多情流水伴人行

——网络流相关

周以凡

浙江省慈溪中学

2014 年 5 月 22 日

Q&A

- Q:你是谁？慈溪中学是啥？
- A:我是周以凡，初代退队狗，没什么名气，不过听说某个平行宇宙里有个周以凡是个人赢。慈溪中学是一个弱校，没有听说过也无所谓。
- Q:你这题目是啥意思，装13么？
- A:被发现了，其实就是网络流，这里需要特别感谢SZY大爷的题名。

Q&A

- Q:网络流不是两年之前就讲过了么？有意义？
- A:至少还是有点不一样的啦，况且我们要照顾一下两年前没来的同学。然后因为我已经退役多时，讲课内容非常简单，算是让大家放松心情的。
- Q:这样啊，可以睡觉么？
- A:睡觉伤颈椎啊，可以抬头望天Counting Stars。

定义

- **流网络**$G(V,E)$是一个有限的有向图，每条边$(u,v) \in \mathbf{E}$都有一个非负实数的容量$c(u,v)$。如果$(u,v) \notin \mathbf{E}$，我们假设$c(u,v) = 0$。
- 在图中，我们区别两个顶点**源点**s和**汇点**t(有时也写作s')。
- **网络流**是对于所有结点u和v都满足以下性质的实数函数$f : \mathbf{V} \times \mathbf{V} \to \mathbf{R}$:

  容量限制: $f(u,v) \leq c(u,v)$;

  斜对称: $f(u,v) = -f(v,u)$;

  流守恒: 除了s,t，结点均满足$\Sigma_{v \in \mathbf{V}} f(u,v) = 0$;

定义

- 边的**剩余容量**$c_f(u,v) = c(u,v) - f(u,v)$。它定义了**剩余网络**$G_f(V,E_f)$，表明网络剩余可用流量的多少。
- 根据斜对称性我们知道，增加u至v的流量的同时需要减少v至u的流量，因为我们考虑**净流量**。这就表现为增加一条边的流量时，我们减小它的剩余容量，同时增加其**反向边**的剩余容量。当我们沿反向边增加流量时，就表现为原边的**退流**。

定义

- **增广路**是一条路径$p = (s, u_1, u_2, \cdots, u_k, t)$，并且$k = min\{c_f(u, v) : (u, v) \in p)\} > 0$，表示我能沿着这条路径传送$k$的流量。
- **网络的流**$f = \Sigma_{(s,v)\in \mathbf{E}} f(s, v) = \Sigma_{(v,t)\in \mathbf{E}} f(v, t)$;
- **最大流**是网络中能达到的最大的网络的流，根据**增广路定理**我们知道网络达到最大流当且仅当剩余网络中没有增广路。

定义

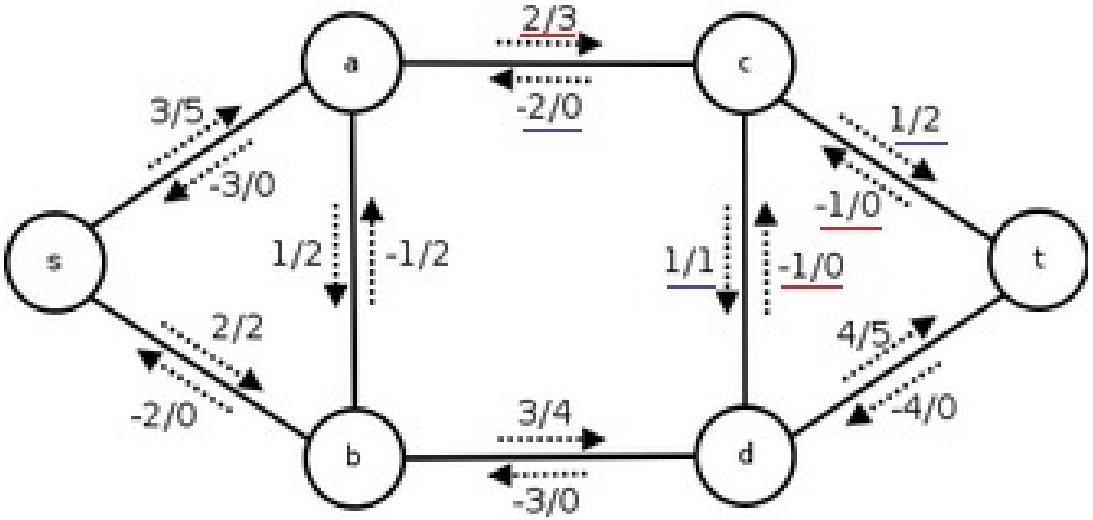
- **边的费用**为每条边具有的费用$w(u,v)$，表示在这条边上每流过1单位的流量，需要$w(u,v)$的费用。
- **最小费用最大流**是保证最大流的情况下，使得费用最小;

  最大费用可行流只需要费用最大，而不需要保证最大流。如果每次寻找费用最大的增广路进行增广，只需要在不存在费用为正的增广路时停止算法即可。

定义

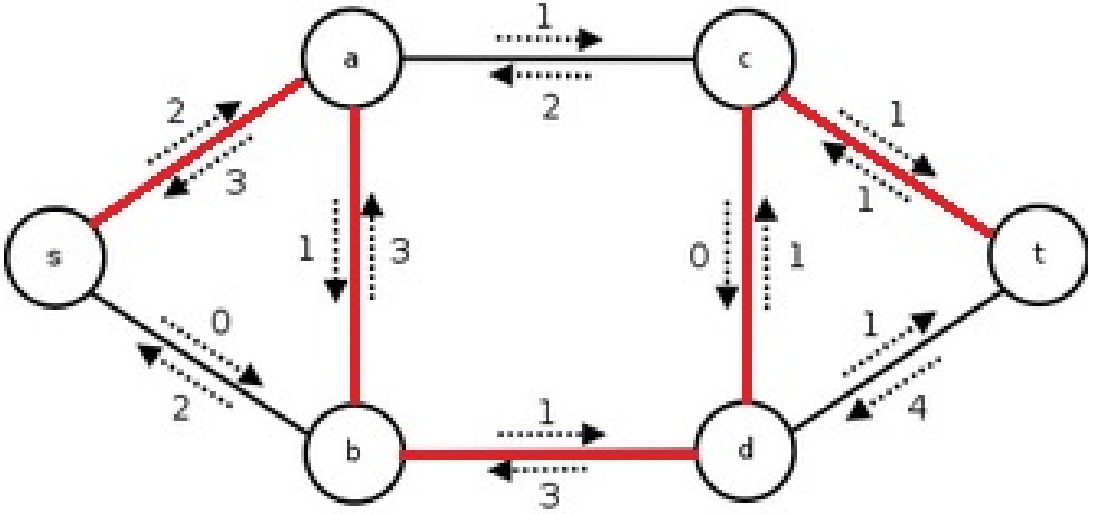
- **割**是边的集合，删去集合中所有边将使得s无法到达t。**割边**是集合中的一条边。
- **最小割**是容量和最小的割。
- 我们根据**最大流最小割定理**知道最大流等于最小割。
- 一条边(u, v)能属于某个最小割当且仅当剩余网络中不存在从u到v最小剩余容量大于0的路径。
- 最小割将点划分为两个集合，我们称包含s的为S集，包含t的为T集。

流网络



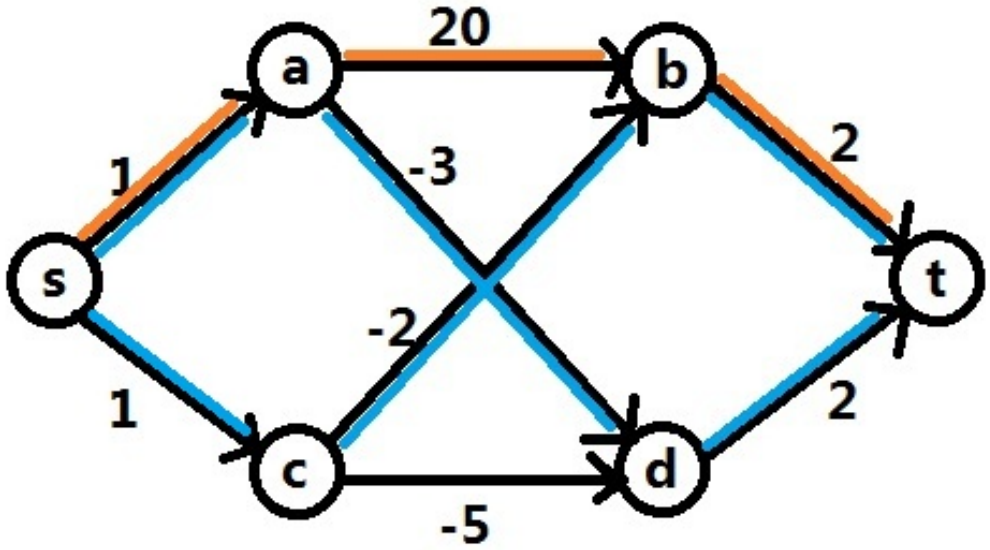
- 图中用f/c表示流量以及容量。通过这张图我们能观察到是如何满足**容量限制**，**斜对称**和**流守恒**。以及当前的总流量为5。

剩余网络



- 将图转化为剩余网络，我们注意到这并不满足最大流，显然路径(s, a, b, d, c, t)是一条**增广路**。尽管原图中边(d, c)容量为0，这时就表现为在边(c, d)上的**退流**。

费用流网络

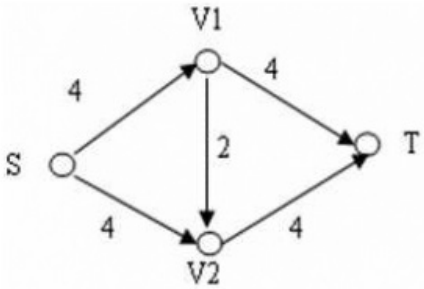


- 图中所有边的容量为1，费用为标注的数字。我们注意到最小费用最大流为蓝色标注的边，流量为2而费用为1，而最大费用可行流为橙色标注的边，流量为1而费用为23。

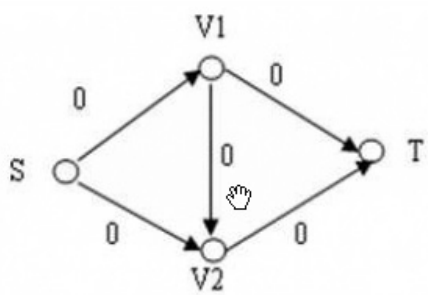
Edmonds-Karp

- 我们熟知的求解最大流的算法是Edmonds-Karp算法，它基于Ford-Fulkerson方法。其思路是每次寻找一条从s到t的增广路进行增广，直到不存在增广路为止。利用增广路定理就可以证明其正确性。
- 在EK算法中，我们使用BFS寻找一条边数最少的增广路，忽略所有剩余容量为0的边。每次找到增广路后，确定出增广路上的最小剩余容量*MinCap*，将网络的流量增加*MinCap*，对于路径上的所有边的剩余容量减去*MinCap*，将路径上所有边的反向边的剩余容量增加*MinCap*。

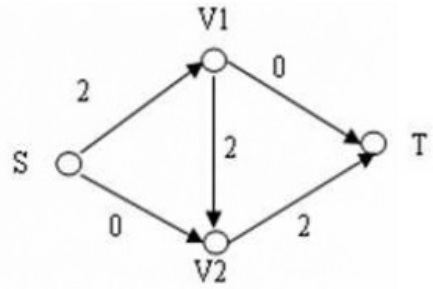
Edmonds-Karp

- 

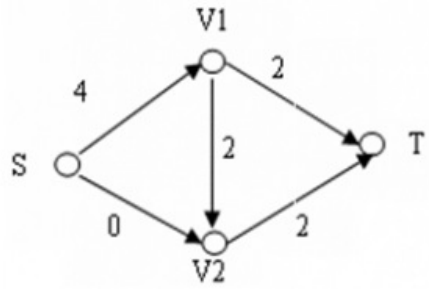
一个简单的网络流图



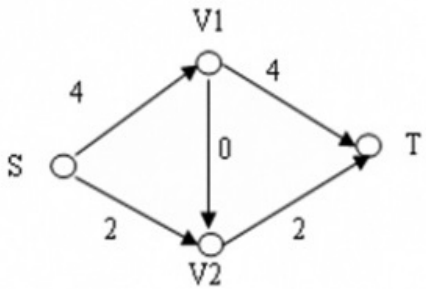
取一个初始可能流（零流）



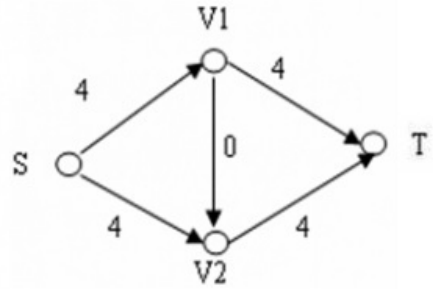
第1次增广(S, V1, V2, T)



第二次增广(S, V1, T)



第三增广(S, V2, V1, T)



第四次增广(S, V2, T)

- 这里需要注意，对于没有下界的网络流，我们都可以取零流作为初始的可能流。之后我们需要关心其复杂度。

Edmonds-Karp

- 增广的次数为$O(VE)$，证明如下:
  我们考虑当前由$s \to t$的最短路组成的边集，集合大小不超过$|E|$。在每次增广中，我们至少使得集合中某条边的剩余容量变为0，并且在最短路长度增加之前，集合中不会新增元素。图中最短路的长度不会超过$|V|$。因此最多进行$O(VE)$次增广。
- 当然增广的次数显然不会超过最大流的值，因此在最大流较小时，EK算法有非常好的表现。
- 所以EK算法的复杂度为
  $O(VE^2)$或者$O(MaxFlow \times E)$。

Dinitz

- 显然相同长度最短路并不只有一条，在一次BFS后我们能得到每个点距离s的距离标号。我们得到剩余网络G_f的分层图G_L，其中
  $E_L = \{(u, v) \in E_f : dist(v) = dist(u) + 1\}$。
- 在分层图上我们得到阻塞流(blocking flow)，即不存在从s到t的最小剩余容量大于0的路径。然后沿着阻塞流增广，并重复这个步骤直到BFS时s与t不连通。这实际上就是我们常用的最大流算法dinic。
- 在搜索增广的过程中如果某个点存在剩余流量，直接将$dist$修改为-1以提高效率。

Dinitz

- 我们知道进行一次阻塞流的增广需要$O(VE)$。而最短路每次都至少增加1，因此至多有$|V|-1$个阶段，因此dinic算法的复杂度为$O(V^2E)$。
- 大部分人都听说过二分图匹配用dinic算法有惊喜。事实上，在边的容量均为1时，我们能在$O(E)$的时间内寻找阻塞流，并且已经有人证明了此时阶段数不超过$O(\sqrt{E})$和$O(V^{2/3})$，因此复杂度为$O(min(V^{2/3},E^{1/2})E)$。
- 在二分图上，阶段数被证明是$O(\sqrt{V})$，并且只需要除了源汇的结点满足只有一条容量为1的出边或只有一条容量为1的入边，复杂度同样是$O(\sqrt{V}E)$。

SAP

- 到目前为止dinic算法是不能再进行优化了，那么我们换一种思路，如果我们将$dist(x)$定义结点x至t的(期望)最短距离，显然我们对于一条剩余容量大于0的边(u, v)有$dist(u) \leq dist(v) + 1$，我们定义一条边为有效边，当且仅当其存在剩余容量且$dist(u) = dist(v) + 1$。
- 能够证明由有效边组成的到t的路径一定是最短路径。
- 此时我们只需要每次沿着有效边寻找增广路，当不存在有效的出边时修正$dist$即可。通过这样的方法我们将每次寻找增广路的复杂度均摊为$O(V)$，因此整个算法复杂度也为$O(V^2E)$。
- 这也是最大流算法SAP的基本想法。

LCT

- 此时SAP算法的瓶颈在于寻找最短路和进行增广，而dinic算法的瓶颈在于寻找阻塞流。我们可以尝试使用动态树来维护有效边(也指dinic算法中分层图中的边)的生成森林，而Link-Cut Tree则是一个很好的选择。

LCT

- 我们回顾(?)一下LCT能实现的功能:

  $link(v, w, x)$: 添加边(v, w)，权值为x，使得某个树根v成为w的儿子;

  $cut(v)$: 切断非根结点v与其父亲的连边，同时返回边权;

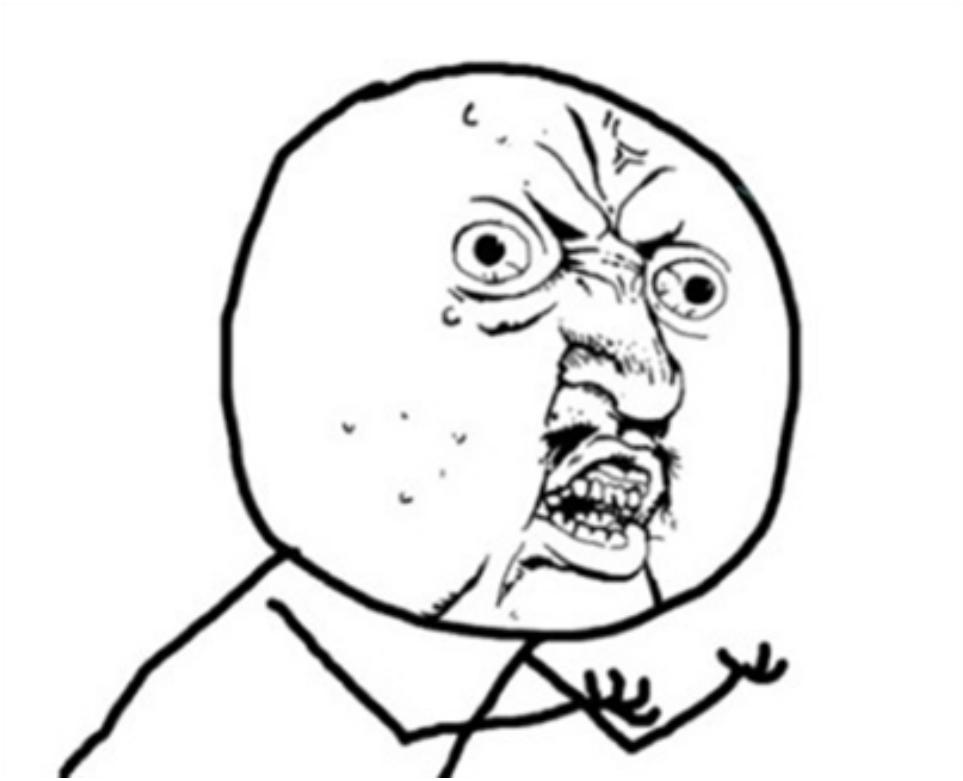
  $parent(v)$: 返回v的父亲，如果不存在则为NULL;

  $root(v)$: 返回v所在树的根;

LCT

- $cost(v)$: 返回v和其父亲之间边的权值;
  $mincost(v)$: 返回v至其根的路径上，距离根最近的具有最小$cost(w)$的w;
  $update(v, x)$: 将v至根的路径上的所有边权增加x;

LCT

- 我们注意到在SAP中一旦推流之后，所有树上的边可以直接修改，然而对于反向边的处理就比较麻烦，我们需要在修改边的时候同时处理反向边，
- 这实在是太麻烦了！本来LCT就够慢的了你还要做啥!
- 

LCT Dinitz

- 我们可以考虑利用LCT加速dinic算法中阻塞流的寻找。首先我们能知道在一个阶段中，分层图的反向边不会产生任何影响。因此我们只需要知道整个阻塞流，就可以之后$O(E)$修改所有边的剩余容量。
- 如果我们在剩余网络上直接进行修改的话，我们只需要在BFS得到分层图的时候，记录原先的剩余容量即可。

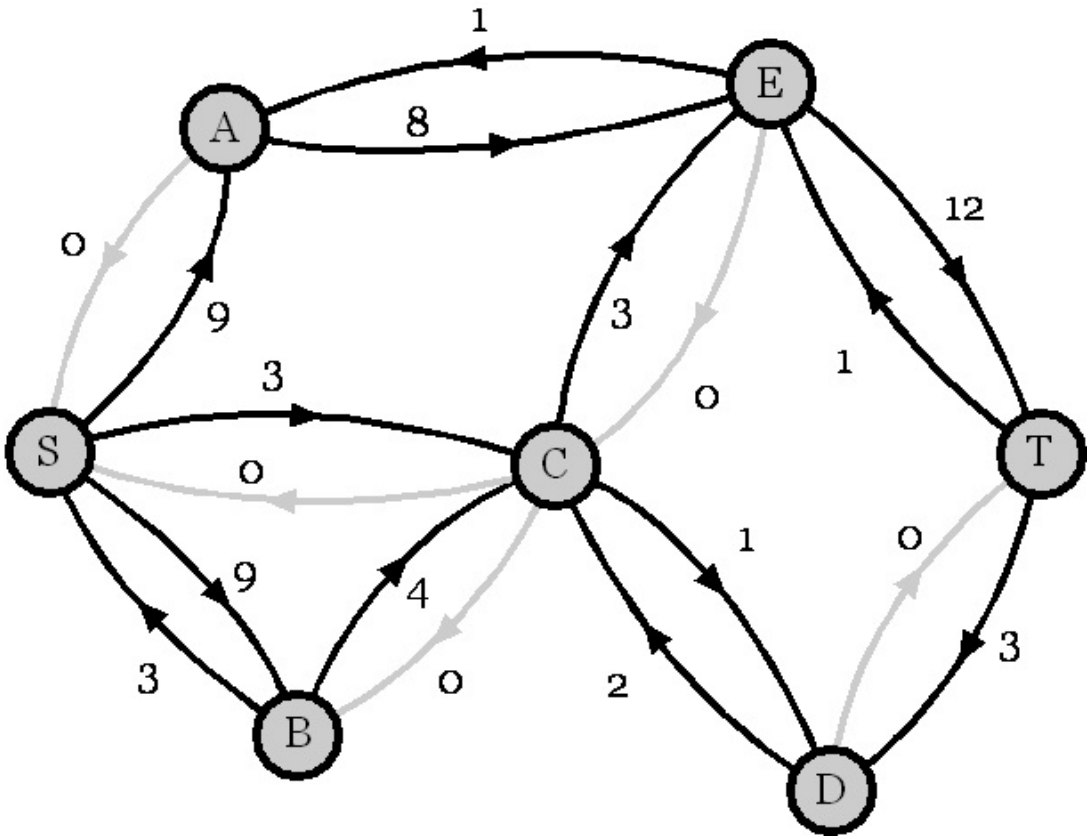
LCT Dinitz

- 我们将dinic中寻找阻塞流的算法修改如下，初始每个点单独组成一棵树：
- Step 1. 令$v = root(s)$，即s所在树中距离t最近的点，如果$v = t$，执行Step 4；否则执行Step 2；
- Step 2($v \neq t$; 扩展路径).
  如果v没有出边，执行Step 3；否则选择一条边(v, w)，并$link(v, w, capacity(v, w))$，继续执行Step 1；
- Step 3(v至t被阻塞). 如果$v = s$，s和t已经被阻隔，结束；否则在分层图中删去v的所有入边，并对树中所有的边(u, v)，进行$cut(u)$，执行Step 1；

LCT Dinitz

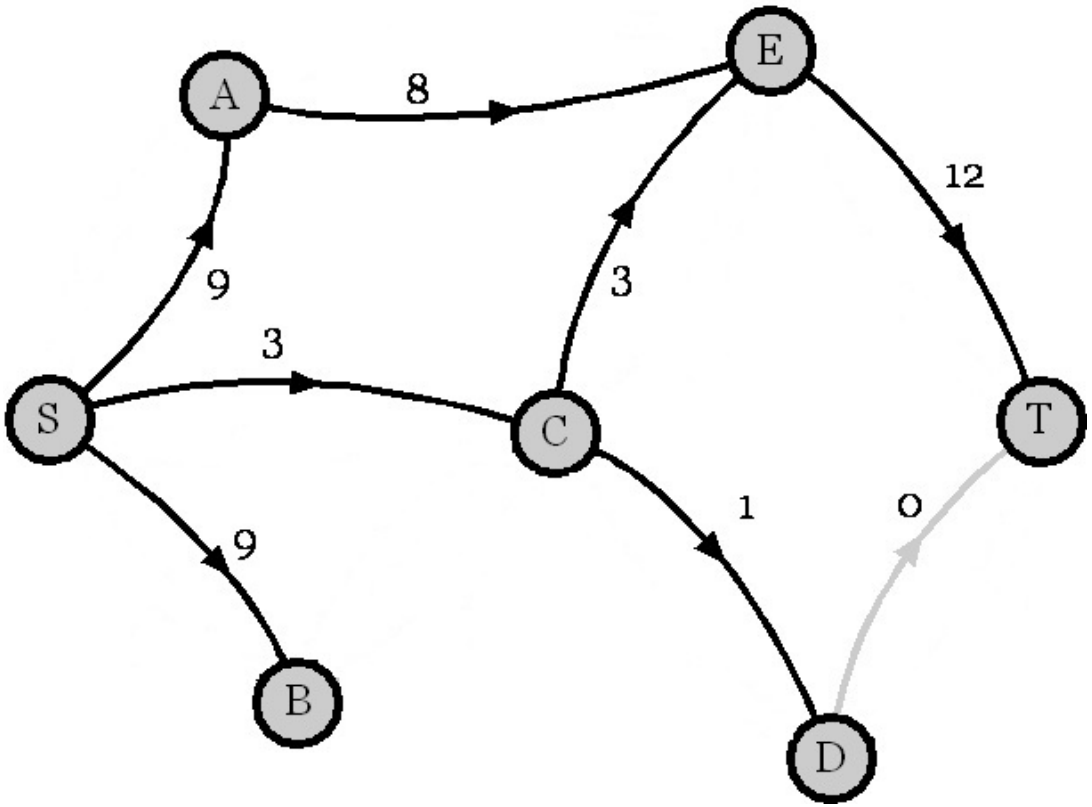
- Step 4($v = t$; 增广路径). 此时我们得到了一条s至t的路径，我们令$v = mincut(s), c = cost(v)$，执行$update(s, -c)$，然后执行Step 5；
- Step 5(删边). 令$v = mincost(s)$，如果$cost(v) = 0$，在分层图中删去边$(v, parent(v))$，并在树中删去该边；重复执行多次直到$cost(v) > 0$，继续执行Step 1；

LCT Dinitz



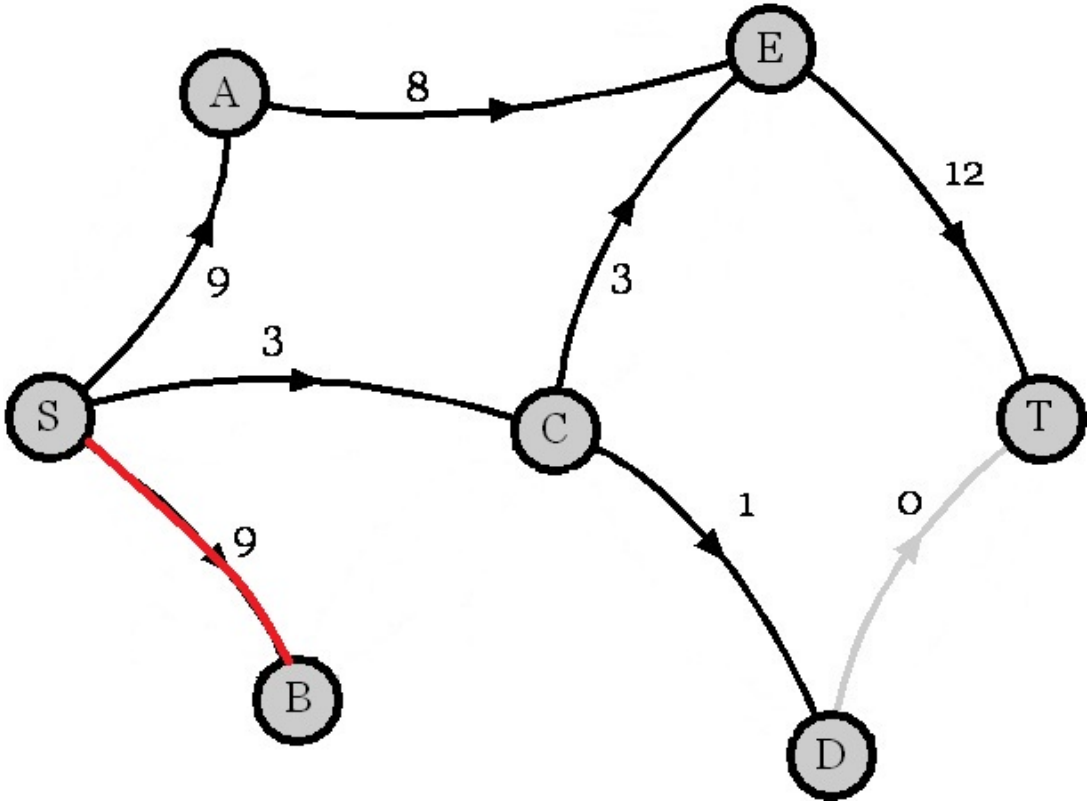
- 我们以这个流网络为例。

LCT Dinitz



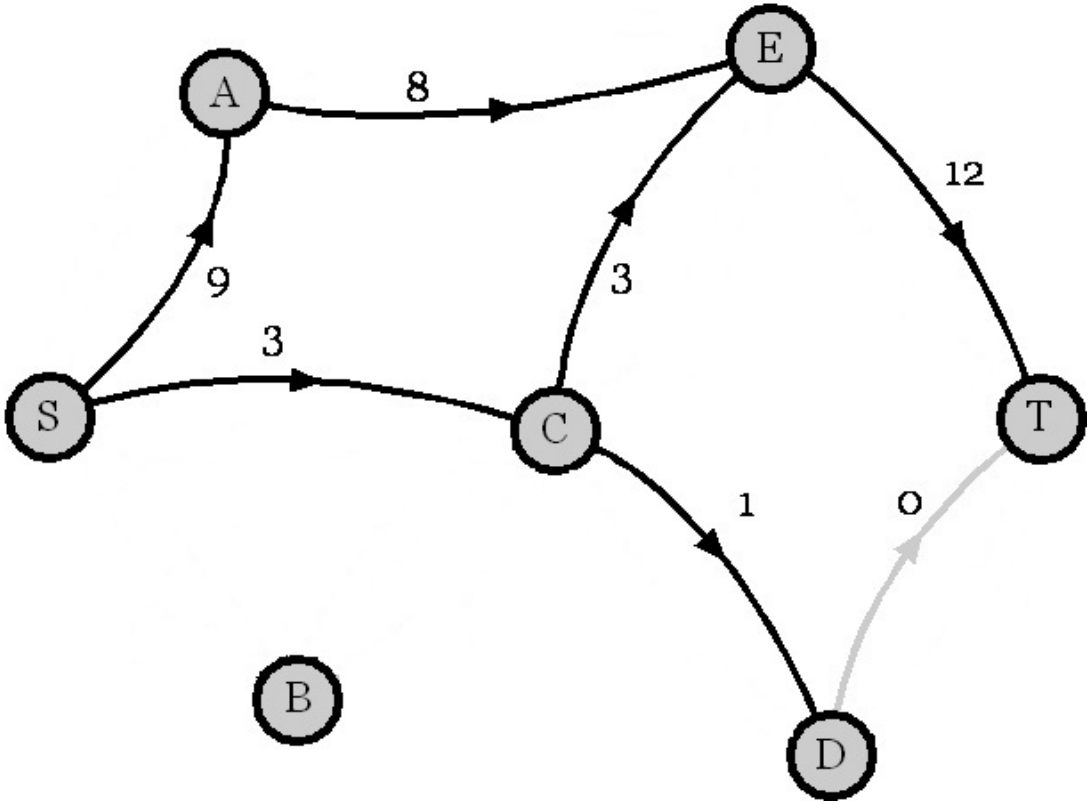
- 首先从S开始BFS，得到分层图。

LCT Dinitz



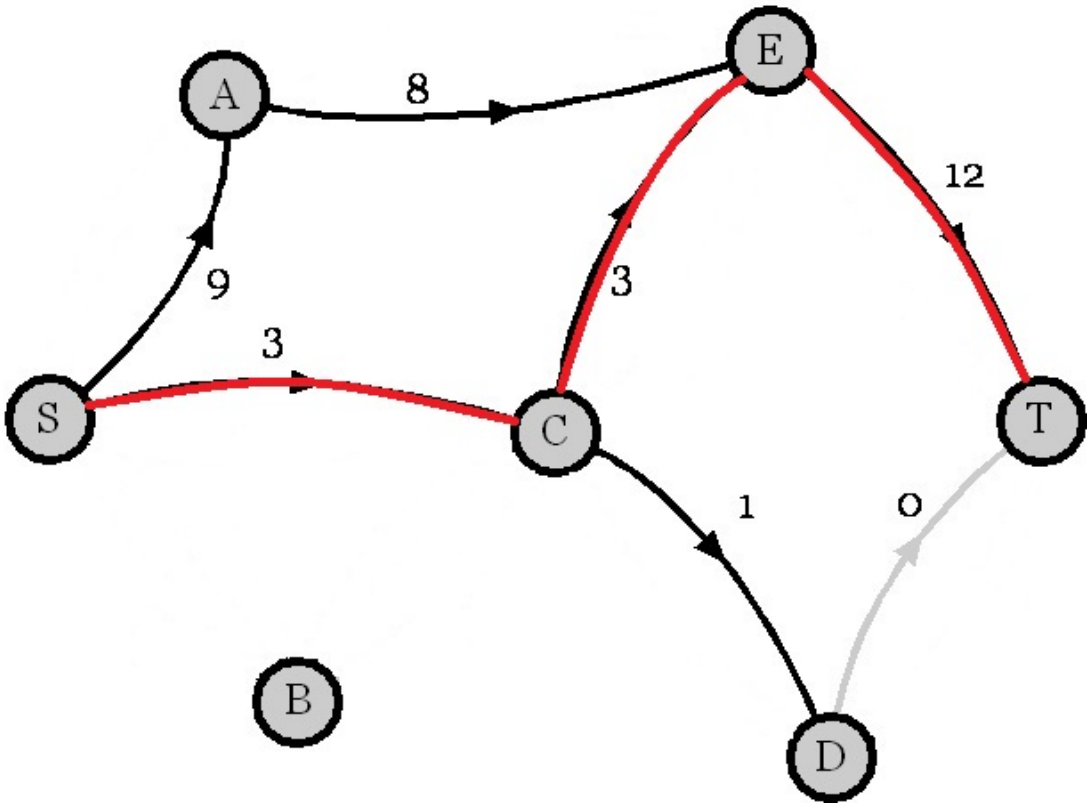
-
- 从S选择出边(S, B)。

LCT Dinitz



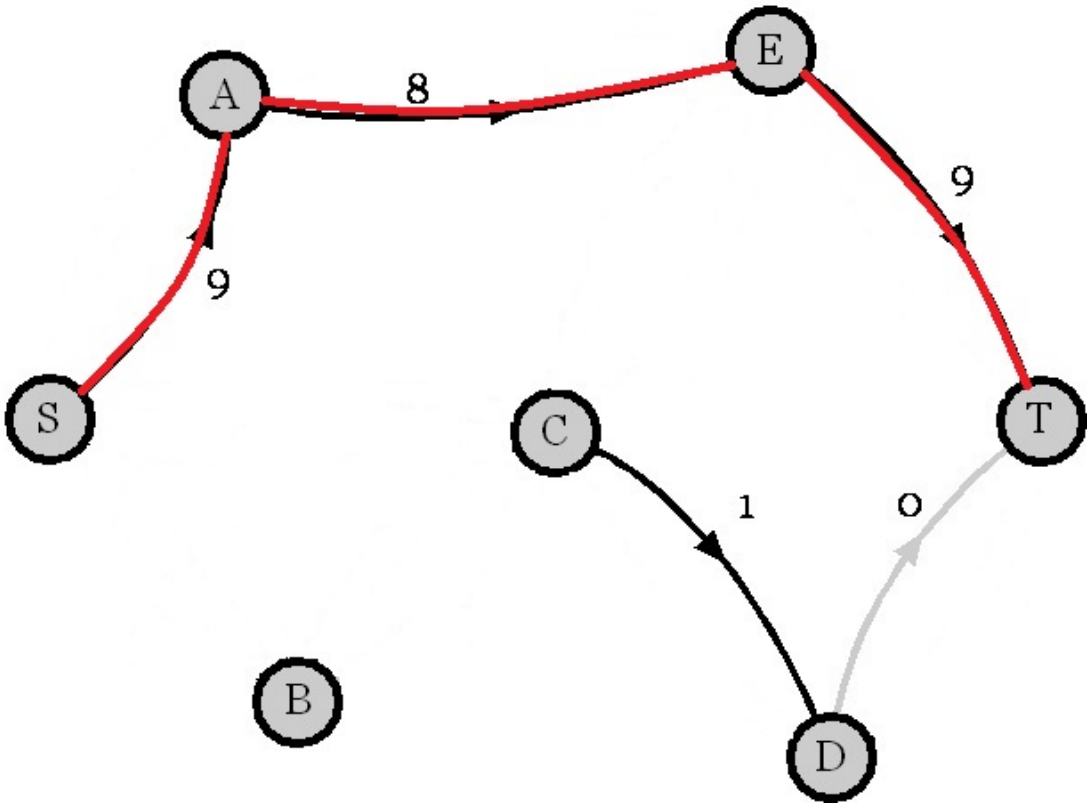
- 然后从B无法找到出边，删去入边(S, B)。

LCT Dinitz



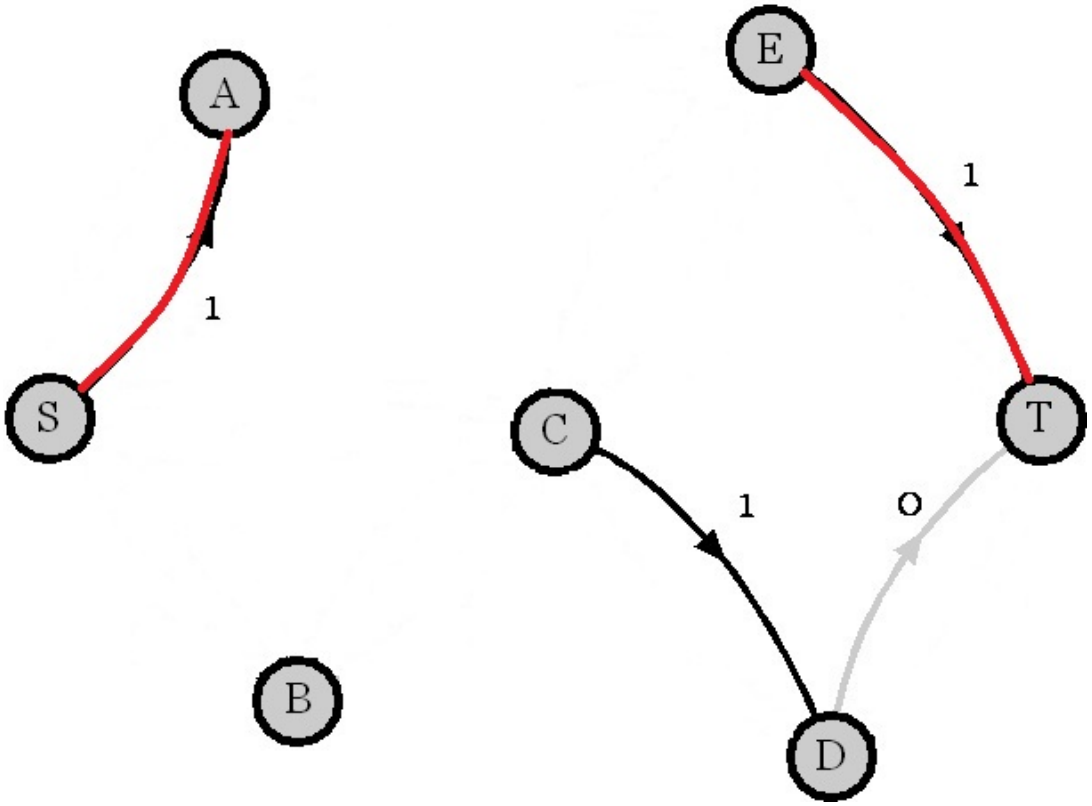
- 之后从S选择出边(S, C)，随后依次选择(C, E)、(E, T)。

LCT Dinitz



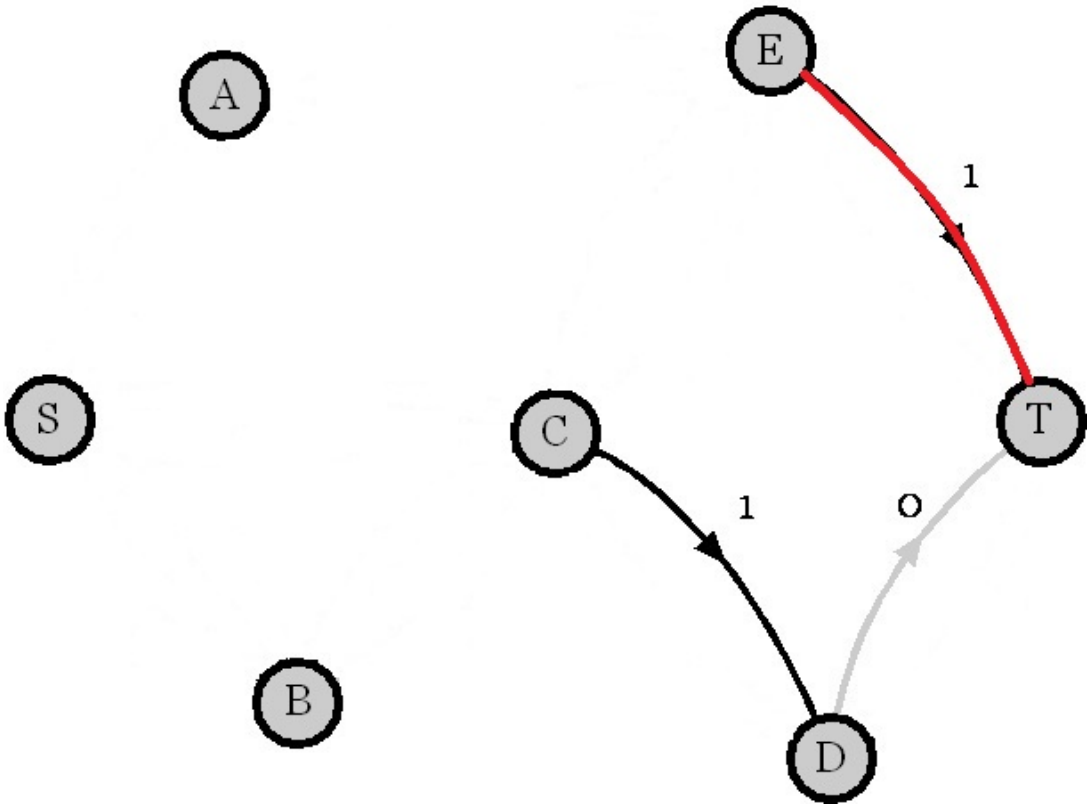
- 此时S, T相连，将路径上的所有边均减去最小的剩余容量3，并且删去所有0边；随后依次选择(S, A)、(A, E)。

LCT Dinitz



- 将S, T路径上的边减去均减去8，并且删去所有0边。

LCT Dinitz

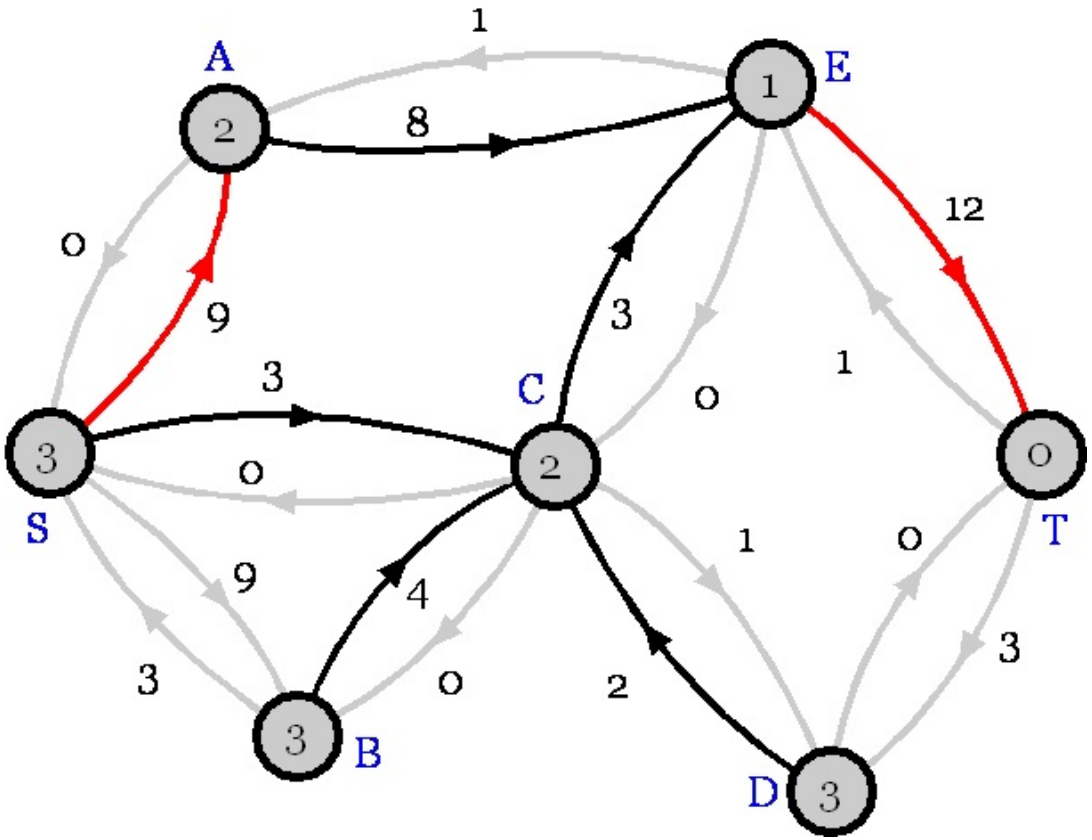


- 从A出发无法找到出边，删去入边(S, A)，此时S同样没有出边，一次阻塞流的寻找结束。

LCT SAP

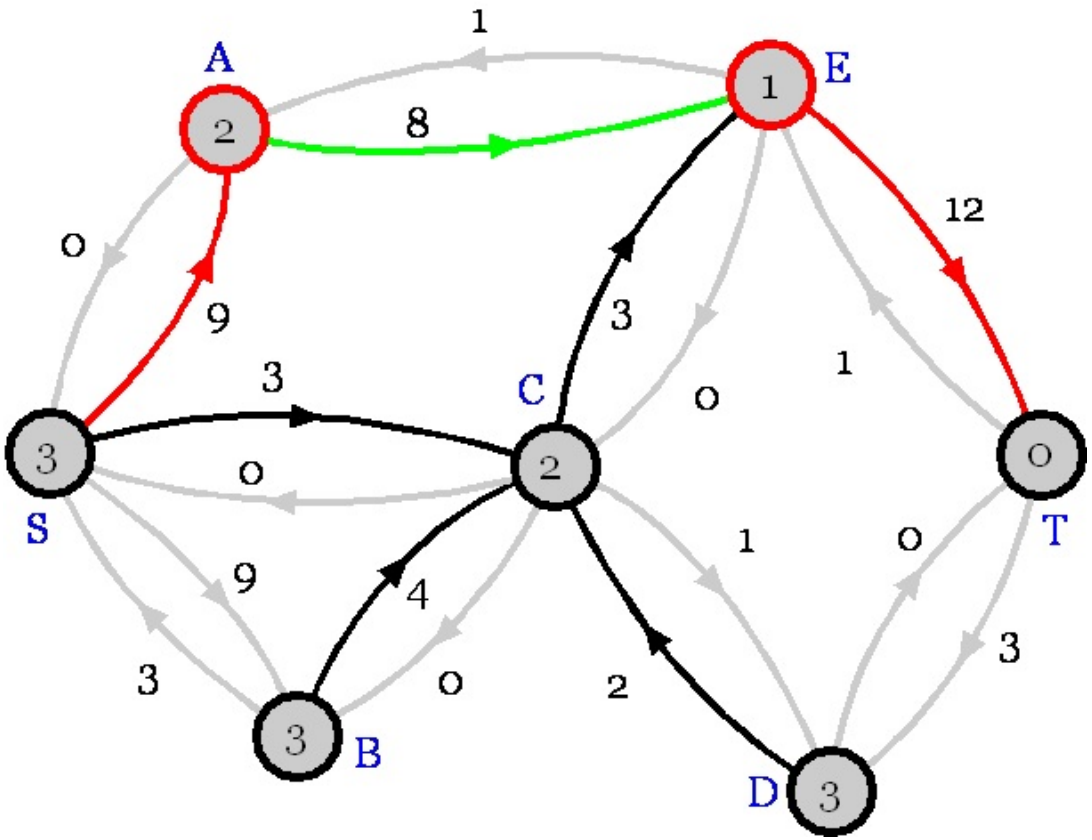
- 实际上对SAP进行优化也并不麻烦，只需要在dinic寻找阻塞流的算法上略加修改。我们在算法同时维护*dist*表示为到*t*的(期望)距离，每次选取边时需要满足SAP中有效边的定义，并且在没有出度时，在删除树中的边后对结点进行重标号。
- 显然我们不能像之前那样把边从图中删去。
- 这里最麻烦的地方就是我需要同时维护反向边的剩余容量。

LCT SAP



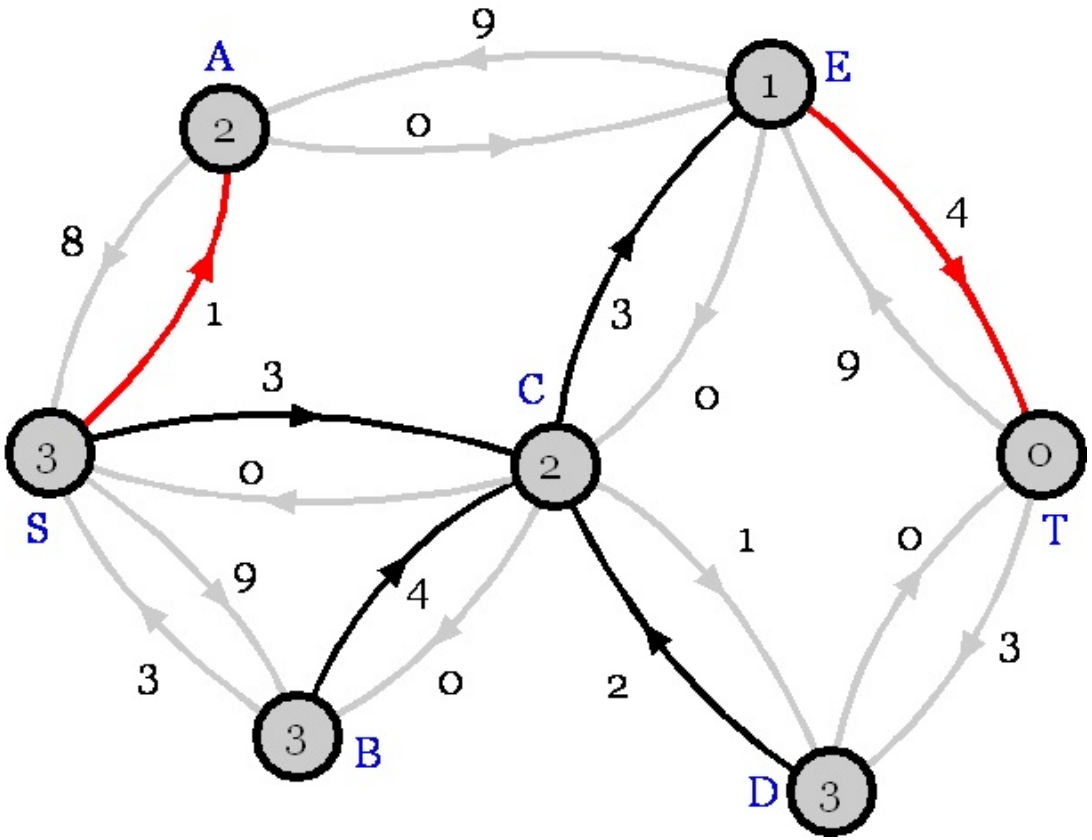
- 我们以这个流网络为例，我们选取算法中途的某个状态，此时红黑边均为有效边，红边为树中的边。

LCT SAP



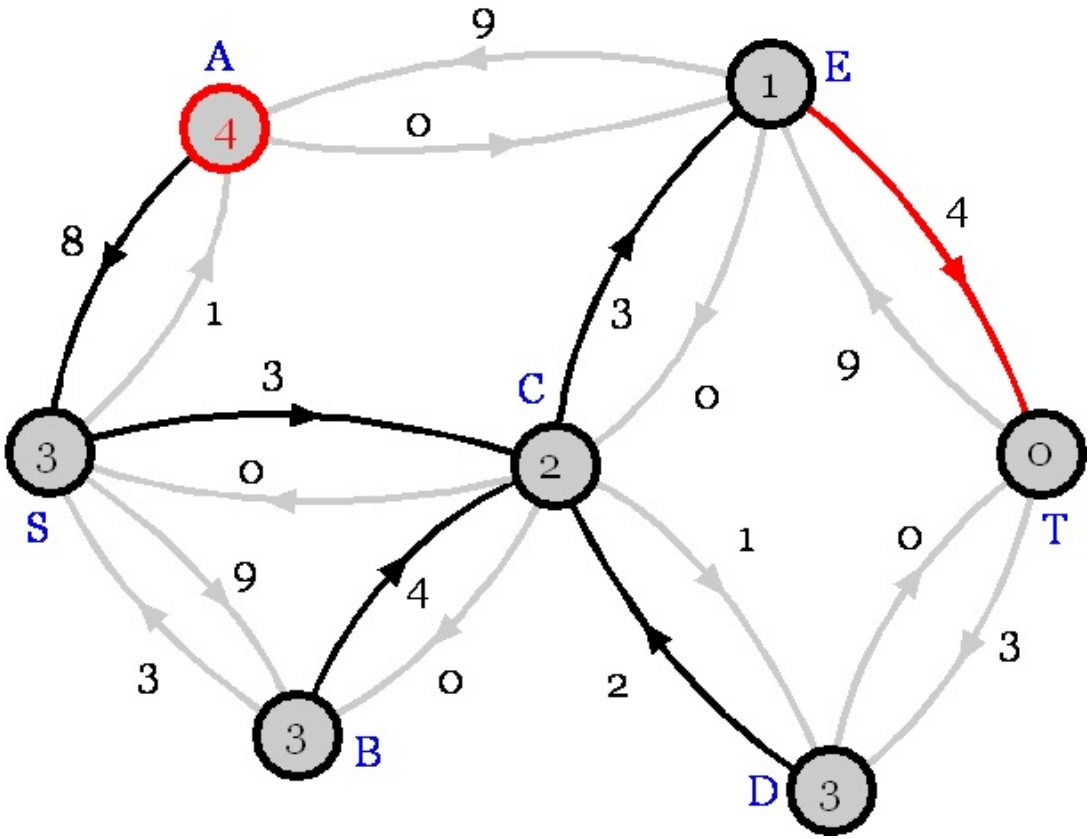
-
- 从A出发选择(A, E)。

LCT SAP



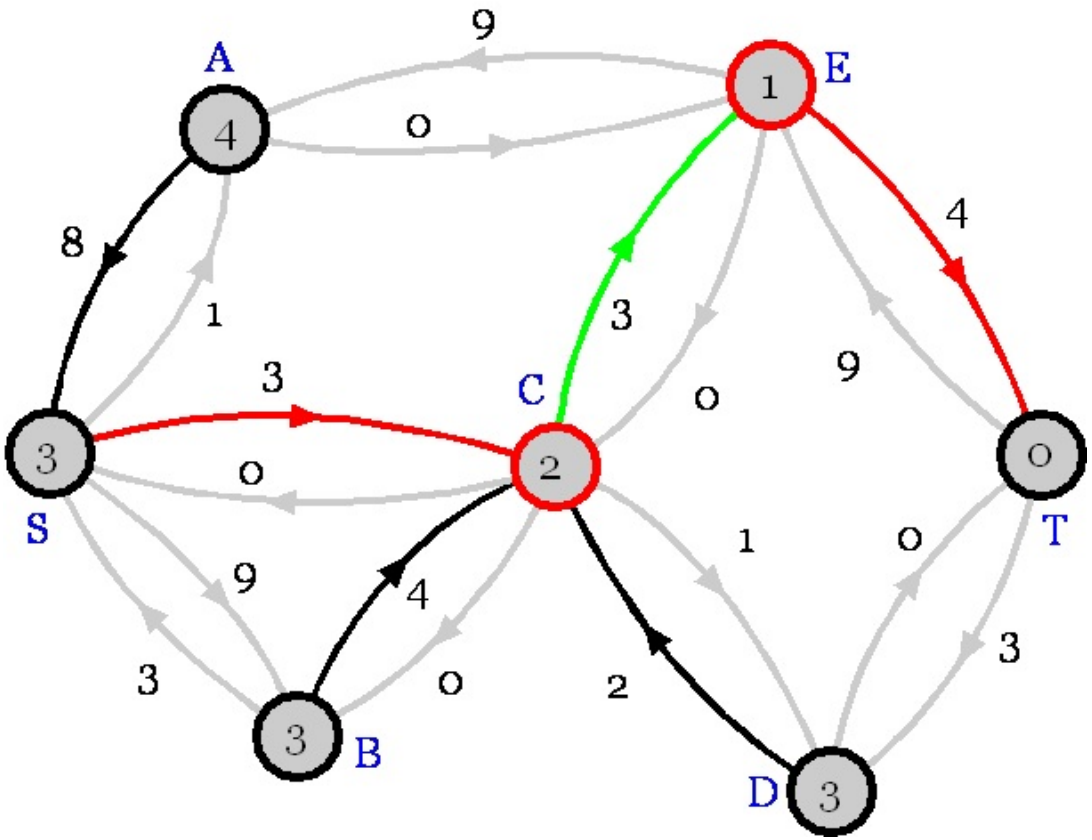
- 此时S, T相连，将路径上的所有边的剩余容量减去8，其反向边增加8。

LCT SAP



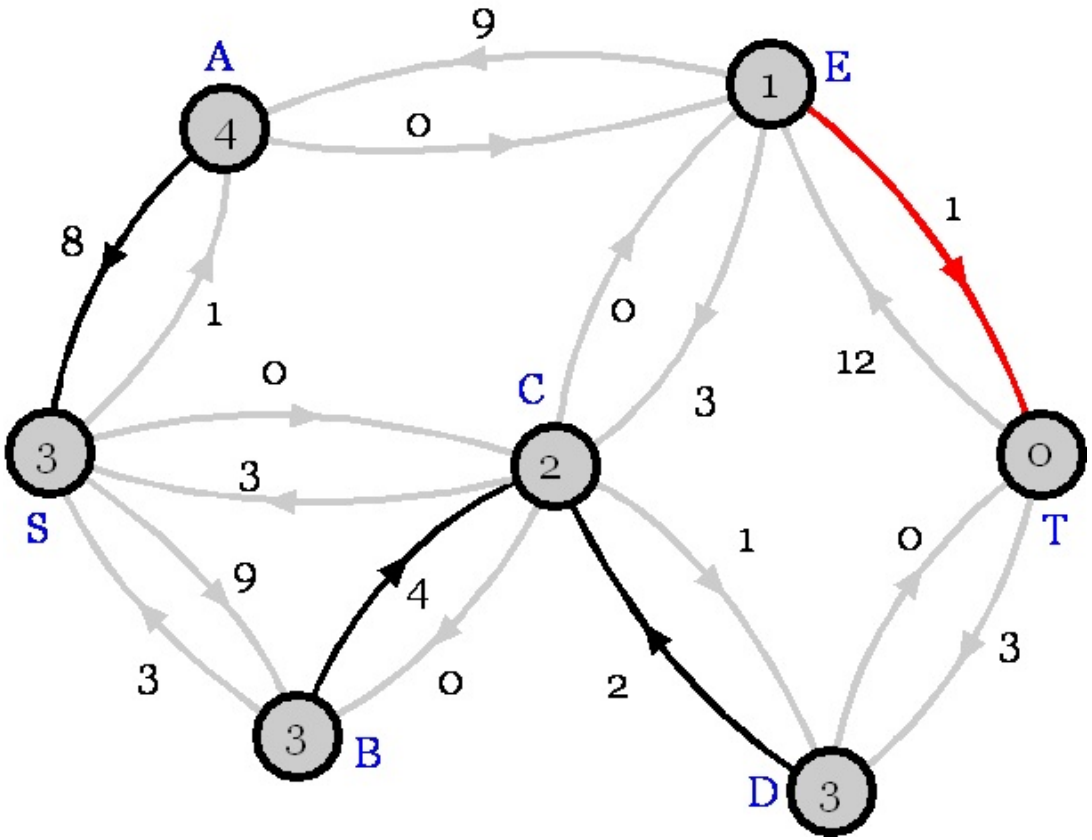
- 继续从A寻找出边，此时不存在出边，删去边(S, A)，修改$dist(A)$。

LCT SAP



-
- 继续选择(S, C)、(C, E)。

LCT SAP



- 此时S, T相连，将路径上的所有边的剩余容量减去3，其反向边增加3。

Complexity

- 我们知道最多$O(VE)$次增广，每次增广对应一次*update*，增广部分时间复杂度$O(VE \log V)$。
- 对于LCT优化的dinic算法，我们注意到每条分层图中的边最多进入树中1次，最多被切断1次，这涉及到$O(E)$次LCT上的操作，因此求一次阻塞流的复杂度为$O(E \log V)$，总时间复杂度为$O(VE \log V)$。
- 对于LCT优化的SAP算法，在*link*和*cut*操作上，因为$0 \leq dist(x) \leq |V|$，因此这里的复杂度也为$O(VE \log V)$。

Complexity

- 但是实现之后，对于OI中的题目优化效果并不明显[1]，可能是由于LCT的常数太大，并且一般题目建图的规模不大且具有特殊性。
- 所以只能理性愉悦一下了。

[1]感谢林永迪同学提供的代码实现

Push-relabel

- 刚学网络流的时候，老师会这样跟我们说，你就把边想成水管，然后流量就是水流。我当时就觉得很不靠谱，自然界是有叫洪水的东西对吧。
- 目前为止我们的算法都基于增广路的寻找，我们一直保证当前的流满足流守恒。假如我们观察从高处倒下的水，水流会向低处前进，直到形成一条或多股稳定的溪流，在此之前并不满足流守恒。我们可以从源点倒入许多水，水向低处流动，在每个存在积水(盈余的流量)的结点，让水流向高度低的节点。

Push-relabel

- 事实上确实有这样的算法，叫做预流推进法。
- 我们定义一个**高度函数**h表示一个结点的高度(类似至t的(期望)最短距离)，同时根据这个定义出**可推流边**(类似有效边)。一个结点可以预先存储一些流量(积水)，这些有预流的结点被称为**活跃点**。

Push-relabel

- 算法过程如下：
- 开始先从s向相邻的结点推送尽可能多的流量，将这些顶点设为活跃点；
- 每次选取一个活跃点u，沿着所有可推流边(u, v)向v推送尽可能多的流量，并将v设为活跃点；如果不存在活跃点，算法结束；
- 如果u仍然存在盈余的流量，对u进行重标号$h(u) = min\{h(v) + 1 : (u, v) \in E_f\}$；否则$u$变为非活跃点。跳转上一步。

Complexity

- 在考虑预流推进法的复杂度之前，我们需要在算法中加入**当前弧优化**，在非递归的SAP中同样有这个概念。显然对于一个结点，在高度未变更之前，前一次推流后无法使用的边，当前一定无法使用。
- 我们可以对每个节点记录一个$current_edge(x)$。每次当该边无法使用时，跳转下一条x的出边；
  当$current_edge(x)$指向$end()$时，对x进行重标号，并将其重置。

Complexity

- 影响复杂度的是对于活跃点的选取方式，如果我们用队列维护活跃点，并且使用FIFO策略，我们得到$O(V^3)$的复杂度。
- 如果我们使用优先队列维护活跃点，每次选取高度最高的活跃点，我们得到$O(V^2\sqrt{E})$。当然如果你不慎写错程序，选取了高度最小的点，复杂度就会变成$O(V^2E)$。
- 在wiki上我看到可以使用动态树将复杂度优化至$O(VElog(V^2/E))$，有兴趣的同学可以自己去查阅。

SPFA费用流

- 对于费用流，我们一定知道(?)用SPFA寻找增广路来做，我假设这个算法叫做SPFA费用流。其思路与EK算法类似，每次寻找一条费用最小的增广路进行增广。
- 但是最短路显然不止一条，我们可以学习dinic算法来改进当前的费用流算法。
- 我们从s开始做SPFA，得到每个点的距离标号$dist$，对于边(u, v)，$dist(v) \leq dist(u) + w(u, v)$。
- 模仿dinic，我们定义满足$dist(v) = dist(u) + w(u, v)$的边$(u, v)$为有效边，有效边组成的图为分层图，直接增广即可。

ZKW费用流

- 但是总感觉效率还是不够高，不是有一个叫做ZKW费用流的算法么。
- 我们学习SAP的距离标号定义方式，用$dist(u)$表示每个点到t的(期望)距离，在不存在$u \to t$的路径时，修改$dist(u)$即可。
- 但是修改距离标号成为了一个很大的问题，距离显然不是增加1了，那到底是多少呢？

ZKW费用流

- 我们在增广时记录所有访问过的点集$\mathbf{V}$，如果$t \in \mathbf{V}$，清空点集重新增广；否则我们知道$\mathbf{V}$中所有点的距离标号都需要增加了，为了满足最短路性质，我们选取$d = min_{i \in \mathbf{V}, j \notin \mathbf{V}, c_f(i,j)>0}(w(i,j) - dist(i) + dist(j))$，将$\mathbf{V}$中所有点的$dist$增加d。
- 如果我们将边的费用$w(i,j)$改变为$w'(i,j) = w(i,j) - dist(i) + dist(j)$，我们就得到了ZKW费用流。

ZKW费用流

- 需要特别注意，如果图中存在容量大于0且费用小于0的边，我们不能将距离标号初始成0，因为可能存在到t的负费用路径。

其他

- 随着科技的进步，最大流的复杂度显然能做到更好，在2012年，来自MIT的James B. Orlin已经研究出了$O(VE)$的最大流算法，甚至对于$|E| = O(V)$的图能做到$O(V^2/\log V)$。
- 不过怎么看像是一堆算法拼起来的。
- 另外我们知道网络流问题本质都是线性规划的问题，直接用求解线性规划的方法来解网络流问题也是一个可行的方法，而且总有人号称单纯形法代码更短，效率更高。
- 我想了想employee这题，不禁陷入了深深的思考。

贪心初始流

- 对于没有下界的网络流，我们一般以零流作为初始流；而有下界的网络流可以经过转化先求出可行流。
- 但是零流作为初始流是不是会被卡呢？
- 答案是肯定的，比如ZOJ2364。题目给出一个分层图，要求求出一个阻塞流。
- 用dinic或SAP求最大流会TLE，而最高标号预流推进可以轻松通过此题。
- 但还可以想办法抢救一下。

贪心初始流

- 首先将节点按照层次排序，每个节点维护两个信息**流入流量**in_i和**流出流量**out_i；
- 设$in_s = +\infty$，表示s初始有正无穷的流量，从s开始考虑每个结点u，在保证$in_u \geq out_u$的情况下尽可能向下一层结点v推送流量，同时增加in_v和out_u。该步结束后，out表示了每个结点能够向下一层推送的最多流量。
- 将in清空，设$in_t = +\infty$，表示t能吸收正无穷的流量，从t开始考虑每个结点v，在保证$out_u \geq in_u$的情况下尽可能从上一层结点u吸收流量，同时增加in_u，减少in_v，修改边的剩余容量。该步中，in表示每个结点还能够吸收的流量。

贪心初始流

- 目前我只找到对于分层图的贪心初始流，对于一般图也许可以用各种方法找出一个较优的初始流。
- 不过至少可以用其优化dinic算法。

Problem

- 因为我到现在还没把线性规划与网络流24题看过一遍，所以就只介绍作为工具的算法。至于网络流的各种模型，我就不在此误导各位了> _ <。
- 但是大爷们看起来特别想秒题的样子，所以我准备了一些简单的题目。

预警

- 1. 所有题面经过改编，故事均发生在某个与我们相距很远的平行宇宙，一切内容均不属实。
- 2. 题目解法不一定与网络流有关系。

LIS [2]

题目描述

当地球上比走大爷更厉害的程序猿(媛)都前往半人马座阿尔法星系参加UOI时，地球却遭到了达斯维达大人的攻击。敌人的N艘战舰一字排开，从左至右每艘星舰拥有3个属性：规模A_i，摧毁代价B_i，额外属性C_i。达斯维达大人其实患有强迫症，如果战舰被摧毁，规模A的最长上升子序列长度减小了，他就会撤退；地球的指挥官也有强迫症，他要求走大爷给出最小摧毁代价的情况下，额外属性排序后字典序最小的摧毁方案，否则他就不进行反击。

数据范围

$1 \leq N \leq 700, \quad 1 \leq A_i, B_i, C_i \leq 10^9, \quad C_i$ 两两不同。

[2]Source: SDOI 2014 Round1 day2

Solution

- 令f_i为以i结束的最长上升子序列长度，建图：
- 如果$i < j, \quad f_i = f_j - 1, \quad A_i < A_j$, 连边$(i, j)$;
- 如果$f_i = 1$, 连边(s, i);
- 如果$f_i = MAX\{f_i\}$, 连边(i, t).
- 问题等价于删去非s, t的点，使得s, t不连通。
- 拆点之后考虑最小割即可。

Solution

- 之后按C升序考虑每一项，判断对应的边是否能属于某个最小割。
- 对于边(u, v)，我们检查剩余网络中是否存在u到v的路径即可。但是删去一条边后，如果重新计算最大流会TLE。

Solution

- 事实上我们可以尝试在剩余网络上经过边(u, v)退流:
- 寻找从t到v流量为c的流，寻找从u到s流量为c的流。
- 其中c为边(u, v)原先的流量。

FoxAndCity [3]

题目描述

走大爷帮助击退了达斯维达大人，中国的少女们都对他产生了浓浓的爱慕之情，位于N个城市(从0开始标号，走大爷在0号城市)的妹子想要找他____。一些城市之间存在道路连接，并且一条道路的长度都认为是1，妹子们同时出发，且会沿着最短的路前进。然而同时应付N个城市的妹子着实是个头疼的事情，走大爷希望能向政府要求建立一些道路，使得每个城市的妹子到达时间与自己期望的时间的差值的平方和最小，求这个最小值。

数据范围

$2 \leq N \leq 40$,　0号城市的期望时间为0。

[3]Source: Single Round Match 590 Round 1 - Division I, Level Three

Solution

- 初看这题似乎和网络流没有啥关系，我们需要考虑至0的最短路长度$dist$。
- $dist_0 = 0$;
- 对于原图中的边(u, v)，有$|dist_u - dist_v| \leq 1$;
- 我们相当于对每个点确定一个满足以上规则的$dist$，然后使得代价最小。

Solution

- 我们使用最小割。
- 因为最短距离不超过$N-1$，我们可以将一个点拆成$N-1$个点($dist_i \leq k$)。在这里的最小割中，我们定义属于T集的点的事件为真。
- 通过添加容量为$+\infty$的边(u, v)，我们可以使得，如果v属于T集，那么u也必定属于T集。在这里表现为，如果v的事件为真，那么u的事件也为真。

Solution

- 从s向每个($dist_i \leq 0$)连边，容量为将$dist_i$设置成0的代价，其中除了点0以外，此代价均为$+\infty$;
- 每个($dist_i \leq N-1$)向t连边，容量为$+\infty$，因为该事件必定成立;
- 每个($dist_i \leq k-1$)向($dist_i \leq k$)连边，容量为将$dist_k$设置成k的代价;
  其中点0的这部分边的容量都应当设置成$+\infty$;

Solution

- 对于$|dist_u - dist_v| \leq 1$，我们将其拆成两个条件：
- $dist_u \leq dist_v + 1$;

  $dist_v \leq dist_u + 1$;
- 对于原图中的边(u, v)，我们只需要建立两条容量为$+\infty$的边即可：
- $(dist_u \leq k) \rightarrow (dist_v \leq k - 1)$;

  $(dist_v \leq k) \rightarrow (dist_u \leq k - 1)$;
- 按照等式，如果$dist_i \leq k$成立，$dist_i \leq k + 1$显然成立，但实际上我们不需要专门为这个添加边。

BoardPainting [4]

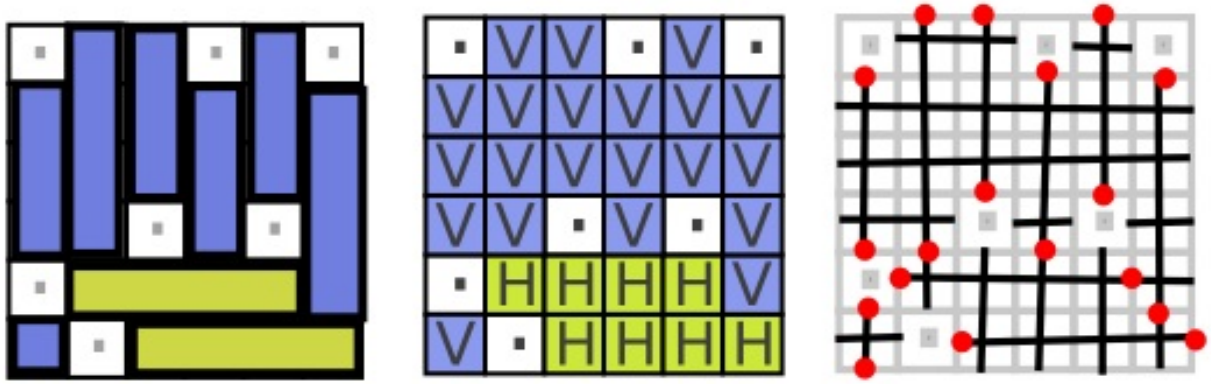
题目描述

不久之后，达斯维达大人又卷土重来，这次他率领了一个$N \times M(1 \leq N, M \leq 50)$方阵的战舰，每个战舰完全相同。地球的指挥官发现这其中一些只是影像而已，他画出地图，其中#表示真实的战舰。但是战舰实在太多了，指挥官放弃了进攻。当地球陷入水深火热之中时，马老板表示可以使用一款叫做天天爱消除的游戏，借助上帝的力量进行反击。每次可以选择横向或纵向连续的一些战舰进行消除，并且不能选择影像或者已经被消除的战舰。走大爷想知道最少需要消除几次。

[4] Source: Single Round Match 577 Round 1 - Division I, Level Three

Solution

- 题目其实是要求用尽可能少的一边宽度为1的矩形覆盖所有的#，并且不能重叠或覆盖到空格。
- 矩形只会有横向或纵向两种，我们将横向矩形覆盖的格子标记为H，纵向的记为V。



- 矩形的个数与两类格子之间的边，以及V格在纵向与边界或空格、H格在横向与边界或空格的边(即图中红点)有关。

Solution

- 显然可以应用最小割，其中某点属于S集表示该格为V格，否则为H格。
- 从s向每个#格子连边，容量为该格横向邻接的空格个数(边界外均认为是空格);
- 从每个#格子向t连边，容量为该格纵向邻接的空格个数;
- 相邻两个#格之间连接双向容量为1的边。
- 求最小割，除以2即为答案。

CurvyonRails [5]

题目描述

在战争平息之后，走大爷开始经营他的后宫。走大爷的土地被划分为$N \times M(1 \leq N, M \leq 25)$的网格，每个格子可能是草原或是城镇，每个城镇可能属于一个妹子。为了更好地在自己的土地上行动，走大爷打算在所有城镇里建立铁路。铁路不能经过草原，每个城镇内的铁路连接了四个方向中的两个，且两头必须和其余铁路连接。而妹子们都具有深深的腐女气质，因此她们不喜欢直的铁路。如果她们拥有的城镇里的铁路是直的，她们就会非常不开心。走大爷需要让尽可能少的妹子不开心。

[5] Source: Single Round Match 570 Round 1 - Division I, Level Three

Solution

- 如果我们不考虑妹子们，直接使用黑白染色即可。从s向每个白点连容量为2的边，从黑点向t连容量为2的边，然后白点向相邻的黑点连容量为1的边即可。
- 现在如果考虑转角的问题，可以把每个格子拆成两个，一个表示水平方向，一个表示垂直方向。
- 从s向每个白点连容量为1的边，从每个黑点向t连容量为1的边，从每个白点向对应方向的黑点连容量为1的边。

Solution

- 现在我们使得每个城镇都是转角了。但是可能会无解，我们需要在水平点和垂直点之间交换流量。
- 我们在同一个点拆成的两个点之间连边，两个方向的容量都是1。如果这样的边上有流量，就说明这个城镇的铁路是直的。
- 令属于某个妹子的城镇的这样的边费用为1，应用最小费用最大流即可。

OldBridges [6]

题目描述

走大爷的妹子年龄层次丰富，这天他正带着两个小萝莉在水上花园玩耍。花园由N个岛屿组成，其中一些之间连有桥。其中一个小萝莉想要从a_1到a_2，再返回a_1，这样重复a_n趟，而另一个小萝莉想要从b_1到b_2，再返回b_1，这样重复b_n趟。这时走大爷发现其中一些桥因为年久失修，已经成为了危桥，只能让小萝莉通过至多2次。如果阻止小萝莉，她们会不开心。所以走大爷想知道是否存在安全的路线，让小萝莉开心玩耍。

数据范围

$4 \leq N \leq 50$, ~~小萝莉的年龄为11岁~~.

[6] Source: Single Round Match 556 Round 1 - Division I, Level Three

Solution

- 显然如果要应用最大流，我们肯定会将危桥的容量设为2，并且将非危桥的容量设为$+\infty$。
- 之后我们需要考虑危桥从任意方向只能通过2次。
- 最终的算法非常简单，我们只需要设危桥双向流量均为2，检查从$\{a_1, b_1\} \to \{a_2, b_2\}$的最大流以及$\{a_1, b_2\} \to \{a_2, b_1\}$的最大流是否均大于等于$2 \times (a_n + b_n)$即可。
- 其中我们需要利用多源多汇的网络流，a部分流量设为$2 \times a_n$，b部分为$2 \times b_n$，因此实际上最大流至多为$2 \times (a_n + b_n)$。

Solution

- 接下来我们需要证明这样是正确的。
- 我们先令前者的流为F_1，后者为F_2。
- 显然路径可以反向行走，一条从$a_1 \to a_2 \to a_1$的路径可以认为是从$a_1 \to a_2$的2的流量，对于b也同理，因此F_1及F_2的流量必定大于等于$2 \times (a_n + b_n)$。
- 因此这是个必要条件。

Solution

- 如果我们认为$+\infty$为偶数，那么因为所有边的容量均为偶数，F_1, F_2的流量必定为偶数。
- 令$F_A = (F_1 + F_2)/2$，即对所有的边u，$F_A(u) = (F_1(u) + F_2(u))/2$; 令$F_B = (F_1 - F_2)/2$。
- 显然对于任意一条边u，$|F_A(u)| + |F_B(u)| \leq capacity(u)$,
- 因为$|(F_1(u) + F_2(u)|/2 + |(F_1(u) - F_2(u))|/2 \leq max\{F_1(u), F_2(u)\}$,
- 那么F_A就可以作为a的路线，F_B可以作为b的路线，这个条件是充分的。

TheTilesDivOne [7]

题目描述

走大爷意识到自己$N \times M(1 \leq N, M \leq 47)$网格的土地有太多的草原，他打算将其划分出一些牧区发展畜牧业。但妹子秉承着弯的哲♗学，给走大爷提出了一些要求:

(1)每个牧区必须呈L形，即面积为3的转角;

(2)牧区不能重叠或占据城镇;

(3)将网格黑白染色，左上角为黑色，L形的转角处必须是黑色;

(4)L形可以进行旋转;

走大爷想划分出尽可能多的牧区。

[7]Source: Single Round Match 575 Round 1 - Division I, Level Three

Solution

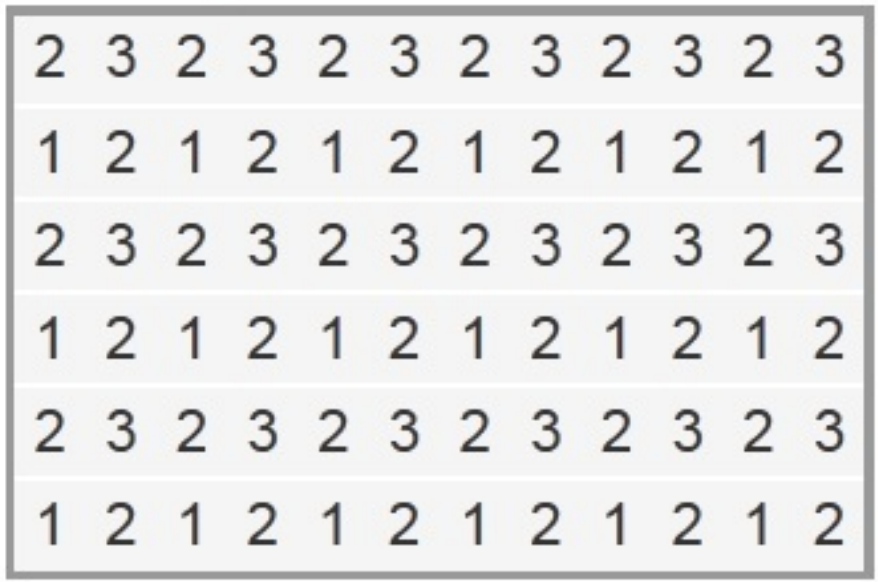
- 如果仅仅是黑白染色，从白色格子连出1的边，从黑色格子连出2的边，我不能确保牧区确实是弯的。
- 那么尝试一下用三种颜色染色。

Solution

- 1:偶数行的白格;

  2:黑格;

  3:奇数行的白格;



- 容易发现，相邻的3个颜色为1,2,3的格子将会组成一个合法的牧区。

Solution

- 对于每个颜色为1的格子u，从s向u连接容量为1的边;
- 对于每个颜色为3的格子u，从u向t连接容量为1的边;
- 对于相邻的格子u, v，如果u的颜色$+1$等于v的颜色，从u向v连接容量为1的边;
- 为了确保每个格子最多被使用1次，将每个格子u拆成两个点in_u和out_u即可。

GoodCompanyDivOne [8]

题目描述

杜老板的把妹学校有N个员工，除了0号员工，其余每人有一个编号小于自己的上级或没有上级。每个员工以及其直接下属构成一个部门，一个部门是好的当且仅当成员可以教别人不同的把妹技巧。杜老板打算让自己下属每人向走大爷学习两种把妹技巧，并且能使得每个部门都是好的。现在知道走大爷可以教K种把妹技巧，教一个人第i种技巧需要收费$cost_i$，求杜老板最少的花费。

数据范围

$$1 \leq N \leq 30, \quad 2 \leq K \leq 30, \quad 1 \leq cost_i \leq 100.$$

[8] Source: Single Round Match 619 Round 1 - Division I, Level Two

Solution

- 定义$f(x,p)$表示考虑完了以x为根的子树，x学的其中一项技巧为p的最小代价。

Solution

- 使用最小权匹配求解$f(x,p)$。
- 其中对于所有的儿子s_i，在这个部门中教授技巧p_i的代价为$f(s_i,p_i)$;
- 对于x，在这个部门中教授技巧$p_i \neq p$，代价为$cost_i$，教授技巧p，因为本来就需要x学习p，因此只需要选择非p的代价最小的技巧即可。
- 之后枚举每个无上级的员工学的其中一项技巧即可。

References

- Wikipedia Dinic's algorithm
- Wikipedia Maximum flow problem
- Wikipedia Push – relabel maximum flow algorithm
- 袁昕颢《动态树问题及其应用》
- 鲁逸沁《网络流问题选讲》
- A Date Structure for Dynamic Trees
- Max flows in O(nm) time, or better
- Shaymin's hideout 最大流算法——预流推进
- 沐阳 分层图阻塞流 贪心预流

Contact me

- email: shineflyyy@foxmail.com
- QQ: 1020971941

Thanks

- 祝大家省选顺利!