网络流

Disposrestfully

雅礼中学

August 14, 2019

主要是总结网络流的各种套路.

前置技能是最大流和费用流算法,以及相关概念知识.

因为网络流题偏重于思维,需要自行思考,所以为了不破坏各位的思考过程,课件上的题会比较简单,难度较大的题目将以题单的形式给出,并且附上题解,如果在课后要做题的话可以做题单上面的题,不建议去写课件上给出了题解的大部分题目.

讲题人很菜,各种问题都可能有,请多包涵.

Type.1

这类问题的基本形式都是对若干个元素进行决策.

在大多数情况下只会有两种决策,不能同时选也不能同时不选.

不同元素的决策之间可能存在限制/依赖/对答案的额外贡献.

Type.1

典型的最小割建图.

Type.1

典型的最小割建图.

假设在一个点u上我们有两种决策A和B,收益分别为W_A和W_B.

Type.1

典型的最小割建图.

假设在一个点u上我们有两种决策A和B,收益分别为W_A和W_B.

我们把u拆成两个点u_1和u_2,从S向u_1连流量为W_A的边,从u_2向T连流量为W_B的边.割掉某条边代表放弃这种决策.

Type.1

典型的最小割建图.

假设在一个点u上我们有两种决策A和B,收益分别为W_A和W_B.

我们把u拆成两个点u_1和u_2,从S向u_1连流量为W_A的边,从u_2向T连流量为W_B的边.割掉某条边代表放弃这种决策.

两列点中间的边按照题目所给的限制去连.

Type.1

典型的最小割建图.

假设在一个点u上我们有两种决策A和B,收益分别为W_A和W_B.

我们把u拆成两个点u_1和u_2,从S向u_1连流量为W_A的边,从u_2向T连流量为W_B的边.割掉某条边代表放弃这种决策.

两列点中间的边按照题目所给的限制去连.

答案是所有W_A与W_B的和减去最小割.

Type.1

典型的最小割建图.

假设在一个点u上我们有两种决策A和B,收益分别为W_A和W_B.

我们把u拆成两个点u_1和u_2,从S向u_1连流量为W_A的边,从u_2向T连流量为W_B的边.割掉某条边代表放弃这种决策.

两列点中间的边按照题目所给的限制去连.

答案是所有W_A与W_B的和减去最小割.

板题是善意的投票,一个经典的建图是文理分科,没见过的可以之后去看,这里不再赘述.

最大权闭合子图

从刚才的建图方法上做一点(其实没有)延伸,就能得到这个重要的模型.

最大权闭合子图

从刚才的建图方法上做一点(其实没有)延伸,就能得到这个重要的模型.

最大权闭合子图算法运用的依旧是典型的最小割式建图.

最大权闭合子图

从刚才的建图方法上做一点(其实没有)延伸,就能得到这个重要的模型.

最大权闭合子图算法运用的依旧是典型的最小割式建图.

考虑一个点权为W_i的点,若$W_i < 0$,则向T连流量为$-W_i$的边,否则从S向其连流量为W_i的边,原图中的边流量都为inf.

最大权闭合子图

从刚才的建图方法上做一点(其实没有)延伸,就能得到这个重要的模型.

最大权闭合子图算法运用的依旧是典型的最小割式建图.

考虑一个点权为W_i的点,若$W_i < 0$,则向T连流量为$-W_i$的边,否则从S向其连流量为W_i的边,原图中的边流量都为inf.

模板是太空飞行计划,知识点有所欠缺的话可以去补.

价

有n种药材和n种药,每种药由若干种药材制成,且只有一份.

如果令这n种药各自对应一个其使用的药材,存在一种方案使得它们对应的药材两两不同.

第i种药有W_i的价值,由于一些玄妙的原因,如果你使用了k种药,并且它们使用药材的并集大小也为k,这k种药才会发挥作用.

最小化你选择所有药的价值之和.

$n \le 300$

价

每种药的权值设为$W_i - inf$,药材的权值设为inf,选药必须选药材.

价

每种药的权值设为$W_i - inf$,药材的权值设为inf,选药必须选药材.

然后做最小权闭合子图.

价

每种药的权值设为$W_i - inf$,药材的权值设为inf,选药必须选药材.

然后做最小权闭合子图.

原图存在完美匹配,根据$Hall$定理,选了k种药就肯定会选出k种药材.同时为了保证答案的最优性,肯定也不会多选.

Type.2

这类问题一般可以转换为下面这个模型:

Type.2

这类问题一般可以转换为下面这个模型:

一张无向图,第i个点上有a_i个人,边有长度/费用/通过人数限制.

现在需要让第i个点上有b_i个人($\sum a_i = \sum b_i$),求最小代价/是否可行.

Type.2

原图中给定的边不做修改.

Type.2

原图中给定的边不做修改.

对于一个点i,若$a_i > b_i$,则从S向它连流量为$a_i - b_i$的边,费用为0.

Type.2

原图中给定的边不做修改.

对于一个点i,若$a_i > b_i$,则从S向它连流量为$a_i - b_i$的边,费用为0.

否则从它向T连流量为$b_i - a_i$的边,费用同样为0.

Type.2

原图中给定的边不做修改.

对于一个点i,若$a_i > b_i$,则从S向它连流量为$a_i - b_i$的边,费用为0.

否则从它向T连流量为$b_i - a_i$的边,费用同样为0.

当S的所有出边和T的所有入边都满流时存在可行解.

Type.2

原图中给定的边不做修改.

对于一个点i,若$a_i > b_i$,则从S向它连流量为$a_i - b_i$的边,费用为0.

否则从它向T连流量为$b_i - a_i$的边,费用同样为0.

当S的所有出边和T的所有入边都满流时存在可行解.

Coding Contest

n个点,m条道路,每个点上都有一些人和一些食物.

每一条道路在第一次被经过的时候不会发生任何事,后面每被经过一次就有p_i的概率破坏网络.

每条道路最多允许c_i个人走过.

需要给出一种安排,使得每个人都吃上饭,并且网络被破坏的概率最小.

$n \leq 100$

Coding Contest

没什么好讲的.

就按照之前所说的建图方式建图.

p_i取个log,拆下边就是费用流模板了.

上下界可行流

上下界网络流使用的也是调整的思想,先强制流满下界,再调整流量使得每个点流量平衡.

上下界可行流

上下界网络流使用的也是调整的思想,先强制流满下界,再调整流量使得每个点流量平衡.

对于一个点,它的a_i是所有入边的下界之和减去 所有出边的下界之和,b_i是0.

上下界可行流

上下界网络流使用的也是调整的思想,先强制流满下界,再调整流量使得每个点流量平衡.

对于一个点,它的a_i是所有入边的下界之和减去 所有出边的下界之和,b_i是0.

每条边的流量是上下界之差.

上下界可行流

上下界网络流使用的也是调整的思想,先强制流满下界,再调整流量使得每个点流量平衡.

对于一个点,它的a_i是所有入边的下界之和减去 所有出边的下界之和,b_i是0.

每条边的流量是上下界之差.

没有写过上下界的同学可以写一下这道,支线剧情.

上下界最大流

在求解上下界最大流的时候,首先去除源汇,处理出可行流.

上下界最大流

在求解上下界最大流的时候,首先去除源汇,处理出可行流.

然后恢复源汇,在剩余网络上求最大流,两次流量的和即为答案.

上下界最大流

在求解上下界最大流的时候,首先去除源汇,处理出可行流.

然后恢复源汇,在剩余网络上求最大流,两次流量的和即为答案.

最小流在第二步把源汇倒过来跑,然后两次流量相减.

上下界最大流

在求解上下界最大流的时候,首先去除源汇,处理出可行流.

然后恢复源汇,在剩余网络上求最大流,两次流量的和即为答案.

最小流在第二步把源汇倒过来跑,然后两次流量相减.

然后放一个板题,清理雪道.

Type.3

给一张图,要求选出若干条路径.

每条边/每个点被选中的次数有限制/每条边有代价.

一般来说会指定路径的起点和终点.

Type.3

对于一条路径(a, b),从S向a连边权为1的边,从b向T连边权为1的边.

满流则存在可行解,边有代价的话就是费用流.

部分题目可能需要一些讨论.

危桥

n个点的无向图,一些边只能经过两次,其他的边经过次数没有限制.

$Alice$想在A_1和A_2之间往返A_n次,Bob想在B_1和B_2之间往返B_n次.

求是否存在可行方案.

$n \leq 50$

危桥

建图方式很显然,但是如果直接跑网络流的话会出问题.

危桥

建图方式很显然,但是如果直接跑网络流的话会出问题.

可能出现A_1不能到A_2,结果流到了B_2的情况.

危桥

建图方式很显然,但是如果直接跑网络流的话会出问题.

可能出现A_1不能到A_2,结果流到了B_2的情况.

在这种时候我们可以把B_1和B_2换一下,还是满流就有解.

Type.4

操作/限制存在阶段性.

Type.4

操作/限制存在阶段性.

通常与时间相关,与二分答案/枚举答案搭配使用.

Type.4

操作/限制存在阶段性.

通常与时间相关,与二分答案/枚举答案搭配使用.

这类问题建图方式多种多样(一般不会是最小割),但大方向上很好辨别.

Type.4

操作/限制存在阶段性.

通常与时间相关,与二分答案/枚举答案搭配使用.

这类问题建图方式多种多样(一般不会是最小割),但大方向上很好辨别.

讲题人人实在不知道该怎么描述这类模型,现在我们来一个板题.

Blue Mary的旅行

n个城市,m条航线,所有航线起飞和到达的时间都相同,且每天只飞一次.

一共有第T个人要从A城前往B城.

第i条航线每天最多能买到C_i张票.

求所有人到达B城的最早时间.

$n \leq 50$

Blue Mary的旅行

分层图,枚举答案.

Blue Mary的旅行

分层图,枚举答案.

对于一条航线$[st, ed, C_i]$,从第k天的点st向第$k+1$天的点ed连流量为C_i的边.

Blue Mary的旅行

分层图,枚举答案.

对于一条航线$[st, ed, C_i]$,从第k天的点st向第$k+1$天的点ed连流量为C_i的边.

对于每个i,从第k天的点i向第$k+1$天的点i连流量为inf的边.

Blue Mary的旅行

分层图,枚举答案.

对于一条航线$[st, ed, C_i]$,从第k天的点st向第$k+1$天的点ed连流量为C_i的边.

对于每个i,从第k天的点i向第$k+1$天的点i连流量为inf的边.

源点为第一天的A,汇点为最后一天的B,求最大流.

Type.5

比较特殊的线性规划问题.

Type.5

比较特殊的线性规划问题.

一个常见套路是将题目的等量关系转化为流量平衡.

Type.5

比较特殊的线性规划问题.

一个常见套路是将题目的等量关系转化为流量平衡.

最经典的志愿者招募.

Delight for a Cat

长度为n的字符串,每个位置可以填入S或E.

在第i位填入S可以获得s_i的收益,填入E可以获得e_i的收益.

要求连续的k个位置上包含至少t_1个S和t_2个E.

求最大收益及方案.

$1 \leq k \leq n \leq 1000$

Delight for a Cat

令x_i表示第i个位置是否填入S,并且钦定所有位置最开始都填入E.

Delight for a Cat

令x_i表示第i个位置是否填入S,并且钦定所有位置最开始都填入E.

问题变成了最大化$\sum_{i=1}^{n} x_i(s_i - e_i)$.

且对于任意的p都有$t_1 \leq \sum_{i=p}^{p+k-1} x_i \leq k - t_2$

Delight for a Cat

令x_i表示第i个位置是否填入S,并且钦定所有位置最开始都填入E.

问题变成了最大化$\sum_{i=1}^{n} x_i(s_i - e_i)$.

且对于任意的p都有$t_1 \leq \sum_{i=p}^{p+k-1} x_i \leq k - t_2$

然后我们添加一些辅助变量,把式子列出来,再移下项,把所有的常数都放到一边,并且令这一边的值大于零.可以发现所有的变量在式子中只出现两次,且一次在左一次在右.

Delight for a Cat

把一个等式看做一个点,左边看做流入,右边看做流出.

Delight for a Cat

把一个等式看做一个点,左边看做流入,右边看做流出.

把一个变量看成一条边,连接两个等式,流量为其值域,费用为收益.

Delight for a Cat

把一个等式看做一个点,左边看做流入,右边看做流出.

把一个变量看成一条边,连接两个等式,流量为其值域,费用为收益.

常数项看做源汇点连的边,流量为这一项的值,费用为零,最后做费用流.

Delight for a Cat

把一个等式看做一个点,左边看做流入,右边看做流出.

把一个变量看成一条边,连接两个等式,流量为其值域,费用为收益.

常数项看做源汇点连的边,流量为这一项的值,费用为零,最后做费用流.

输出方案看哪些边满流即可.

Type.6

当网络满足一些特殊性质的时候,可以用另外一些算法求解网络流.

Type.6

当网络满足一些特殊性质的时候,可以用另外一些算法求解网络流.

一般而言,最大流贪心求解,最小割dp,费用流可以数据结构模拟(但是这部分题目基本都有清真做法,见林克卡特树),也可以贪心.

Type.6

当网络满足一些特殊性质的时候,可以用另外一些算法求解网络流.

一般而言,最大流贪心求解,最小割dp,费用流可以数据结构模拟(但是这部分题目基本都有清真做法,见林克卡特树),也可以贪心.

通常需要割流转换.

Goods transportation

n座城市,编号为1到n.

编号为i的城市生产出了P_i件商品,里面的人可以购买S_i件商品.

每个城市可以向每个编号比它大的城市运送C件商品.

求最多能卖出多少件商品.

$n \leq 10000$

Goods transportation

建图方式很明显,源点向每个点连P_i的边,每个点向汇点连S_i的边,并且向所有编号大于它的点连C的边.

Goods transportation

建图方式很明显,源点向每个点连P_i的边,每个点向汇点连S_i的边,并且向所有编号大于它的点连C的边.

最大流的复杂度无法接受,考虑求最小割.

Goods transportation

建图方式很明显,源点向每个点连P_i的边,每个点向汇点连S_i的边,并且向所有编号大于它的点连C的边.

最大流的复杂度无法接受,考虑求最小割.

设$dp[i][j]$表示前i个点中有j个在S集的最小割,转移方程显然.

网络流题优化建边的方式应该是最多的.

网络流题优化建边的方式应该是最多的.

序列上可以用线段树/ST表优化建边.

网络流题优化建边的方式应该是最多的.

序列上可以用线段树/ST表优化建边.

树上问题,如果连边和子树相关可以转化为序列问题,连边和路径相关可以考虑倍增/树链剖分/点分治(理论上来说点分治做法是最优秀的,但代码复杂度非常大,而且没有出题人会卡其他做法).

网络流题优化建边的方式应该是最多的.

序列上可以用线段树/ST表优化建边.

树上问题,如果连边和子树相关可以转化为序列问题,连边和路径相关可以考虑倍增/树链剖分/点分治(理论上来说点分治做法是最优秀的,但代码复杂度非常大,而且没有出题人会卡其他做法).

二维的问题可以考虑二维ST表/主席树/线段树合并.

网络流题优化建边的方式应该是最多的.

序列上可以用线段树/ST表优化建边.

树上问题,如果连边和子树相关可以转化为序列问题,连边和路径相关可以考虑倍增/树链剖分/点分治(理论上来说点分治做法是最优秀的,但代码复杂度非常大,而且没有出题人会卡其他做法).

二维的问题可以考虑二维ST表/主席树/线段树合并.

上面的几类优化方法可以混合使用.

流量限制通常在边上,如果对点有限制,可以把一个点拆成入点和出点(或更多的点),在中间连边.

流量限制通常在边上,如果对点有限制,可以把一个点拆成入点和出点(或更多的点),在中间连边.

贡献和时间相关的情况下考虑分层图/按时间或时间段拆点.

流量限制通常在边上,如果对点有限制,可以把一个点拆成入点和出点(或更多的点),在中间连边.

贡献和时间相关的情况下考虑分层图/按时间或时间段拆点.

在某些题目中可以通过拆汇点(表达不太严谨)避免分层图,见紧急疏散

流量限制通常在边上,如果对点有限制,可以把一个点拆成入点和出点(或更多的点),在中间连边.

贡献和时间相关的情况下考虑分层图/按时间或时间段拆点.

在某些题目中可以通过拆汇点(表达不太严谨)避免分层图,见紧急疏散

如果费用与流量之间的等量关系比较奇怪,考虑拆边.

题面里出现了网格/棋盘之类的东西,考虑黑白染色(或更复杂的染色).

题面里出现了网格/棋盘之类的东西,考虑黑白染色(或更复杂的染色).

如果需要多次修改网络,并且修改的是少数几条边的流量,考虑退流.

题面里出现了网格/棋盘之类的东西,考虑黑白染色(或更复杂的染色).

如果需要多次修改网络,并且修改的是少数几条边的流量,考虑退流.

如果发现存在大量冗余的边,但在初始化的时候并不能判断一条边是否有用,并且点之间分配流量的先后顺序是确定的,考虑动态建边(常见于费用流).

题面里出现了网格/棋盘之类的东西,考虑黑白染色(或更复杂的染色).

如果需要多次修改网络,并且修改的是少数几条边的流量,考虑退流.

如果发现存在大量冗余的边,但在初始化的时候并不能判断一条边是否有用,并且点之间分配流量的先后顺序是确定的,考虑动态建边(常见于费用流).

距离限制考虑切糕.

ExtremeSpanningTrees

给定n个点的带权图和两棵生成树.

将一条边的边权从U改为V的代价是$|U-V|$.

求将第一棵树变成最小生成树,第二棵树变成最大生成树的最小代价.

$n \leq 50$

ExtremeSpanningTrees

考虑一个更一般的问题,我们设旧边权为w,新边权为v,存在若干形如$v_i \leq v_j$的限制,最小化$\sum |w_i - v_i|$.

ExtremeSpanningTrees

考虑一个更一般的问题,我们设旧边权为w,新边权为v,存在若干形如$v_i \le v_j$的限制,最小化$\sum |w_i - v_i|$.

尝试去确定每条边最后的权值,考虑二分,对于一个x确定在最优解中哪些边的权值不小于x.

ExtremeSpanningTrees

考虑一个更一般的问题,我们设旧边权为w,新边权为v,存在若干形如$v_i \le v_j$的限制,最小化$\sum |w_i - v_i|$.

尝试去确定每条边最后的权值,考虑二分,对于一个x确定在最优解中哪些边的权值不小于x.

将每条边视为网络流上的点,向最终边权不小于它的边连边,如果该点满足$w_i < x$,那么它的权值为-1,否则为1.

ExtremeSpanningTrees

考虑一个更一般的问题,我们设旧边权为w,新边权为v,存在若干形如$v_i \le v_j$的限制,最小化$\sum |w_i - v_i|$.

尝试去确定每条边最后的权值,考虑二分,对于一个x确定在最优解中哪些边的权值不小于x.

将每条边视为网络流上的点,向最终边权不小于它的边连边,如果该点满足$w_i < x$,那么它的权值为-1,否则为1.

最后在最大权闭合子图里的点权值不小于x.

复杂度是做log次网络流的复杂度.

Thanks