进顶点的流量会表示为负值，流出顶点的流量表示为正值。

注意这里的流量f(u,v)是可正可负的，负流量的引入是FF系算法理论依据的证明所必须的。最大流问题就是最大化$|f|=\sum_u f(s,u)=\sum_v f(v,t)$。

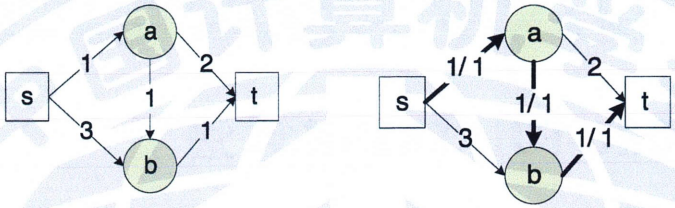
FORD-FULKERSON算法

求解最大流

贪心+暴力：建图记录每条边的最大流量，然后bfs或者dfs找出能够把流从源点传输到汇点的路径，并把这条路径上的流量流掉（边权减小）。反复执行此过程直到找不到增广路径为止。

问题

这个贪心并不正确。例如下图中第一次找增广路选择了s-a-b-t，则按照此路径传输流时，会得到下图结果，此时就无法再从s向t传输流，算法终止，最大流是1，答案错误。



FORD-FULKERSON算法

Ford-Fulkerson方法中提出了残量网络和增广路径的概念，是解决最大流问题的重要思想和理论依据，有几种不同的实现方法。下面先给出几个概念。

核心概念：残量网络、增广路径。

理论依据：增广路定理。

残量网络

残量网络：给定流网络 G 和流 f，定义 $c_f(u,v)$ 为其残量网络的容量。Ford-Fulkerson 方法对流量进行操作的目标是增加总流量，为此，算法运行过程中可能会缩减某些弧上的流量。为了表示对一个正流量的缩减，将有向弧 (u,v) 的反向弧 (v,u) 加入残量网络中，并将其容量设置为 $f(u,v)$。残量网络中包含原图中不存在的弧：

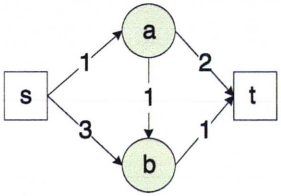
$$c_f(u,v)=\begin{cases} c(u,v)-f(u,v), & (u,v)\in E \\ f(u,v), & (v,u)\in E \\ 0, & \text{otherwise} \end{cases}$$

设 $E'=\{(u,v):(u,v)\in E\text{或}(v,u)\in E\}$，有向图 $G'=(V,E')$。残量网络 G_f 中的弧由网络 G' 中所有满足 $c_f(u,v)>0$ 的弧构成。残量网络描述了流网络 G 中各弧的剩余容量以及可以通过"回流"删除的流量大小。注意若 $c_f(u,v)=0$，表示这条弧上的流量已经饱和了，该弧就不属于残量网络 G_f 了。

增广路径、割

增广路径：给定流网络 $G=(V,E)$ 和流 f，增广路径 p 是残量网络 G_f 中一条从源点 s 到汇点 t 的简单路径，且路径上的弧均有流量剩余。

割：流网络 $G=(V,E)$ 的割 (S,T) 将点集 V 划分为 S 和 $T=V-S$ 两个集合，使得 $s\in S$，$t\in T$。定义割 (S,T) 的容量是 $c(S,T)=\sum_{u\in S}\sum_{v\in T}c(u,v)$。一个流网络的最小割就是该网络中容量最小的割。



最大流最小割定理

最大流最小割定理：设f为流网络 $G=(V,E)$ 中的一个流，该流网络的源节点为 s，汇点为 t，则下面的3个条件是等价的：

1）f 是 G 的一个最大流；

2）残量网络 G_f 中不包括任何增广路径；

3）$|f|=c(S,T)$，其中 (S,T) 是流网络 G 的割。

增广路定理：一个流是最大流当且仅当其残量网络不包含任何增广路径。

FORD-FULKERSON算法

根据最大流最小割定理，求解最大流问题可以采用如下方法：

（1）初始化流网络的流 f，令 $f(u,v)=0,(u,v)\in E$。并构建残量网络 G_f：

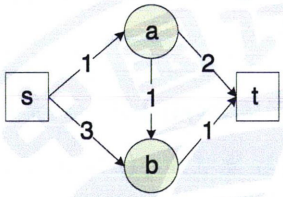
$$c_f(u,v)=\begin{cases}c(u,v),(u,v)\in E\\0, \qquad (v,u)\in E\end{cases}$$

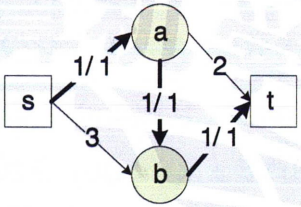
（2）在残量网络 G_f 中寻找增广路径 p，并沿增广路径 p 更新残量网络 G_f。具体来说，设 $c_f(p)=min\{c_f(u,v),(u,v)\in p\}$ 表示增广路径 p 的流量，对于路径 p 上的每一条弧 (u,v) 令 $c_f(u,v)=c_f(u,v)-c_f(p),c_f(v,u)=c_f(v,u)+c_f(p)$。

（3）重复步骤2，直到残量网络中不存在增广路径为止。

解决最大流问题

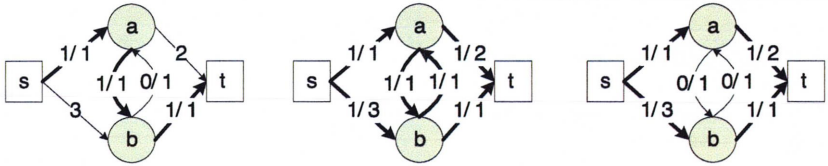
回顾前面的例子。在流量网络上贪心时，第一次找增广路选择了s-a-b-t，则按照此路径传输流时，会得到下图结果，此时就无法再从s向t传输流，算法终止，最大流是1，答案错误。





通过“回流”解决问题

设增广路径p的流量为$c_f(p)$，对路径上的每一条弧(u,v)，其残量网络的流量$c_f(u,v)$减小$c_f(p)$，其反向边$c_f(v,u)$的流量增加$c_f(p)$。算法通过给反向弧增加流量来回撤正向弧的流量。



实现

在算法实现过程中，我们只要维护残量网络即可。

建图时要为每条弧增加一个反向弧，即若u到v有容量c，则不仅要增加一条u到v的弧，容量为c，同时还要增加一条反向弧v到u，容量为0，而且这一对弧要互相关联在一起，保证能找到对方。

步骤2每次增广的时候，若弧(u,v)产生一个正向流量$c_f(p)$，在残量网络上，记弧(u,v)的容量减小$c_f(p)$，其反向弧的容量增加$c_f(p)$。

代码细节

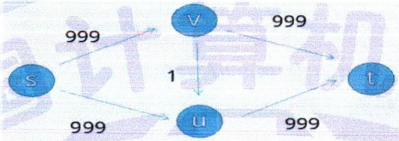
```
struct edge
{
    int v, next;
    ll flow;
}e[2 * maxm];
int head[maxn], tot = -1;

inline void addedge(int u, int v, ll c)
{
    e[++tot] = (edge){v, head[u], c};
    head[u] = tot;
    e[++tot] = (edge){u, head[v], 0};
    head[v] = tot;
}
```

编号x的边的反向边是x^1

时间复杂度

FF算法的效率取决于步骤2（找增广路）及其执行次数。使用bfs或dfs在残量网络中寻找一条增广路的复杂度是$O(|V|+|E'|)\sim O(m)$的，用f^*表示网络的最大流，每次最少增广1流量，则FF方法的时间复杂度为$O(m|f^*|)$。

算法导论中给出了一个最坏情况的例子。如下图所示：



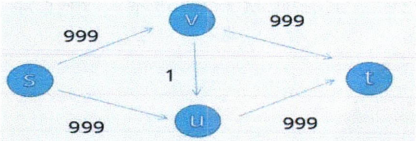
如果第一次选择了s->v->u->t的路径增广，共需要999*2次循环找增广路，非常低效。非常容易出特殊数据来把这个算法的复杂度卡满。

EDMONDS-KARP算法

Edmonds-Karp算法是对上述基础Ford-Fulkerson方法的改进。Edmonds-Karp算法与基础Ford-Fulkerson方法的唯一区别是EK算法每次通过广度优先搜索寻找一条从源点 s到汇点 t 的最短的增广路进行增广。

通过广度优先搜索找一条最短路径的复杂度是 $O(m)$，Edmonds-Karp算法的增广次数的上界是 $O(|V||E|) = O(mn)$，算法的总复杂度是 $O(nm^2)$。

沿最短路增广可以很好的解决下面这种情况：



DINIC

Dinic算法也是对基础Ford-Fulkerson方法的改进，主要思想是通过建立分层图来加速增广过程。

Dinic算法步骤如下：

（1）初始化流网络的流 f，令 $f(u,v)=0,(u,v)\in E$。并构建残量网络 G_f。

（2）从源点 s 开始沿着 $c_f(u,v)>0$ 的弧进行广度优先搜索，每一个点到源点的距离记为 dis_i。如果遍历不到汇点，即找不到增广路径，算法结束。

（3）在残量网络上按照如下方式增广：设当前点为 i，接下来只选择距离为 dis_i+1 的点进行扩展。扩展到汇点 t 就构成了一条增广路径 p，沿增广路径 p 更新残量网络 G_f。然后在分层图上继续寻找增广路并更新残量网络，直到在当前分层图中找不到增广路径，回到步骤2。

DINIC

Dinic算法按照 dis[i] 将所有点分层，构建了分层图：同一层的点到源点的距离相等。然后在分层图上做深度优先搜索来找增广路径，增广路径上的每条边一定从分层图的某一层去往下一层。

Dinic算法在增广的时候，只选择距离恰好是自己距离加1的点扩展，保证了每次以最短路增广。

其次Dinic算法可以进行多路增广，即在找到了一条增广路之后，并不是退回源点重新寻找，而是尝试在此次获得的增广路上距离汇点最近的满流的弧的起点继续增广。

时间复杂度

在Dinic的执行过程中，每次重新分层，汇点所在的层次是严格递增的，所以最多重新分层n次。

在同一个层次图中，算法是以最短路进行增广的，最多有m条增广路，搜索每条增广路前进和回退最多n次，因此完成每个层次图的复杂度上界为$O(nm)$。

综上所述，Dinic算法时间复杂度的理论上界是$O(n^2m)$。

优化

1. 当前弧优化；
   - 当前弧优化其实就是用一个数组记录每一次对图进行分层之后，第i个点已经增广到哪一条边，这样每次增广时都不走原来走过的边。
2. 断层优化；
   - 如果从一个点出发再也没有可行弧，可以把这个点标记为废点，令**level[u]=-1**。记录每一层的废点的个数，如果出现了断层，源点再也不可能到达汇点，可以直接退出循环。
3. 多路增广优化
   - 多路增广优化就是每次找到增广路之后都不回退到源点重新开始找增广路，而是回退到最近的限流边，然后继续增广，很容易通过**dfs**实现。

最大流题目选讲

例题

Poj 1698 爱丽丝拍电影

Poj 3281 进餐

Poj 1149 猪圈

CQOI2009 跳舞

SCOI2012 奇怪的游戏

SDOI 2015 星际战争

CQOI2015 网络吞吐量

POJ 1698 爱丽丝拍电影

爱丽丝要拍电影，有n部电影，规定爱丽丝每部电影在每个礼拜只有固定的几天可以拍电影，只可以拍前面w个礼拜，并且这部电影要拍d天，

问爱丽丝能不能拍完所有的电影

第一行代表有多少组数据

对于每组数据第一行代表有n部电影

接下来2到n+1行，每行代表一个电影，每行9个数，前面7个数，1代表拍，0代表不拍，第8个数代表要拍几天，第9个数代表有几个礼拜时间拍

POJ 1698 爱丽丝拍电影题解

建图：超级源点s，n部电影，每一天（共7*50=350天），汇点t

s向电影i连边，流量为第i部电影需要工作的天数

第i部电影向可以出演的每一天连边，流量为1

然后所有天向汇点t连边，流量为1

然后套模板就行了

POJ 3281进餐

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。

一头牛只吃特定的几种食物和特定的几种饮料，John手里每种食物和饮料都只有一个，问**最多**能够满足几头牛的需求(水和食物都必须合他们的胃口)。

POJ 3281 进餐题解

建图：超级源点s，f种食物，n头牛，n头牛，d种饮料，汇点t

s向食物连边，限制每种食物只有一种

食物向牛连边，牛向饮料连边，要限制饮食偏好

牛向牛连边，限制只能吃一种食物，且必须同时吃饭和喝饮料才能计算流量

然后所有饮料向t连边，限制每种饮料只有一种

此图所有边的流量初始化为1

SDOI 2015星际战争

在战斗的某一阶段，Y军团一共派遣了N个巨型机器人进攻X军团的阵地，其中第i个巨型机器人的装甲值为Ai。当一个巨型机器人的装甲值减少到0或者以下时，这个巨型机器人就被摧毁了。

X军团有M个激光武器，其中第i个激光武器每秒可以削减一个巨型机器人Bi的装甲值。激光武器的攻击是连续的。

这种激光武器非常奇怪，一个激光武器只能攻击一些特定的敌人。Y军团看到自己的巨型机器人被X军团一个一个消灭，他们急需下达更多的指令。

为了这个目标，Y军团需要知道X军团最少需要用多长时间才能将Y军团的所有巨型机器人摧毁。但是他们不会计算这个问题，因此向你求助。

星际战争题解

二分+最大流

源点到武器的流量为时间*武器每秒的输出，武器和它能攻击的机器人之间的流量设为无穷大，机器人到汇点的流量为机器人的装甲值。

这个题目要注意精度问题

SCOI2012奇怪的游戏

Blinker最近喜欢上一个奇怪的游戏。

这个游戏在一个 N*M 的棋盘上玩，每个格子有一个数。每次 Blinker 会选择两个相邻的格子，并使这两个数都加上 1。

现在 Blinker 想知道最少多少次能使棋盘上的数都变成同一个数，如果永远不能变成同一个数则输出-1。

题解

巧妙的最大流

对棋盘进行黑白染色，设最后数值都变为x
设黑格个数为num1 数值和为sum1
设白格个数为num2 数值和为sum2

num1 * x − sum1 = num2 * x − sum2
x = (sum1 − sum2) / (num1 − num2)
num1 ≠ num2时 可以解出 x 再用网络流check即可

num1 = num2时 可以发现 对于一个合法的x， 任意k>=x都是合法的解
所以可以二分x 然后用网络流check

题解

建图：

如果点k为白

建边(s, k, x − v[k])

如果为黑

建边(k, t, x − v[k])

对相邻点u、v (u为白)

建边 (u, v, inf)

CQOI2009跳舞

一次舞会有n个男孩和n个女孩。每首曲子开始时，所有男孩和女孩恰好配成n对跳交谊舞。每个男孩都不会和同一个女孩跳两首（或更多）舞曲。有一些男孩女孩相互喜欢，而其他相互不喜欢（不会“单向喜欢”）。每个男孩最多只愿意和k个不喜欢的女孩跳舞，而每个女孩也最多只愿意和k个不喜欢的男孩跳舞。给出每对男孩女孩是否相互喜欢的信息，舞会最多能有几首舞曲？

CQOI2009跳舞题解

二分+最大流，拆点建图

把每个人拆成喜欢和不喜欢两个点
从S连向男生喜欢，容量为a，再从男生喜欢连向男生不喜欢连边，容量为K
男生喜欢连向女生喜欢，男生不喜欢连向女生不喜欢
女生之间的连边类似于男生

POJ 1149 PIGS

有 M 个猪圈（M ≤ 1000），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（N ≤ 100），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。（猪圈容量无限）

问总共**最多**能卖出多少头猪。

POJ 1149 PIGS题解

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。

POJ 1149 PIGS题解

暴力建图：

三个顾客，就有三轮交易，每一轮分别都有 3 个猪圈和 1 个顾客的节点。

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。

POJ 1149 PIGS题解

刚才的暴力建图有nm个点，2nm条边，太慢了，想办法化简：

每个顾客分别用一个节点来表示。

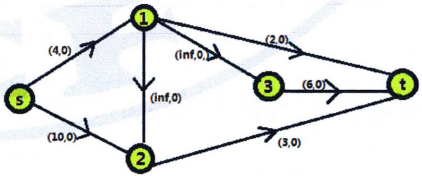
对于每个猪圈的第一个顾客，从源点向他连一条边，容量就是该猪圈里的猪的初始数量。如果从源点到一名顾客有多条边，则可以把它们合并成一条，容量相加。

对于每个猪圈，假设有 x 个顾客打开过它，则对所有整数 $i \in [1, x)$，从该猪圈的第 i 个顾客向第 i + 1 个顾客连一条边，容量为 $+\infty$。

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。

新模型中只有N个点。

POJ 1149 PIGS题解



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。