CS107

网络流建模

基本概念复习

流量网络

→ **有向正权图G=(V,E,C)，C称为容量**

两个特殊点：源$s\in V$，汇$t\in V$

→ **可行流F**：E上一组权值，满足

容量约束：$0\le f(u,v)\le c(u,v)$

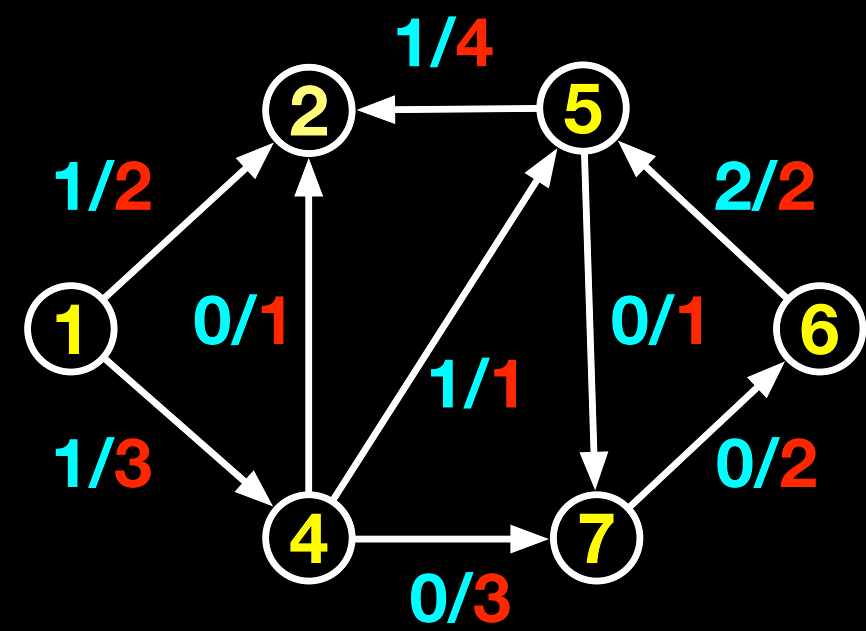
流量恒律：$\sum f(x,*)=\sum f(*,x)$，$\forall x\neq s,t$

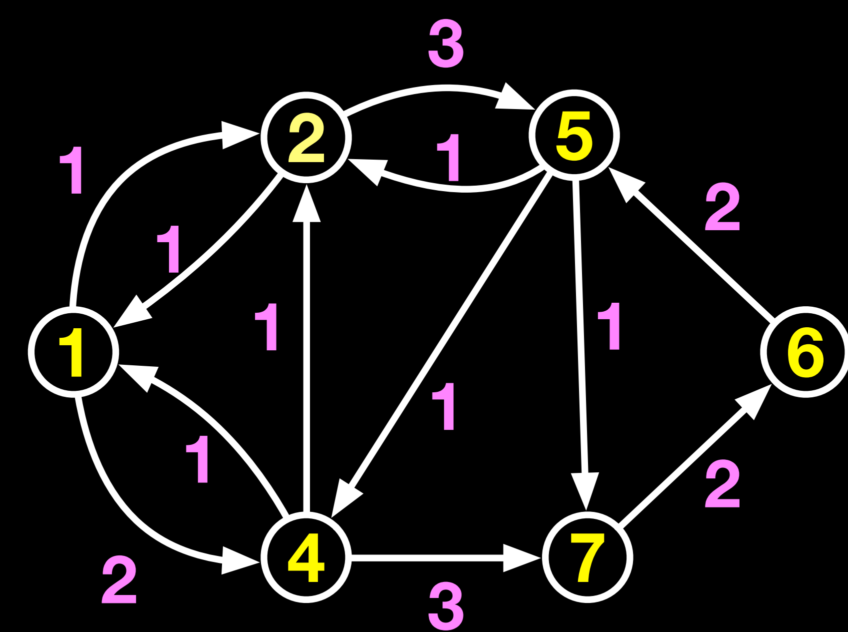
总流量：$|F|=\sum f(s,*)=\sum f(*,t)$

→ **残量网络R=C-F（正向流量可放大多少）**

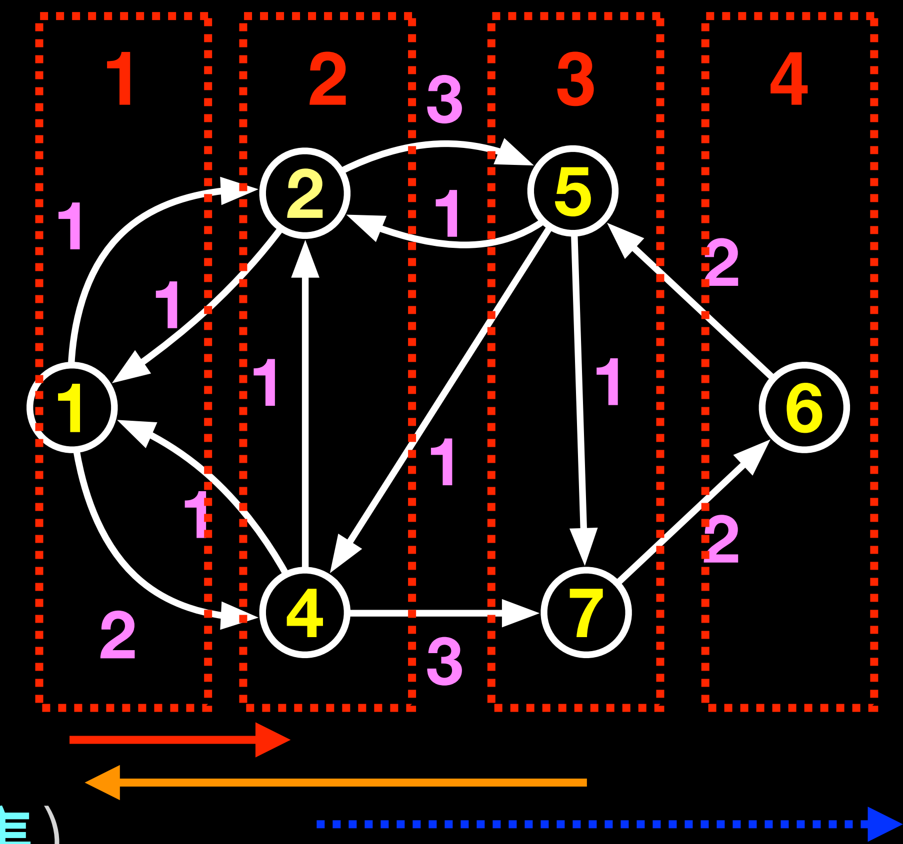
$r(u,v)=c(u,v)-f(u,v)$，$r(v,u)=f(u,v)$

R上的通路P称为**增广路**，增广量为$\min\{r(u,v) \mid (u,v)\in P\}$





最大流问题



→ **调整法**：每次找增广路增广，Over

→ **正确性（无极优解）**

X=s→t路径集合（可走原图逆向边）

L=可行流集合（X中符合容量约束的**子集**）

M=(X,L)是**拟阵**。Over

（差量网络是R的可行流，可证明交换性）

→ **EK算法**：**BFS**找增广路（边数最少）

复杂度：V按BFS步数分类**（分层图L）**，无前向边

每增广1次，L至少删1条边（至多m次L不通）

每次L不连通，重新BFS至少加1层（至多n次）

流量网络建模的一些哲学套路

→ 需要巧妙设计各元素的含义以刻画题目的对象/约束

→ S的邻边可表示某种“资源”

→ T的邻边可表示某种“收益”

→ 单位容量的边可起到隔离作用（“互斥”条件）

→ 单位容量的边可起到作用（“互斥”条件）

→ 单位流量可表示一种方案或一组有关联的对象

→ 流量平衡可表示某种守恒关系（废话）

最大流建模

Dining

POJ 3281（复习）

N头牛、F种食物、D种饮料。**每种食物/饮料只够一头牛吃喝，每头牛有喜欢的食物/饮料种类列表，并只选1种食物/饮料**。求最多能使几头牛同时吃喝到喜欢的食物和饮料（F,D,N≤100）

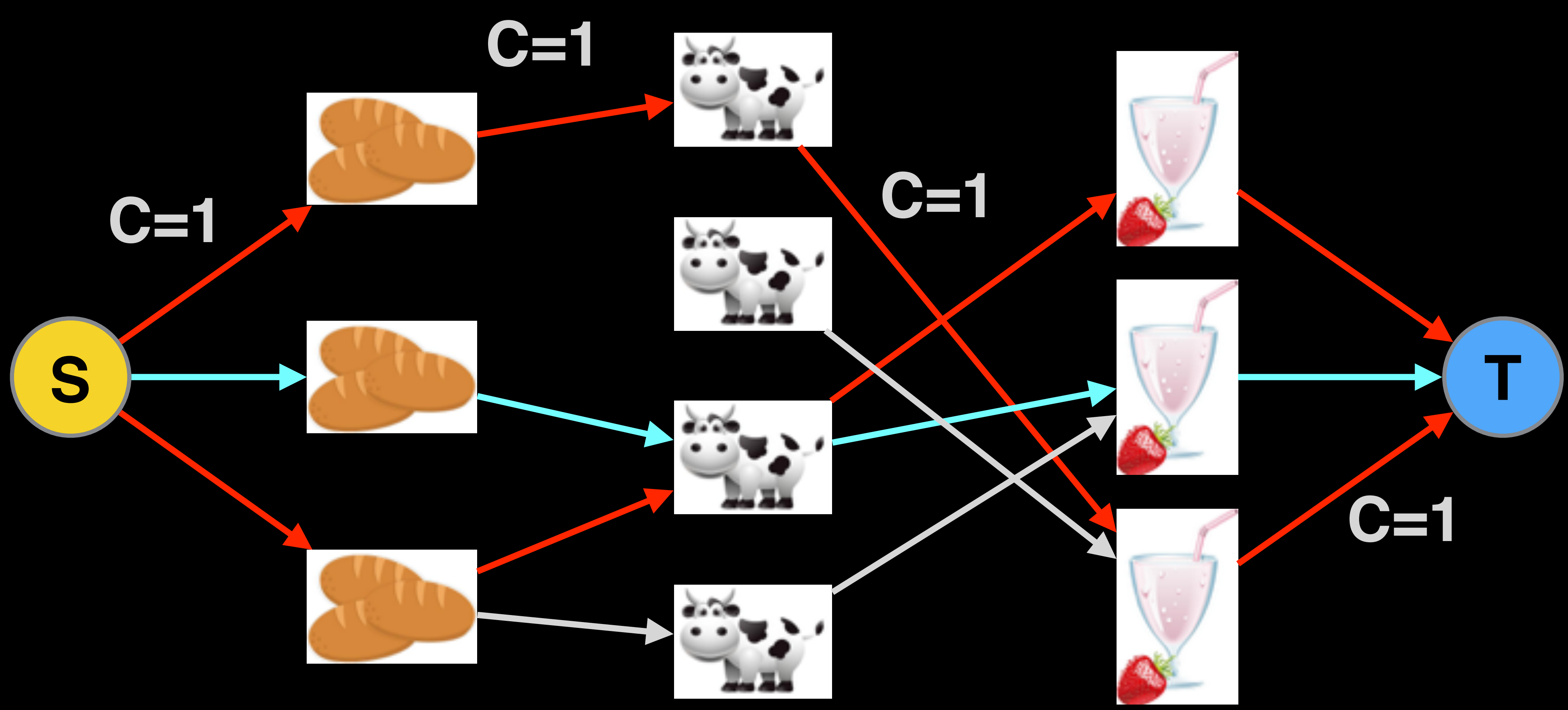
建模

→ 单位流量可表示有关联的一组对象（牛+吃+喝）

单位容量的边可起到隔离作用（事物/饮料都是1牛份）

→ 问题：未限制牛只吃喝1牛份

拆点为边（限流边），再利用单位容量边的隔离作用



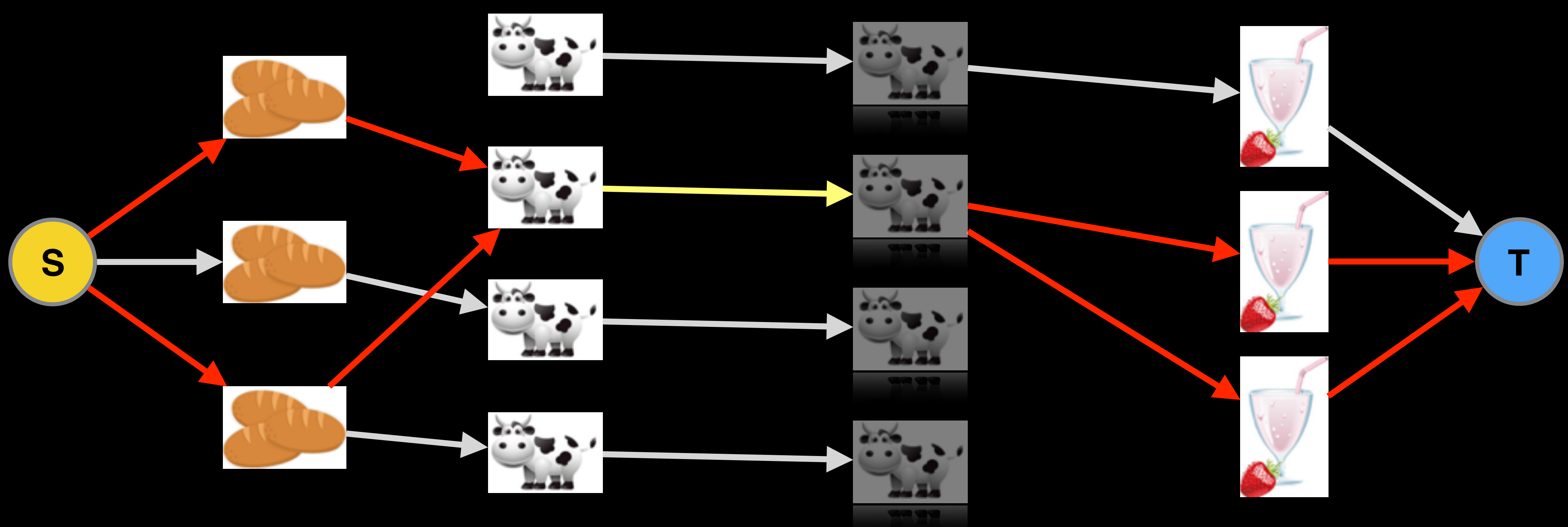
建模

→ **问题**：未限制牛只吃喝1牛份

拆点为边，再利用单位容量边即可**容量为1（限流边）**

→ **其他类问题（单位流量连接有关联的一组对象）**

疫苗(vaccine)、铁道修建(railway)。2019寒C206



金神龙的修炼

→ **最优化问题**

n个集合（灵石），可**交换集合1与其他集合的1对元素**

决策对象： 交换灵石的策略（顺序+方案）

约束条件： **只能用对方没有的换对方多余的**

优化目标： 集合1的UV最大

建模（最大流流量网络是怎么炼成的）

→ **持有的各灵石视为“资源”**

S对每种灵石建边，容量为初始持有量

→ **有了资源怎么用?**

1.贡献到最终的收益

每种灵石Ai对T建边，容量为1

（同类型至多贡献1收益）

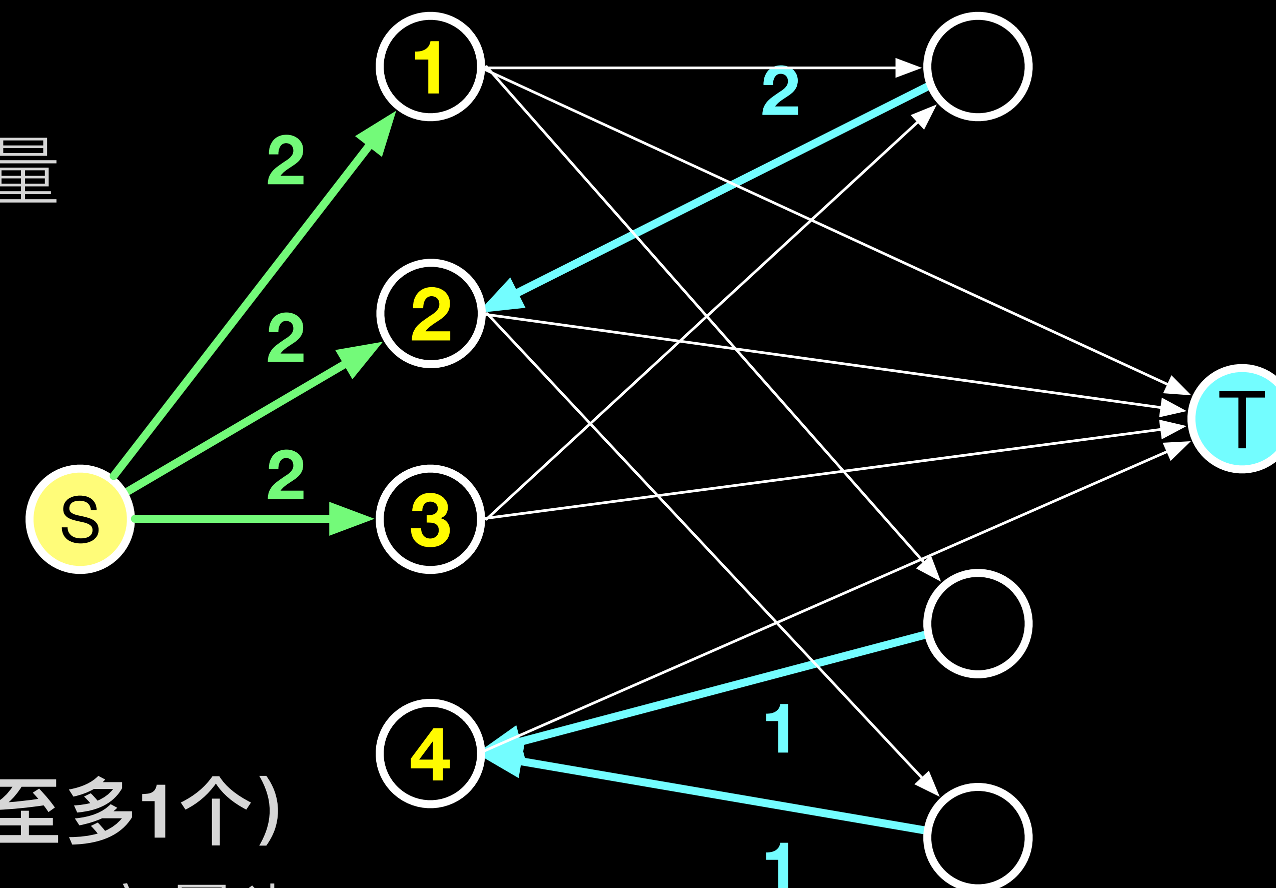
2.去祭坛换（从灵石到祭坛建边）

→ **换出（祭坛只接收没有的种类，且至多1个）**

从祭坛**没有的所有灵石种类**向其建边，容量为1

→ **换入（祭坛只换出多余的灵石）**

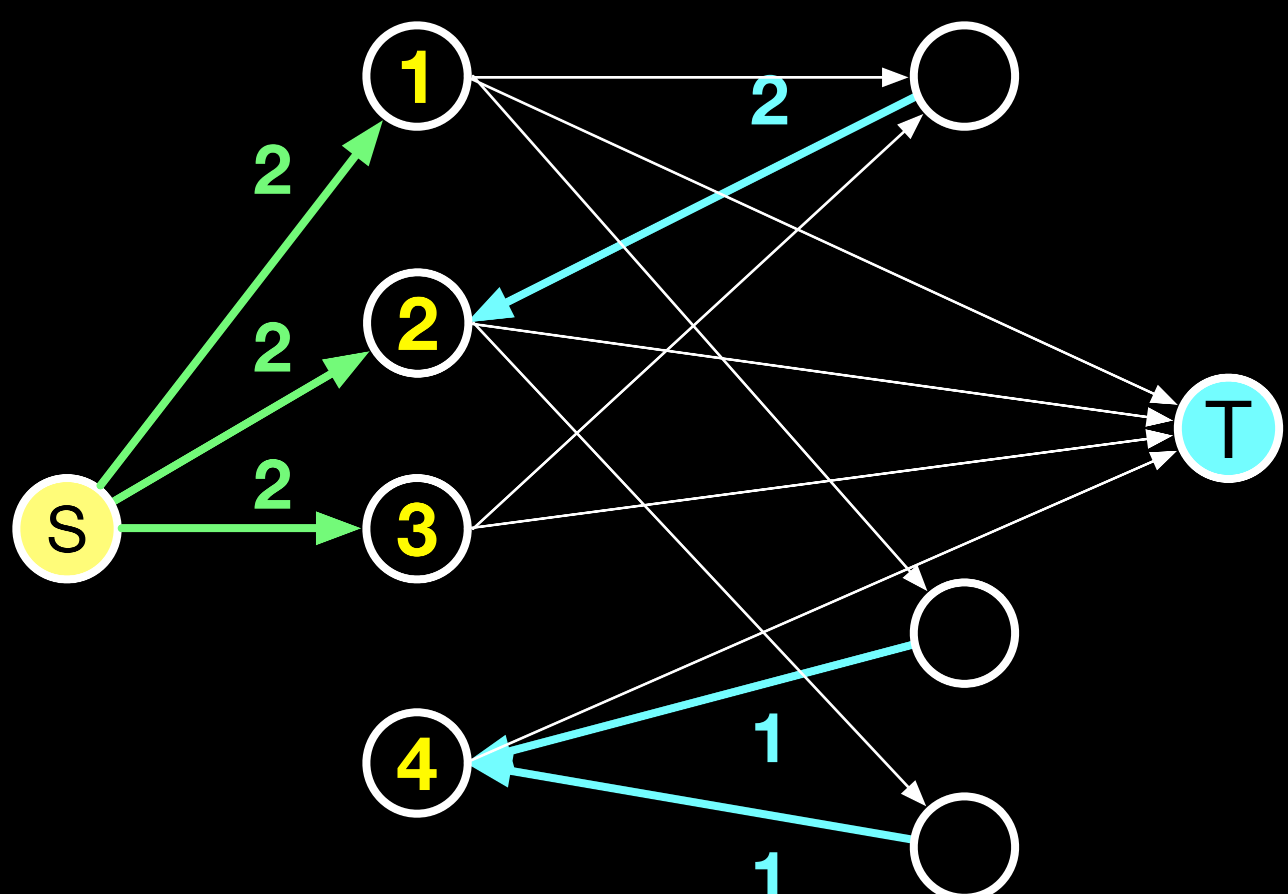
从祭坛**持有>1个的灵石种类**向其建边，容量为**持有量-1**

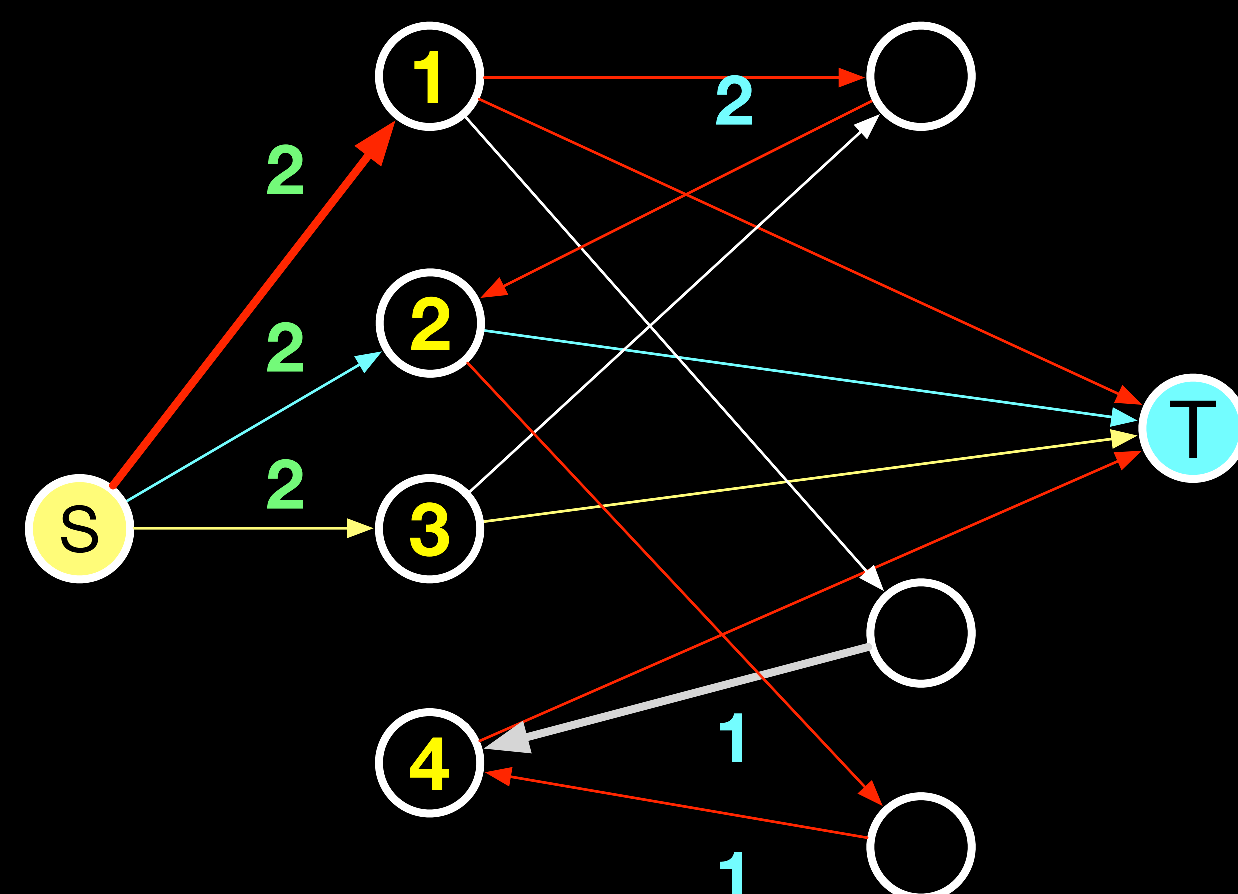


演示

→ **每个最终有贡献的灵石（单位流量）的生命周期**

S→A→B→…→A→B→T（AB重复0~n次不等）





我造数据的时候画的图

→ **防止简单的贪心法骗分**

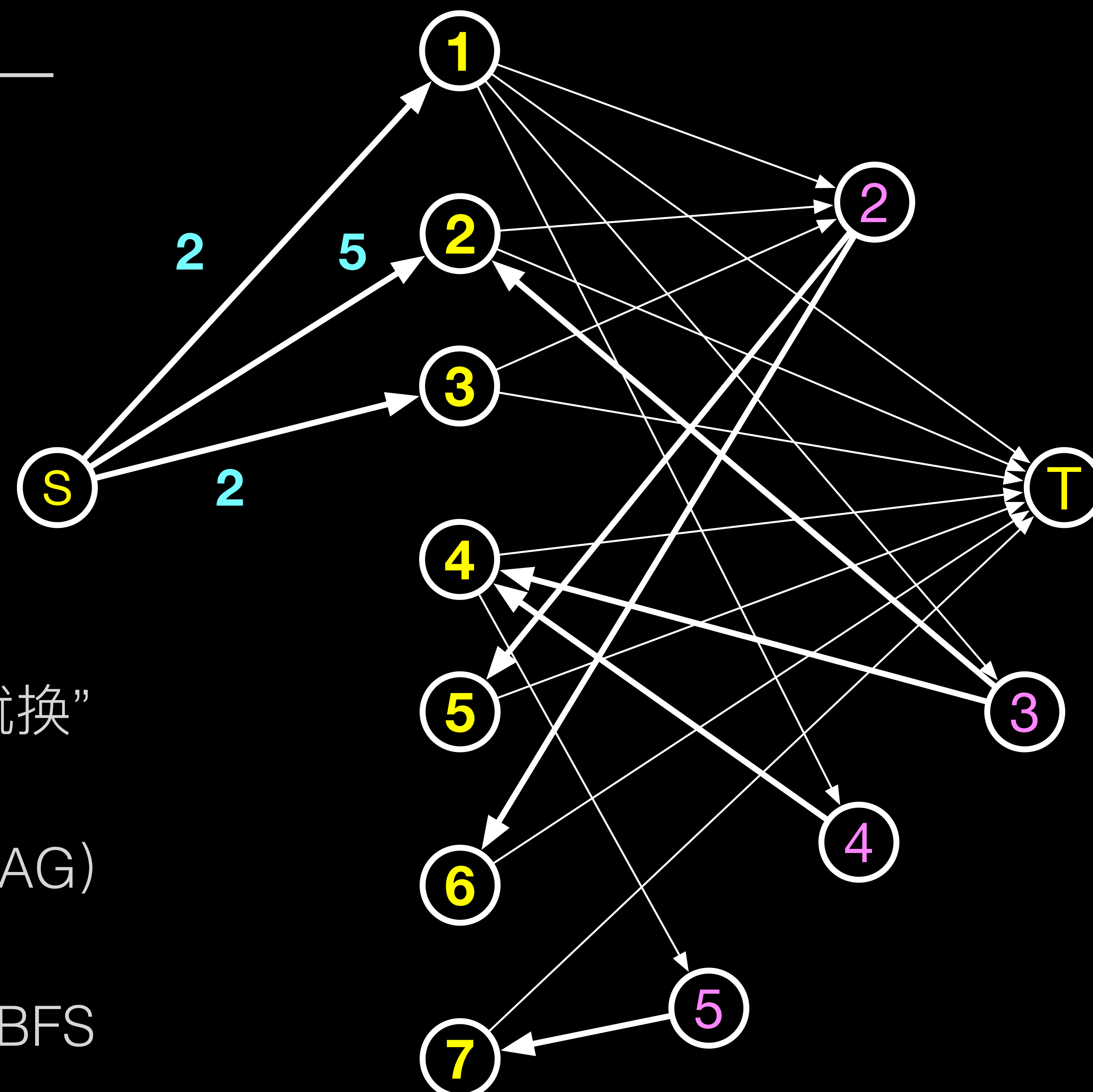
如“发现能换到我没有的，就换”

→ **其他想法**

祭坛UV只增不减（状态是DAG）

而数据规模总体不很大

可对祭坛所有状态**状压**，再BFS



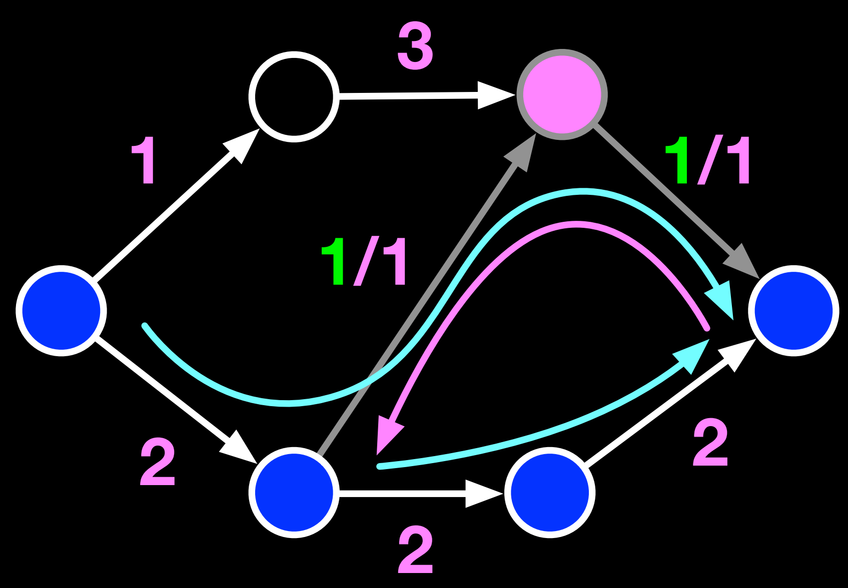
Dinic算法复习

→ 如分层图还连通，不需要重新BFS

→ Dinic增广（分层图L上DFS）

回溯时增广，**合并**后向的多条增广路

单节点后向不能增广，直接从L**删除**



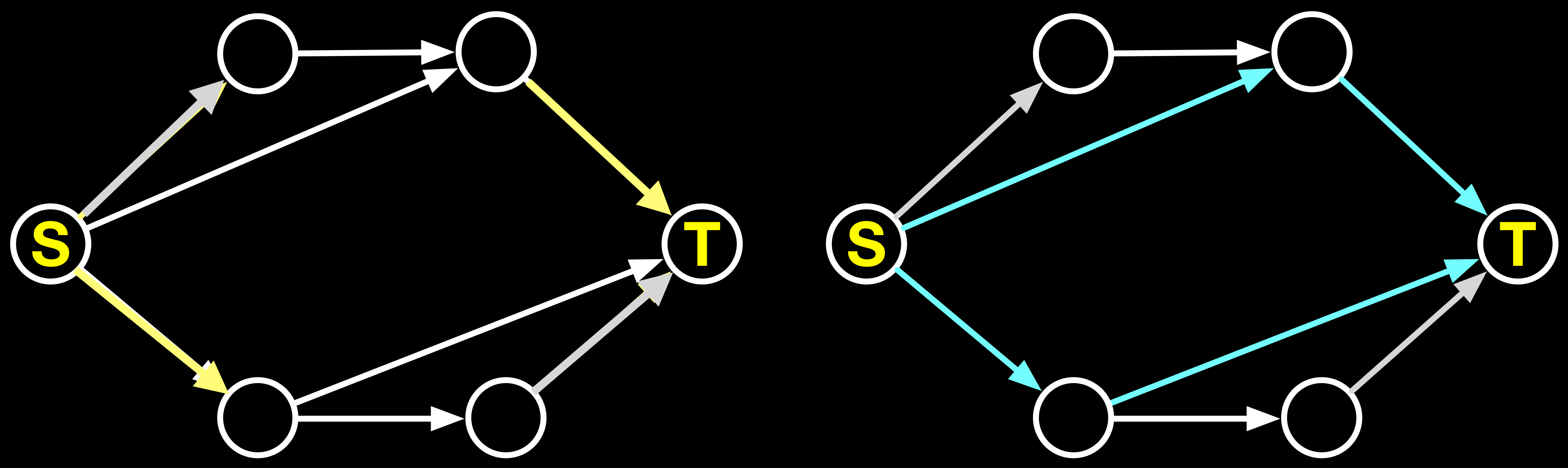

```
int dinic(int x,int flow){
    if (x==n) return flow;
    int rest=flow,k;
    for (int i=hd[x];i && rest;i=es[i].nxt)
        if (es[i].c && d[es[i].t]==d[x]+1){
            k=dinic(es[i].t,min(rest,es[i].c));
            if (!k) d[es[i].t]=0;
            es[i].c-=k,es[i^1].c+=k,rest-=k;
        }
    return flow-rest;
}
```

最小割建模

复习

- **割集：** 删除后使s→t不连通的边集
- **最大流量最小割定理**

min{割集权和}=max{可行流量}



割集建模概述（哲学）

→ **首先，最小割建模是图论问题**

只是恰好可用最大流求解（所以一般不必考虑流量的事情）

→ **通路S→X→T可表示一种冲突关系**（割集必含S→X或X→T）

通路S→X1→X2→...Xk→T可表示一种单选关系

特别的，如w(Xi,Xi+1)=∞，i=1..k-1

则可表示2条已有边的冲突关系（最小割必包含S→X1或Xk→T）

龙族动物园

→ **最优化问题**

决策对象： n*m的矩阵填01

约束条件： 无

优化目标： 蛋疼的总收益最大

1.每个位置填0/1各自有单格收益(Ai,Bi)

2.相邻位置同时天0/1有额外双格收益(Cij,Dij)

建模（最小割图是怎么炼成的）

1	2
4	3

→ **先补集转化**

最大收益=所有收益-最小舍弃(割掉)多少收益

→ **2条连续边可表示一种冲突关系**

→ **Ai与Bi的冲突**

S到位置i建边，权为Ai（表示i放0）

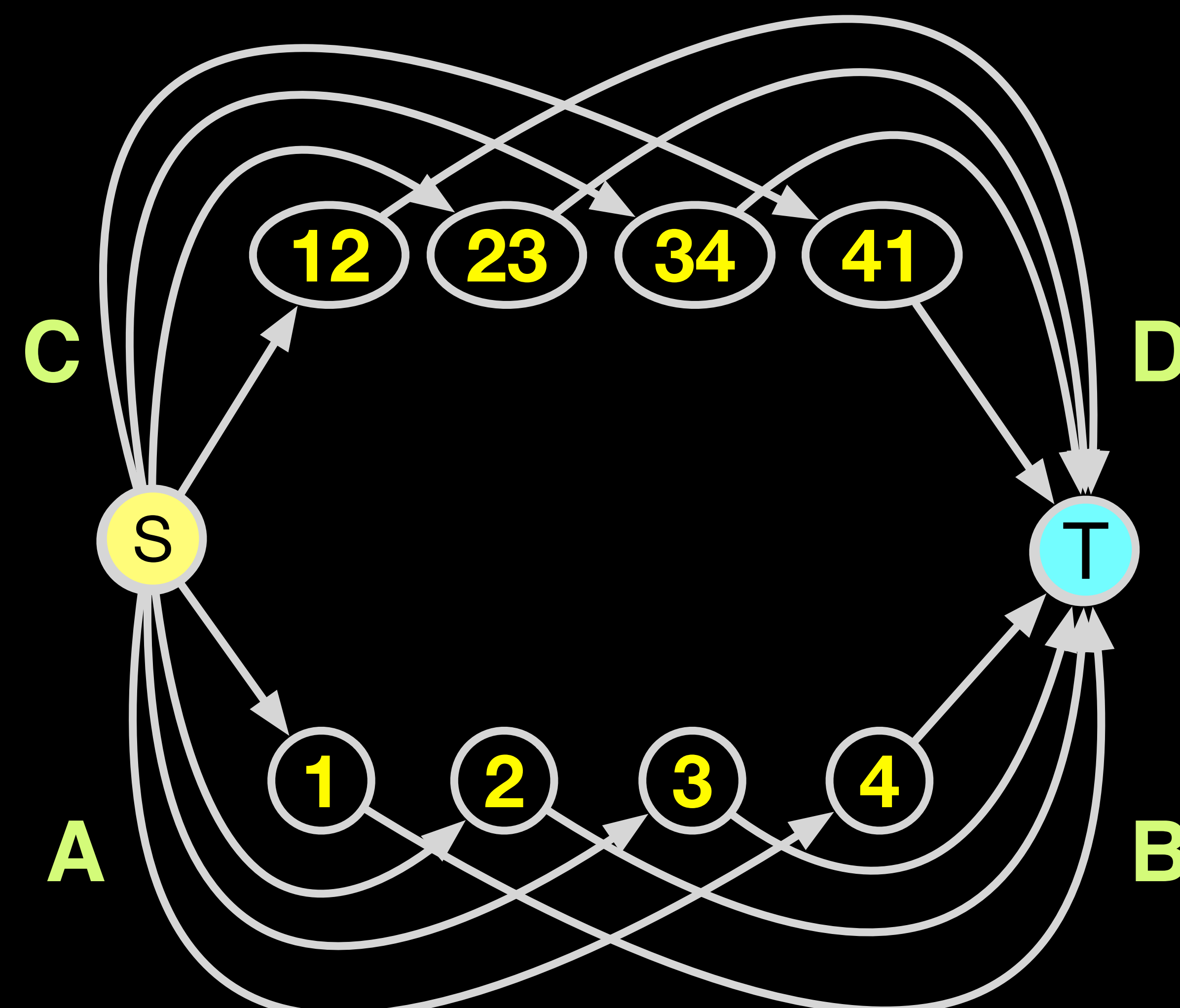
位置i到T建边，权为Bi（表示i放1）

S→i→T表示i不能同时填0/1

→ **Cij与Dij冲突**

S到邻点对ij建边，权为Cij

邻点对ij到T建边，权为Dij



建模（最小割图是怎么炼成的）

0	0
0	1

1	2
4	3

→ **Ai与Dij/Dji冲突，Bi与Cij/Cji冲突**

无穷大权边可限制**现有边**的冲突关系

i→ij/ji建边，ij/ji→j建边（双向边），权为**INF**

S→i→ij→T表示Ai与Dij冲突

S→ij→j→T表示Cij与Dj冲突

→ **右上方案弃边方案如右下图**

纳尼？不是割集？…囧

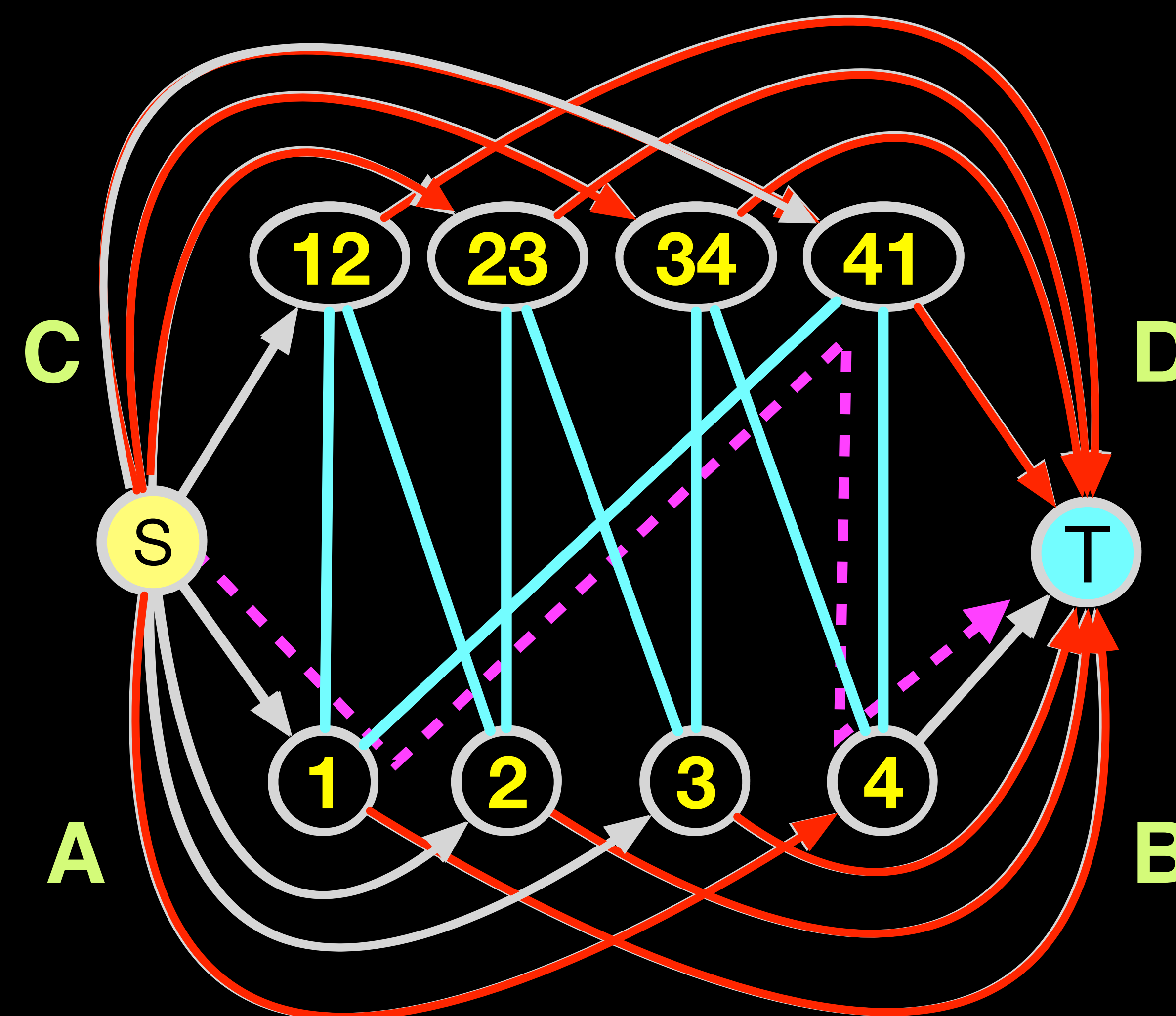
→ **由于冲突路径之间的重叠**

产生了新的（并不成立的冲突）

如“A1和B4冲突”

→ **事实上由于无向边的影响**

除S,T其他点已经成为SCC了（大囧



修正

$$y = \frac{1}{|x| - 1}$$

囧函数

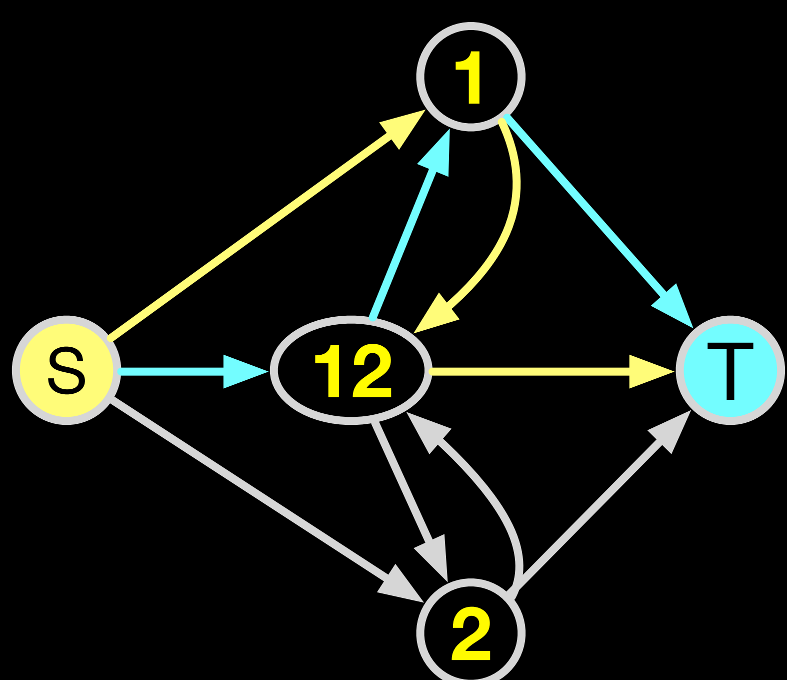
→ **产生额外的冲突的原因：**串路径

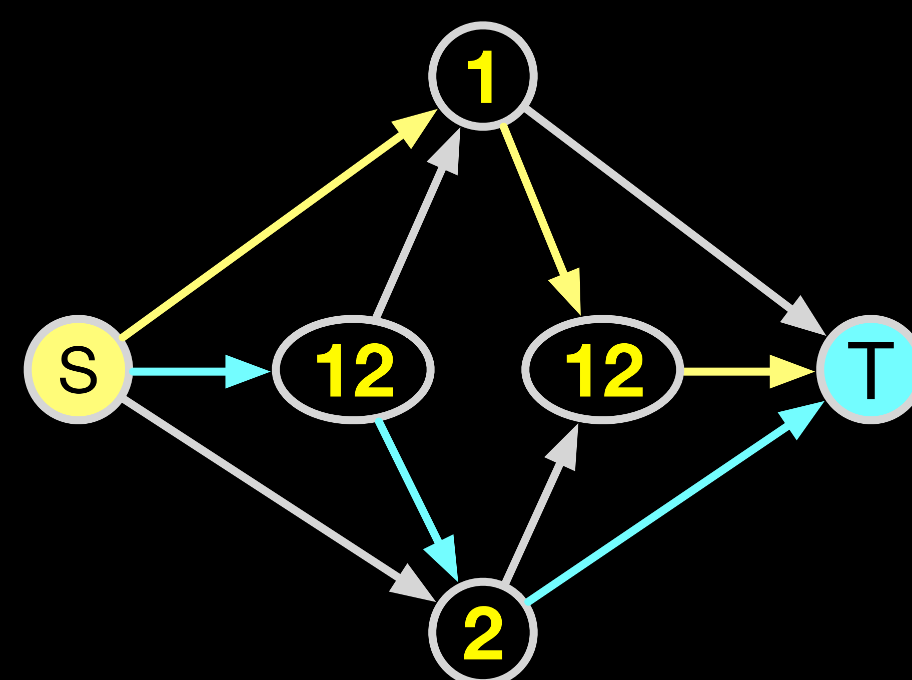
构造路径S→1→12→T和S→12→1→T

由于**12是一个点**，产生了额外路径S→2→12→2→T

→ **将12拆2个点，把两条路径隔离开**

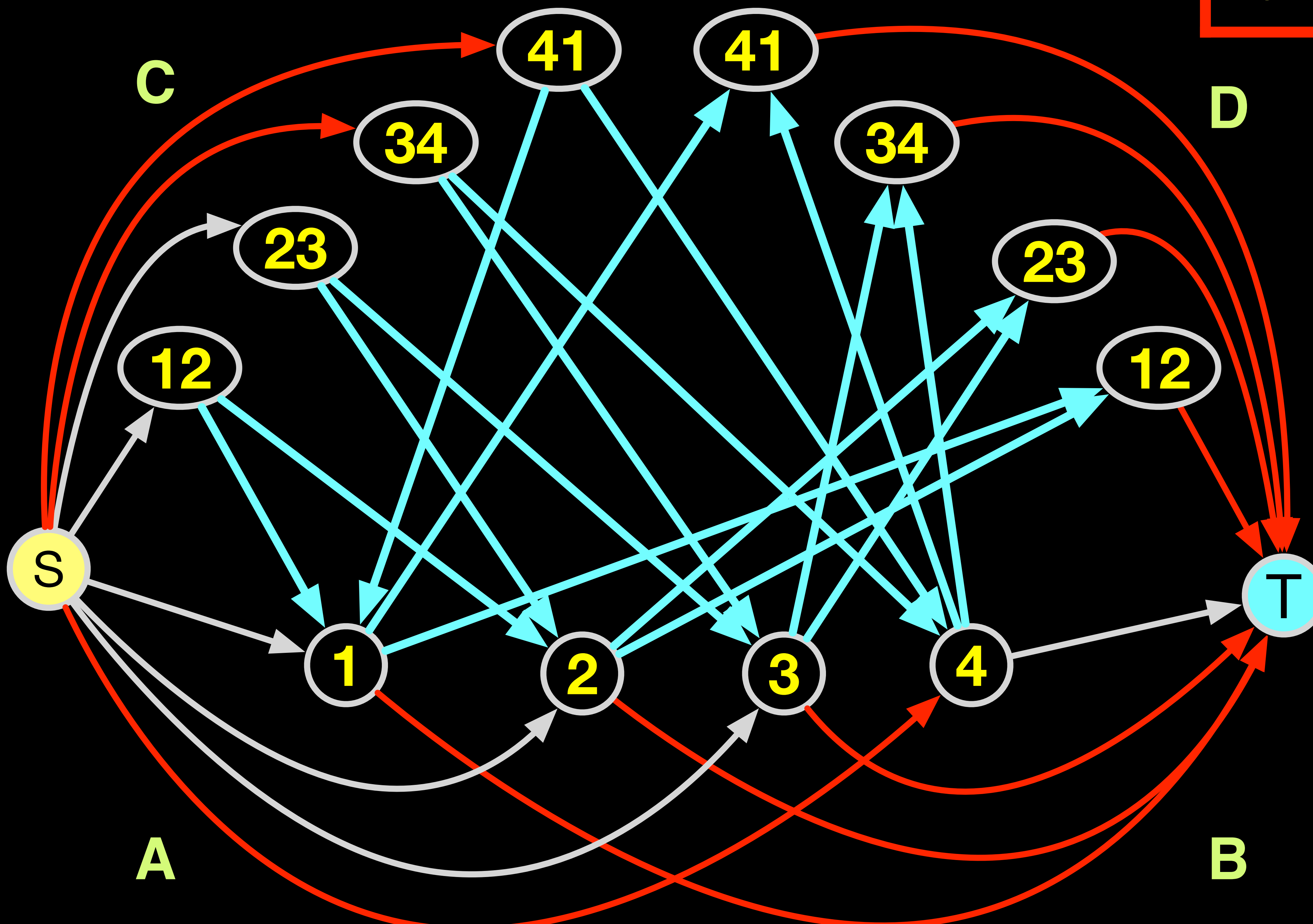
两个新点的实际意义为“12都填0/1”





方案示例（有点复杂）

0	0
0	1



网络流题目代码画风

设计节点编号

```
scanf("%d%d",&n,&m);
int s=0,t=n*m+2*n*(m-1)+2*(n-1)*m+1,cnt=n*m,ans=0;
for (int i=1,x;i<=n;++i) for (int j=1;j<=m;++j)
        scanf("%d",&x),ans+=x,addedge(s,id(i,j),x);
for (int i=1,x;i<=n;++i) for (int j=1;j<=m;++j)
        scanf("%d",&x),ans+=x,addedge(id(i,j),t,x);
for (int i=1,x;i<=n-1;++i) for (int j=1;j<=m;++j){
    scanf("%d",&x),ans+=x;
    addedge(s,++cnt,x);
    addedge(cnt,id(i,j),inf);
    addedge(cnt,id(i+1,j),inf);
}
for (int i=1,x;i<=n-1;++i) for (int j=1;j<=m;++j){
    scanf("%d",&x),ans+=x;
    addedge(++cnt,t,x);
    addedge(id(i,j),cnt,inf);
    addedge(id(i+1,j),cnt,inf);
}
for (int i=1,x;i<=n;++i) for (int j=1;j<=m-1;++j){
    scanf("%d",&x),ans+=x;
    addedge(s,++cnt,x);
    addedge(cnt,id(i,j),inf);
    addedge(cnt,id(i,j+1),inf);
}
for (int i=1,x;i<=n;++i) for (int j=1;j<=m-1;++j){
    scanf("%d",&x),ans+=x;
    addedge(++cnt,t,x);
    addedge(id(i,j),cnt,inf);
    addedge(id(i,j+1),cnt,inf);
}
printf("%d\n",ans-dinic(s,t));
```

图规模：
|V|~5nm, |E|~6nm

建图

建图

建图

建图

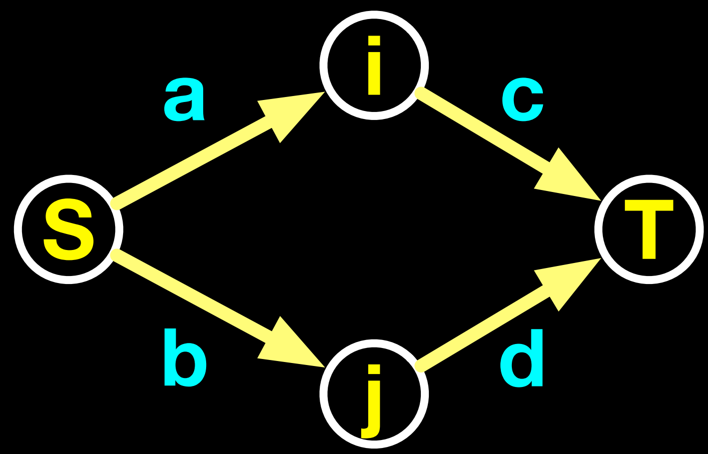
建图

建图

Dinic

另一种建法（二元联合关系）

→ 局部收益涉及2个对象，每个有2种选择S/T
割集中，(a,c)/(b,d)确保至多选1



→ 共4种割法，对应损失如下

i选1，j选1（割ab）：a+b=Ai+Aj+Cij

i选0，j选0（割cd）：c+d=Bi+Bj+Dij

i选1，j选0（割ad）：a+d=Ai+Bj+Cij+Dij

i选0，j选1（割bc）：b+c=Bi+Aj+Cij+Dij

→ 解出a,b,c,d😏

③-①：d-b=Bj-Aj+Dij

②-④：d-b=Bj-Aj-Cij

Cij=-Dij，矛盾o(╯□╰)o

另一种建法（二元联合关系）

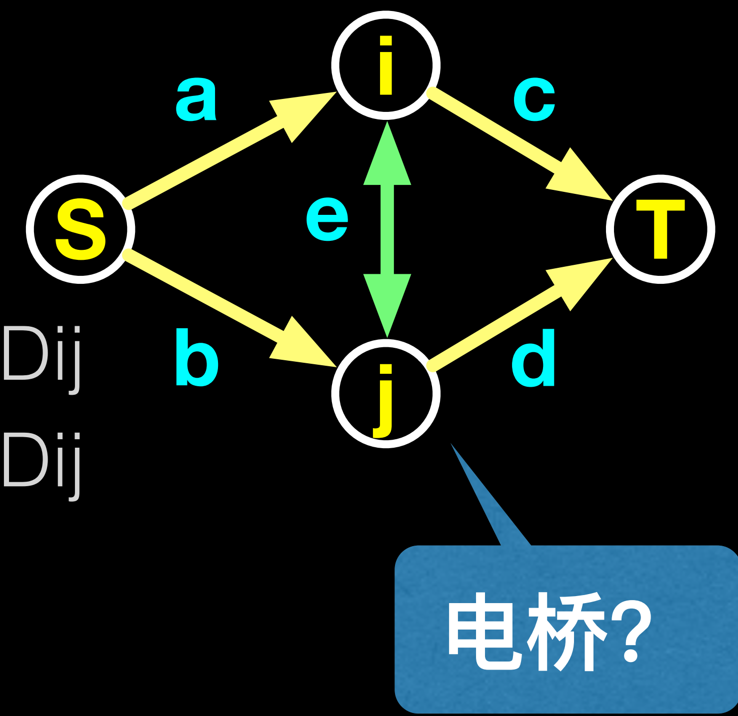
→ **自由度不够，需要额外变量调节**

i选1，j选1（割ab）：$a+b=Bi+Aj+Cij$

i选0，j选0（割cd）：$c+d=Bi+Bj+Dij$

i选1，j选0（割ade）：$a+d+e=Ai+Bj+Cij+Dij$

i选0，j选1（割bce）：$b+c+e=Bi+Aj+Cij+Dij$



→ **解（不定）方程组**

③+④-①-②：$e=(Cij+Dij)/2$

还剩4个变量，3个方程，求一组**特解**即可

$a=Bi+Dij/2$，$b=Bj+Dij/2$，$c=Ai+Cij/2$，$d=Aj+Cij/2$

→ **对多组ij，只要对j做一下叠加即可**

$ai=Bi+sum\{Dij/2\}$, j=i的邻点，b,c,d类似

$eij=(Cij+Dij)/2$（e代表**ij不相同，需要额外割一下eij**）

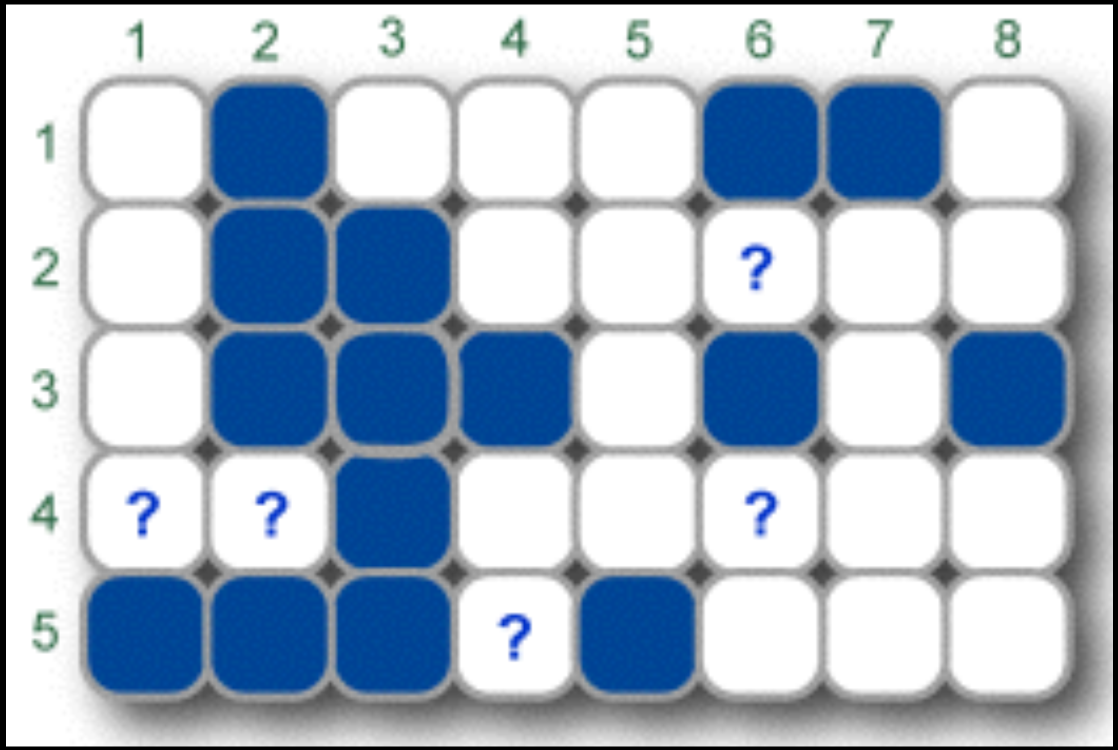
→ **这个电桥图是处理多组2元联合关系类问题的经典单元**

思考题：该图为何不会出现多余限制?

日程表

Google Code Jam 2008 Final E

n*m矩阵染黑白，有些位置已固定，有些位置待决策。收益为**4*黑格总数-2*相邻黑格对数**，求最大收益



谷歌全球编程挑战赛

→ **赛制**

Online Qualification Round：共27小时，实际不需要。达线晋级

Online Round 1（n进4500）：分A/B/C三轮子竞赛，可参加任何一轮或者全参加，直到晋级。每轮晋级1500人

Online Round 2（4500进1000，可获限量版T恤）

Online Round 3（1000进25）

World Finals：每年在不同地方的Google Office举行

→ **奖金**：冠军$15000，亚军$2000，季军$1000，4-5名$100

→ **中国选手成绩**

楼天城：2008/2009冠军（IOI2004金牌，清华大学，导师：姚期智）

漆子超：2009亚军（**IOI2009金牌，清华姚班**）

顾昱洲：2014季军（IOI2012金牌，清华大学，MIT）

→ **对留学的意义**：你懂的

费用流

MCMF模板

费用流量网络

→ **G=(E,V,C,B)，B是又一组边权**

b(u,v)=边u→v上**单位流量**的费用

→ **最小费用最大流问题（Minimum Cost Maximum Flow，MCMF）**

求G上总流量达到最大的前提下，总费用的最小值

算法：每次找R上**（费用的）最短路**增广

证明：拟阵最大权独立集问题，Over

→ **SPFA找最短路√**

→ **Dijkstra找最短路**🤔

残量网络存在负权边，直接Dijkstra不行

但可通过势函数转化：b(u,v)→b(u,v)+h[u]-h[v]

这种颜色
表示扩展内容

```
queue <int> q;
bool spfa(int s,int t){
    memset(d,0x7f,sizeof(d));
    memset(flow,0x7f,sizeof(flow));
    memset(inq,0,sizeof(inq));
    q.push(s),inq[s]=1,d[s]=0,pre[t]=-1;
    for (int x,y;!q.empty();){
        x=q.front(),q.pop(),inq[x]=0;
        for (int i=hd[x];i;i=es[i].nxt){
            if (es[i].f>0 && d[y=es[i].t]>d[x]+es[i].c){
                d[y]=d[x]+es[i].c,pre[y]=x,lst[y]=i;
                flow[y]=min(flow[x],es[i].f);
                if (!inq[es[i].t])
                    inq[es[i].t]=1,q.push(es[i].t);
            }
        }
    }
    return pre[t]!=-1;
}
```

比SSSP多维护一个可增广流量即可

并记录方案

```
void MCMF(){
    for (int x=t;spfa(s,t);){
        mxF+=flow[x=t];
        mnC+=flow[t]*d[t];
        for (;x!=s;x=pre[x]) {
            es[lst[x]].f-=flow[t];
            es[lst[x]^1].f+=flow[t];
        }
    }
}
```

累加流量

累加费用

反转增广路

B=1时：SPFA→BFS，如上算法→EK

P1714 序列

→ **破题（最优化问题）**

决策对象：长n的数列选任意个，使总和最大（暴力容易）

约束条件：连续m个位置至多选k个（剪枝容易）

呵呵

建模

→ **首先需要转化一下**

改为**选k轮，每轮连续m个位置至多选1个**

不多：每m个位置每轮至多占据一个，总共不超过k个√

不少：给一种原始方案，按1~k顺序轮流安排轮次即可

→ **So如上操作方式是等价的**

→ **k=1容易做（序列切割DP）**

贪心k次行不行?

反例：m=3,k=2,a={ 5 4 1 4 5}（贪心方案：5 4 1 4 5）

原因：每轮选择不是独立的（每个ai至多选1次）

m=3,k=2

建模

→ **改为选k轮，每轮连续m个位置至多选1个**

单位流量可表示有关联的一组对象（每轮所选）

i→min{i+k,T}连边（设T=n+1）

→ **同一个点只能选1次**

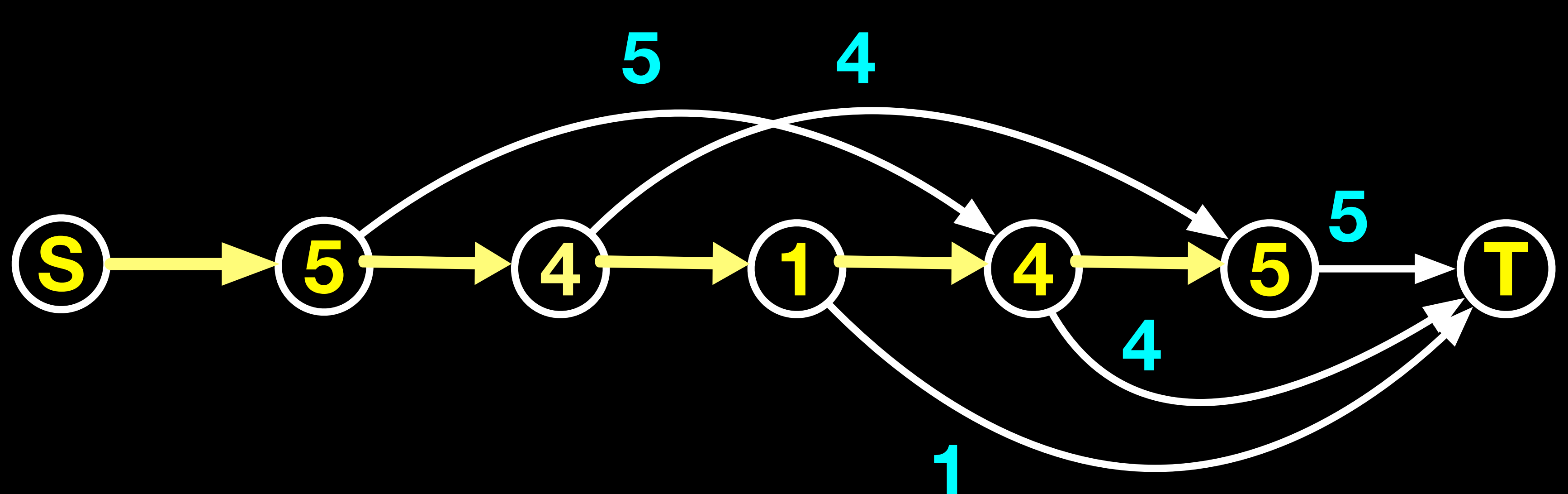
单位容量可以起隔离作用：以上各边容量设为1（**收益**显然为ai）

→ **总共至多选k轮**

S出发的容量可视为“资源”：S→1建边，容量为k，收益为0

→ **还差一点**

每一轮间隔也可以>k：i→i+1建边，容量为k，收益为0



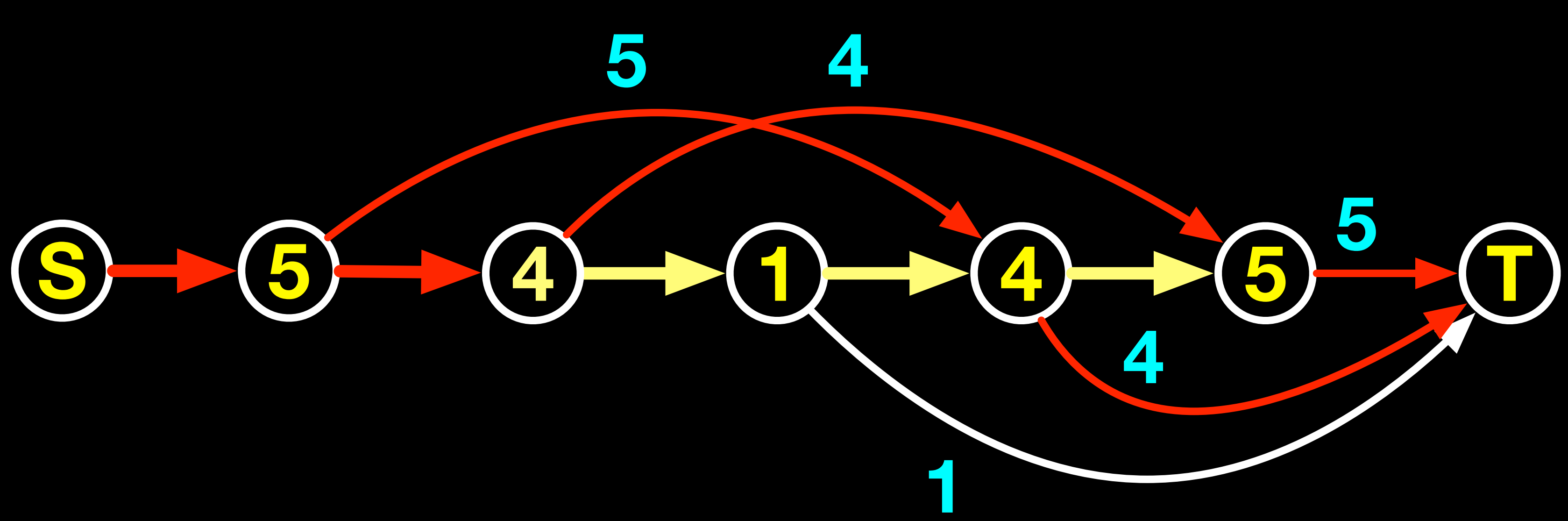
由于是收益，此处求的是**最大**收益最大流
So What，图是DAG，最长路一样求

```
for(int i=1;i<=n;i++){
    if (i<=n-m) add(i,i+m,1,a[i]);
    else add(i,T,1,a[i]);
}
for(int i=1;i<=n-1;i++) add(i,i+1,K,0);
add(0,1,K,0),add(n,T,K,0);
MCMF();
```

建图

建图

Over



P1715 视频课

→ **最优化问题**

决策对象： m种区间的选择个数Zi,i=1..m

约束条件： sum{ Zi*[si≤x≤ti] } ≥a[x]，x=1..n

（这个方括号表示yes/no）

优化目标： min { sum {Zi*ci} }（蛋疼背包问题？）

建模

人数守恒<=>流量守恒

→ 将未雇佣的人也计入（人才库）

则**人有2种状态**（工作/休息），且**总人数守恒**（=inf）

→ **从第x天开始，分两类出边**

x→x+1，表示休息，费用=0

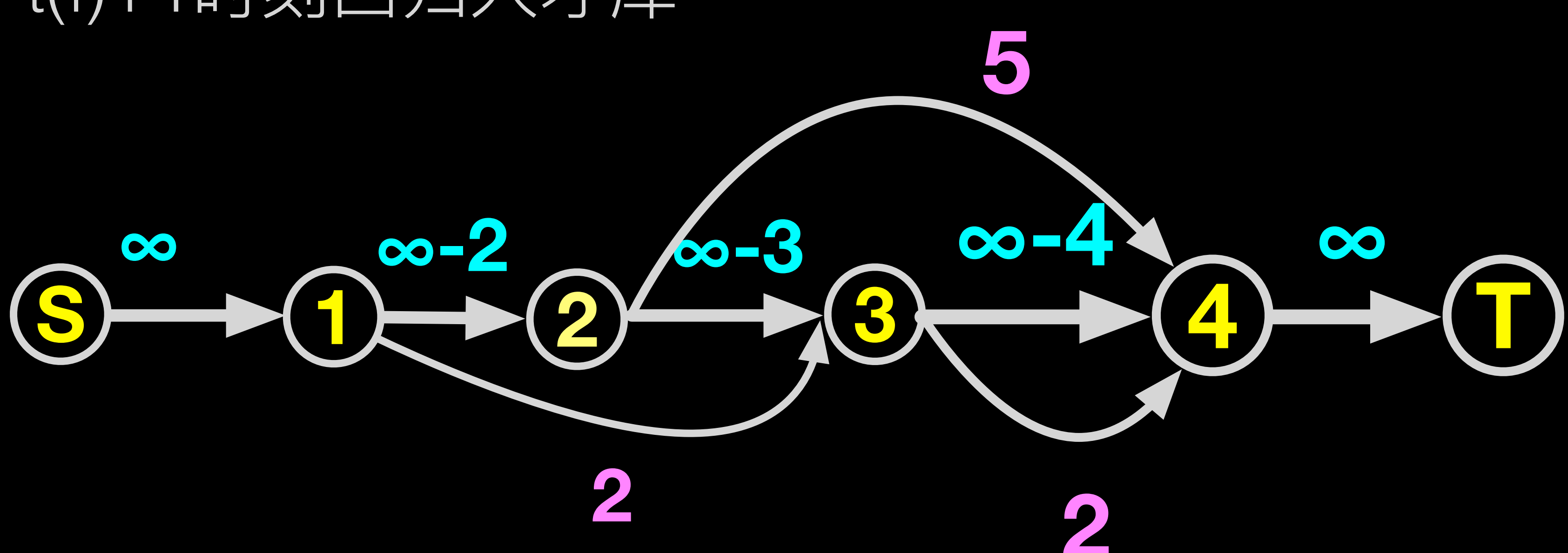
所有s(i)=x的人，表示雇佣此类，费用为c(i)，容量为inf

→ **如何保证x→x+1有至少a(i)个人工作?**

x→x+1休息的人不超过inf-a(x)个（容量）

→ **如何表示s(i)雇佣的人可以工作到t(i)?**

t(i)+1时刻回归人才库



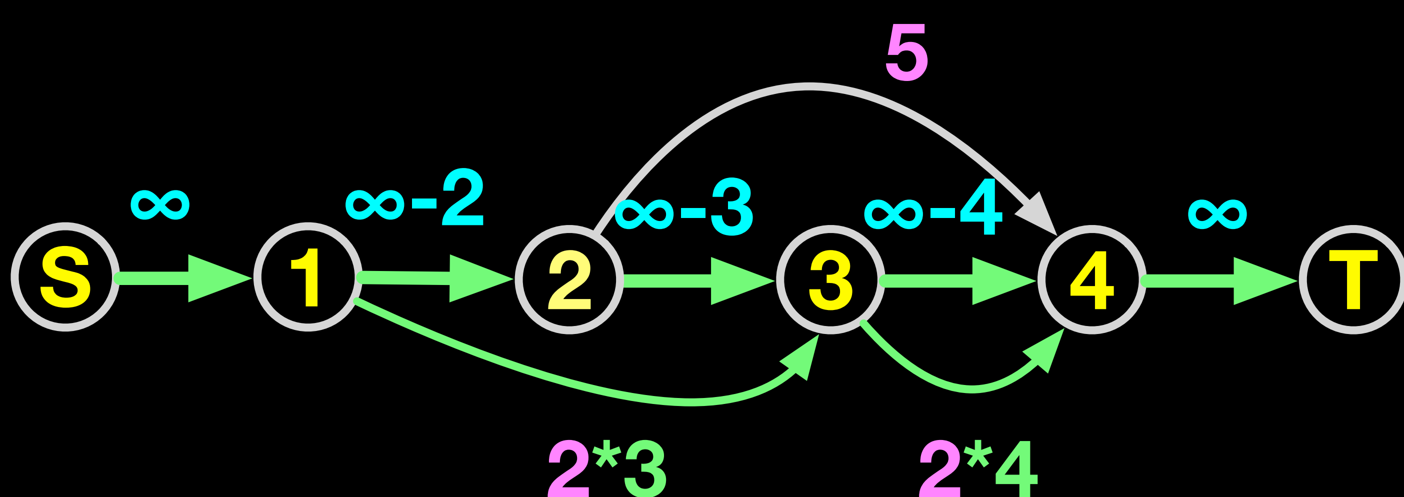
=

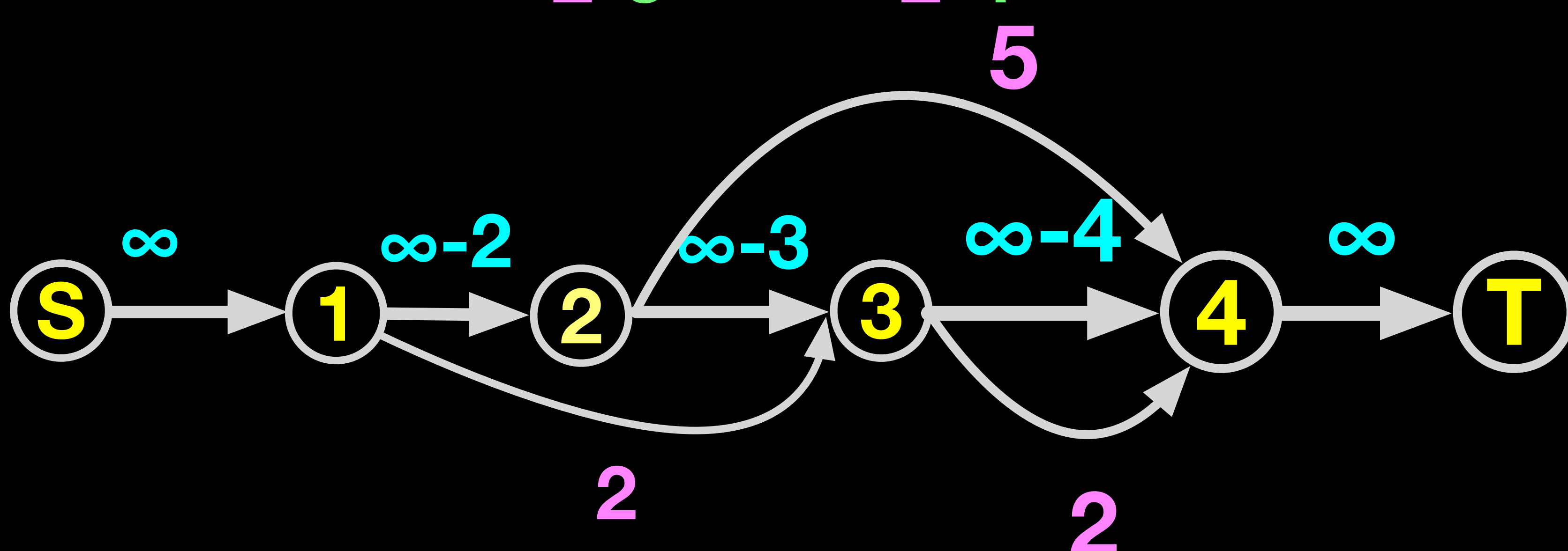
建模

→ **求最大流（流量显然=inf）**

流量守恒保证了约束条件成立，求最小费用最大流即可

→ **流量平衡可作为某种“守恒”约束条件**





ZKW对网络流神一样的论述

→ **拓展问题**

最小费用可行流

容量有下限的最大流

费用有负环的MCMF

→ **ZKW费用流（几种优化）**

使用KM算法（顶标+可行子图扩张）替代SPFA

DFS多路增广（费用流版本Dinic）

原始对偶算法

→ **具体细节，本人体力不支，你们暂且自己研究下吧囧...**

https://artofproblemsolving.com/community/c1368h1020435
https://artofproblemsolving.com/community/c1368