网络流入门

-hzq84621

网络流是干什么的？

网络流可以解决一些很特殊的线性规划问题，以及各种极大极小问题，在OI竞赛中是一种很常见的模型。

线性规划是干什么的？

线性规划是指一类限制条件和目标函数都是线性的最优化问题。

举个例子，$2 \le 2*a+b \le 10$，$0 \le 2*b+c \le 11$，$3 \le 3*b+c \le 33$，求$\max(a+b+c)$

这就是一个线性规划问题

网络流相关定义

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)∉ 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。

容量限制（Capacity Constraints）：一条边实际的流量f(u,v)<=c(u,v)

斜对称（Skew Symmetry）：f(u,v)=f(v,u)

流守恒（Flow Conservation）：对于一个顶点u,除非u=s或u=t，$\sum_{w \in V} f(u,w) = 0$

来自维基百科

你说这个谁懂啊！

网络流模型

让我们用说人话的方法来看网络流模型

网络流模型

现在有一个排水沟渠系统，它有N个点，有M条沟渠，每条沟渠有一个流量k，意味着这条沟渠每秒只能流过k容量的水。

现在水要从点1流到点N，问每秒最多能流多少容量的水。

这就是最基础的网络流模型，不能更裸了。

网络流相关定义

然后我们回来定义一些东西。

为了防止歧义，我们把之前定义的边叫做弧

假设$F=\{i,j\in V|f(i,j)\}$表示所有弧的流量的集合
我们把它叫做一个这个网络的流

假如F满足容量限制，斜对称，流守恒，我们把F叫做一个该网络的可行流

对于一条源点到汇点的有向路径w，路径上和w方向相同的弧叫做前向弧
与w方向相反的弧叫做后向弧，前向弧集合记作A^+，后向弧集合记作A^-

网络流相关定义

不饱和弧：一条流量f(u,v)<c(u,v)的弧称为不饱和弧

非零流弧：一条流不等于0的弧称为非零流弧

可增广路：对于一个可行流F，它的一条有向路径w，假如它的每条前向弧都是不饱和弧，每条后向弧都是非零流弧，那么称w是一条F的可增广路

可增广路的意义是能够增大这条路径上的流量，使得最终的流量增加
这个操作称之为增广，增广的时候要把后向弧流量增加流量增加量，把前向弧流量减少流量增加量，这是为了流错了的时候可以再流回来。

残量网络：增广若干次之后当前的网络

网络流相关定义

最大流：一个S-T流量最大的可行流

最小割：指割去流量总和最小的弧后，使得S-T流量为0

最大流-最小割定理：一个网络的最大流流量等于其最小割

这个东西很好理解。最大流中的“瓶颈”弧流量的和就是最大流的流量，
我们只要把这些“瓶颈”弧割掉，就可以使得S-T流量为0

基本的定义到此结束，我们来看如何解决网络流模型的问题

怎么解决网络流模型的问题？

建模之后套模板

建模之后套模板

建模之后套模板

没错网络流你只要学会建模学会套模板背模板就行了
模板所有的优化也都是能背的

dinic

dinic算法是一种实际竞赛中性价比很高的算法，简单好写，效率O(n^2*m)

然而实际跑的时候会比O(n^2*m)要快很多，因为它复杂度不是满的。

我也没见过什么出题人出网络流题毒瘤到卡dinic

它也是我个人觉得最易于理解的网络流算法，比SAP、Relable_to_front什么不知道高到哪里去了

当然上面这些是什么东西我也不是很清楚，有兴趣的同学可以去nocow网络流词条学习一下。

dinic相关定义

层次图：根据源点到一个点要经过几条弧，可以把这个图分成若干层，这个图就叫做层次图。

dinic的具体操作就是：
1、用残量网络构出层次图
2、在层次图中DFS寻找增广路增广直至不存在该层次图的增广路
3、重复1、2直到无法构出层次图

dinic代码

```
bool build(){//构建层次图
    int x,y,l=1,r=1;
    memset(d,-1,sizeof(d));//表示点在第几层
    q[1]=S;d[S]=0;
    while(r>=l){
        x=q[l];
        for (int i=head[x];i;i=e[i].next){
            y=e[i].to;
            if (e[i].cap&&d[y]==-1){
                d[y]=d[x]+1;
                r++;
                q[r]=y;
                if (y==T)return 1;//若T已经DFS到说明层次图已构完
            }
        }
        l++;
    }
    return 0;
}
int find(int now,int flow){//增广, now表示当前点,flow表示当前流量
    int ret,y,w=0;
    if (now==T)return flow;
    for (int i=head[now];i&&w<flow;i=e[i].next){
        y=e[i].to;
        if (e[i].cap&&d[y]==d[now]+1&&(ret=find(y,min(flow-w,e[i].cap))))
            e[i].cap-=ret,e[e[i].opp].cap+=ret,w+=ret;//多路增广
    }
    if (!w)d[now]=-1;//多路增广
    return w;
}
int dinic(){
    int ans=0;
    while (build())
        while (1){
            int flow=find(S,oo);
            if (!flow)break;
            ans+=flow;
        }
    return ans;
}
```

这个模板还是蛮重要的
去打个20题就会背了

dinic优化

当前弧优化：假如对于一个有若干条弧的点有a的流量需要流出，假如它的前几条弧已经能够流出大于a的流量，那么后面的弧就不用再访问了，可以等下一次增广再去处理。多路增广思路与其基本相同

注意：对于某些网络，加入当前弧优化反而会变慢

16189047	2016-02-19 02:56:22	hzq84621	100733I - The Cool Monkeys	GNU C++	Accepted	108 ms	16300 KB
16189005	2016-02-19 02:50:35	hzq84621	100733I - The Cool Monkeys	GNU C++	Time limit exceeded on test 21	4000 ms	16300 KB

本人亲身经历

Edmonds-Karp

这个东西就是没有层次图处理的裸dinic(没有任何优化单路增广的）

思路就是BFS暴力找出一条可增广路，然后增广，直到找不出可增广路，时间复杂度O(n*m*m)

这个东西有什么用呢？可以用来做最小费用最大流

最小费用最大流

一个网络的最大流流量是唯一的，但是它的最大流是不唯一的

现在我们的每一条弧除了c(u,v)之外，还有一个权值C(u,v)表示在该弧中流过1流量的费用，求最小费用的最大流

最小费用最大流

我们从Edmonds-Karp开始考虑，因为最大流流量是唯一的，所以我们只需要考虑选择哪些费用较小的弧流就行了。

这其实就是个最短路

我们把BFS找增广路的过程换成以费用为权值的SPFA就可以了，这样能每次都找一条费用最小的弧增广，最后得到的答案就是最小费用最大流

最小费用最大流代码

```
void back(int i,int p){//退流
    e[i].cap-=p;e[e[i].opp].cap+=p;
}
int spfa(){
    for(int i=1;i<=n;i++)V[i]=oo;
    w[T]=0;V[S]=0;w[S]=oo;
    int l=1,r=1;q[1]=S;
    while(true){
        int x=q[l];
        for(int i=head[x];i;i=e[i].next){
            int y=e[i].to;
            if((e[i].cap)&&(V[y]>V[x]+e[i].coa)){
                pre[y]=x;//记录前驱节点和边（前驱节点可省去）
                pres[y]=i;
                V[y]=V[x]+e[i].coa;
                w[y]=min(w[x],e[i].cap);
                if(!b[y]){r++;if(r>n)r=1;q[r]=y;b[y]=true;}
            }
        }
        if(l==r)break;
        l++;if(l>n)l=1;
        b[x]=false;
    }
    if(w[T]==0)return 0;
    int k=T;
    while(k!=S){
        back(pres[k],w[T]);
        k=pre[k];
    }
    return w[T]*V[T];
}
int ek(){
    int ans=0,tmp;
    while((tmp=spfa()))ans+=tmp;//不断寻找增广路
    return ans;
}
```

假如有什么最大费用最大流，
把最短路变成最长路就可以了

网络流经典模型

网络流难点就在于建模，建出模型直接套模板就可以了

接下来会讲一些网络流的经典模型以及建模技巧（重点）

二分图匹配

其实很好理解，就是把源点往一边的点连流量为无限的弧，然后另一边的点向汇点连流量为无限的弧，然后中间连流量为1的边跑最大流就可以了

dinic跑二分图匹配的复杂度是O(sqrt(n)*m)所以说匈牙利几乎没有什么卵用

一定要说的话匈牙利的结构有可扩展性，网络流几乎没有，比如有些题目依赖性质最后得到的算法可以用线段树等数据结构维护匈牙利的过程。
其它什么二分图最大权匹配也可以用最小费用最大流解决（然而速度会被艹爆）

二分图匹配

现在有一个n*m的棋盘，每次操作可以将相邻两个格子上的数字都+1，问至少几次操作能使得所有格子上数字都相同。不可能的话输出-1，n,m<=40

考虑把格子按照i+j的奇偶性分别染成黑色和白色，那么每次操作不会影响黑色格子与白色格子内数字和的差

设最后格子中数字都变成k,黑格子一共有num1个，初始数字和为sum1
白格子一共有num1个，初始数字和为sum2，有k=(sum1–sum2)/(num1 – num2)

当num1!=num2的时候可以把k解出来然后跑二分图验证

当num1=num2时因为可以每次铺满一层，所以k具有单调性，可以二分之后跑二分图验证

二分图匹配

建图方法：

对于一个黑格子p，从源点向它连一条流量为x-v[p]的边

对于一个白格子q，从它向汇点连一条流量为x-v[q]的边

相邻的格子直接连一条容量为无限的边

每次检验是不是满流即可。

拆点

有的时候限制不在边上，在点上，怎么办呢？

把一个点拆成两个点，然后中间连一条容量为限制的边就可以了

拆点

现在你有两棵树，每棵树上有若干个高度不同的树枝。每天早上有m只猴子会在任意一棵树的最高的m个树枝上，然后跳到另一颗树上的一个树枝上，重复这个过程直到这m只猴子都到达与初始的树不同的树的m个最低的树枝上。

因为这些猴子非常傲娇所以不会跳到其它猴子跳过的点。

猴子跳跃能力有限所以每次跳跃两个树枝高度差不能超过k。

$m \le n_a, n_b \le 500$

拆点

把每个树枝拆成两个点，流量为1，然后在两个高度差不超过k的不同树上的树枝之间连流量为无限大的弧，然后源点向第一棵树上最高的m个点连一条流量为1的弧，再向第二棵树上最低的m个点连一条流量为1的弧，检验是否为满流即可。

因为不限制初始树所以要第一棵树和第二棵树交换一下再做一遍

不等式相关

上次差分约束不知道费老板有没有给你们讲，没有讲的话你们去问他

很多不等式问题都可以用网络流解决（虽然效率可能比差分约束差很多）
适用性比差分约束广(虽然一般这种问题能用网络流解决都可以用单纯形解决）

不等式相关

题意：现在有n天m类志愿者，第i天至少需要a_i个人工作，第j类人可以从第l_i天工作到第r_i天，费用为c_i,求满足要求的最少费用

绝对经典的线性规划问题网络流解法。我们先假设x_i表示第i类志愿者招募几个显然我们可以列出n个不等式表示每一天的情况，再使用n个费用为0的辅助变量，使得不等式变为所有变量大于0的等式。通过移项使得等式右边都为0

我们把每个等式都作为一个点，假如它的常数项c_i是正数，就从源点到它连一条容量为c_i费用为0的弧，否则从它到汇点留一条容量为$-c_i$费用为0的弧

对于一个变量x_i假如它在第j个等式中系数是1，第k个等式中系数是-1，那么从点j到点k连一条容量为INF费用为v_i的弧

不等式相关

然后跑费用流即可得出结果。考虑为什么这样是对的

等式中的左右相等和网络流的流守恒意义很相近，所以每一个点的流守恒就意味等式是成立的，一条弧流入或流出的流量就意味着某个变量对于等式的贡献（也就是这个变量的值）

而我们要求的目标函数就是变量的值再乘以对应的权值，也就是费用流模型

那么从这个建模思路我们可以发现，网络流只能解决变量系数为1和-1还有0的不等式问题

最大权闭合子图

最大权闭合子图问题是指对于一个有向图，每个点有权值（可正可负）我们要选择一些点使得其权值和最大，它们之间有依赖关系，即若点i向点j有一条边说明选择点i必须选择点j

对于每个权值为负的点从源点向它连流量为权值绝对值的弧，否则从它向汇点连一条流量为权值的弧
存在依赖关系的点之间连流量为INF的弧，总收益-最大流即为解

为什么这样是对的？

考虑最小割模型，若割掉源点到权值为负的点的弧，说明我们需要选择这个点
若割掉权值为正的点到汇点的点，说明我们不选择这个点

最大权闭合子图

题意：现在有n个通讯中转站，建设第i个的费用为P_i，还有m个用户，第i个用户表示假如你建设了第A_i个和第B_i个中转站可以得到C_i的收益。求最大净收益。

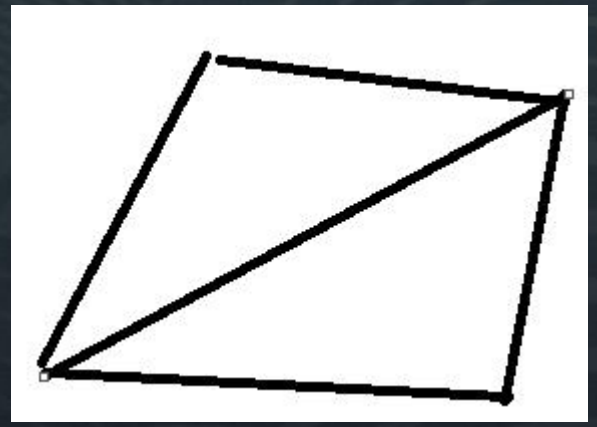
裸的最大权闭合子图，每个用户从A_i和B_i向它连边

最大权闭合子图可以解决很多最大盈利问题，也是很经典的模型（一般都是裸题）

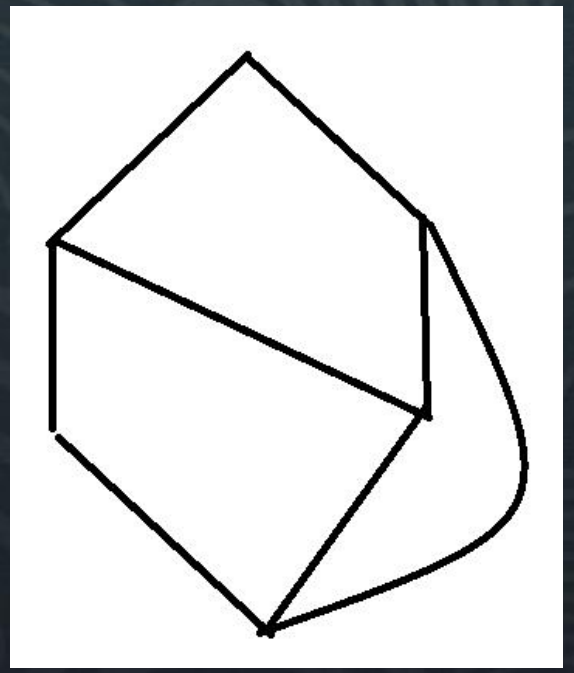
R爷说这个东西有O(点数)做法

平面图在网络流上的应用

平面图是指对于某个图的构造，我们可以在平面上把它画出来而且满足边与边不相交。



这样的

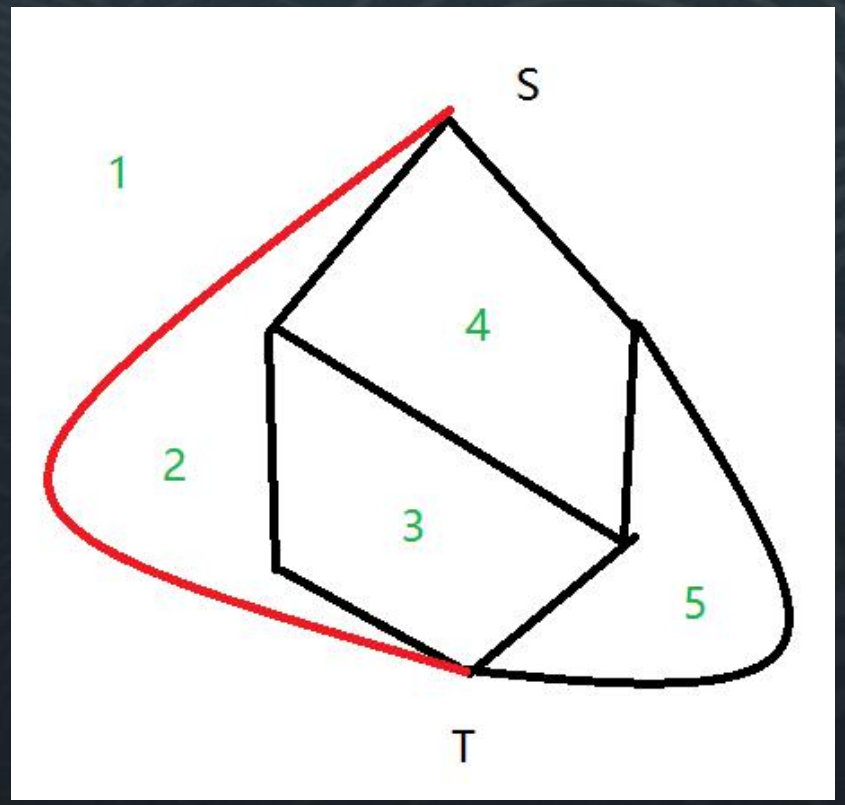


这样的

都是平面图

平面图在网络流中的应用

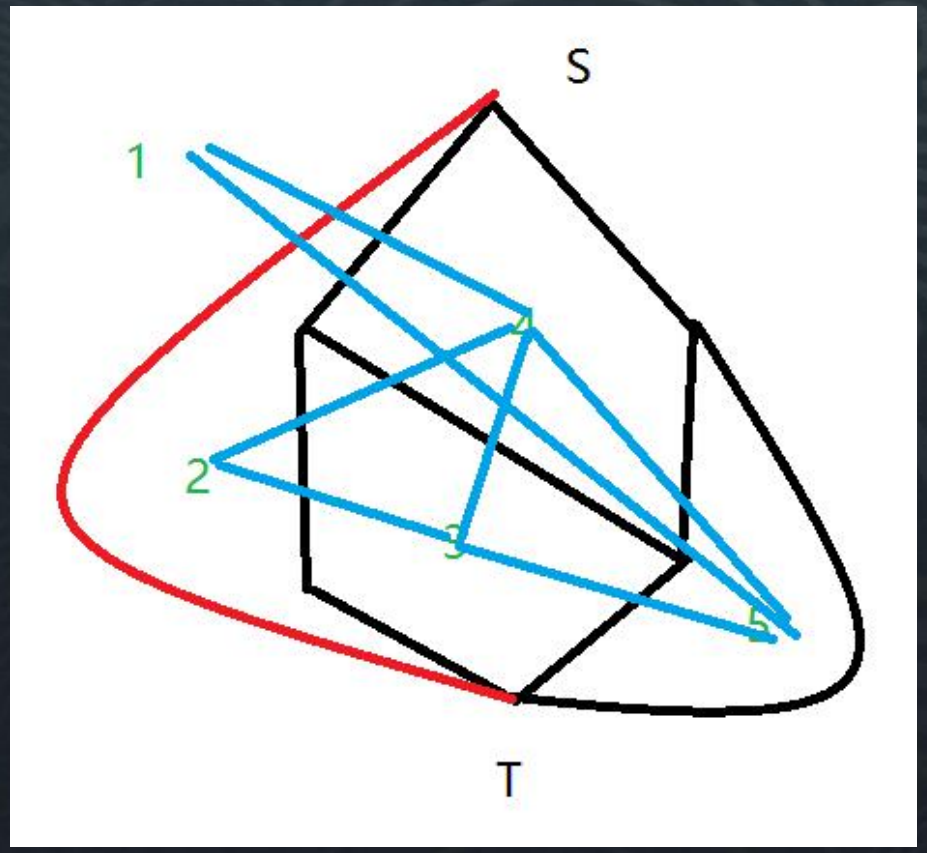
对于一个网络，假如说它是一个平面图，我们可以在求最短路的时间内把它的最小割求出。我们以之前的图为例



我们在图中原有的边的基础上从S到T连一条辅助边
然后给每个区域标号

平面图在网络流中的应用

然后对于每个区域我们把它当做一个点，对于原图的每条边在它连接的两个区域之间连边，权值为它的流量（注意辅助边不用连）



然后在这个图上跑最外层区域到辅助边分出的区域的最短路既是最小割（即图中1到2）

这只是平面图的一个比较特殊的应用。其它各种应用加起来模板8K+，有兴趣的同学可以去了解一下。

GL&HF