网络流初步

二分图匹配

- 对于一个无向图G=(V,E),它的顶点集合恰好可以分成两个不相交的子集(设为A,B)，并且在这个无向图中，任意一条边所连接的两个顶点都分别属于这两个不同的子集，那么我们将这个无向图称作**二分图**（也称**二部图**）
- 在一个二分图中，如果存在它的一个子图，这个子图中的任意两条边都没有公共顶点，我们把这样的子图叫做二分图的一个**匹配。**而如何在二分图中选择一个边数最大的匹配就叫做图的最大匹配问题。
- 如果匹配数$|M| = min\{|A|, |B|\}$，则称此匹配为**完全匹配。**特别地，当|A|=|B|=|M|时，称这个匹配为**完美匹配。**

现在给出一幅二分图，要求出其最大匹配.

匈牙利算法

算法流程

1. 把初始的匹配M置为空
2. 寻找一条增广路，通过异或操作更新最大匹配
3. 重复步骤2直到找不到增广路径

复杂度

时间复杂度$O(VE)$空间复杂度$O(E)$

最大流

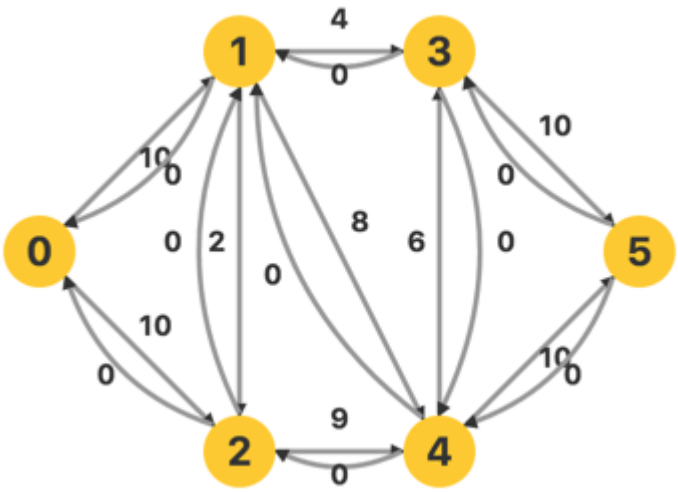
- 设G(V,E)为一个有向图，它的每条边<u,v>都被赋予了一个非负的实数c作为边的**容量**，记为c(u,v)。网络流(network flow)指为这个有向图分配流并且使得它每条边上的流量都不能超过这条边的容量。
- 在运筹学中，有向图称为网络，边称为弧(arc)。在这个有向图中指定两个顶点分别叫做源点Vs和汇点Vt。
- 除源点和汇点，要保持每个顶点进出总流量相同，源点的总出流量和汇点的总入流量相等。
- 流量是指通过每条弧<u,v>的实际流量，记为f(u,v)。所有弧上流量的集合f={f(u,v)}组成了一个网络的网络流。在网络G(V,E)中，可行流为满足如下条件的网络流：

$$\text{弧流量满足：}\ 0 \leq f(u,v) \leq c(u,v) \quad < u,v > \in E$$

网络流相关名词

- 在网络中，链是指网络中的一个顶点序列，这个顶点序列中的前后两个顶点之间有弧相连。链上的弧被分为两类：
- 前向弧：方向与链的正方向一致。前向弧组成的集合记为P+
- 后向弧：方向与链的正方向相反。后向弧组成的集合记为P-
- 网络中的增广路是指从源点到汇点的一条链，并满足如下性质
- 所有链上的前向弧都是非饱和弧，即0 ≤ f(u,v) < c(u,v)

- 所有链上的后向弧都是非零流弧，即0 < f(u,v) ≤ c(u,v)
- 对于网络G(V,E)和可行流f，弧<u,v>上的剩余容量表示该弧上可以增加的流量，记为c'(u,v)=c(u,v)-f(u,v)。每条弧有一条对应的反向剩余容量c'(u,v)=-f(u,v)。由剩余容量组成的网络被称为剩余网络，比如下面这个网络：



最短路算法（EK算法）

复杂度

时间复杂度$O(VE^2)$

太慢了.没什么用

Dinic算法

Dinic算法基本步骤

1. 初始化网络及其网络流
2. 构造剩余网络和层次网络，若汇点不在层次网络中则算法结束。
3. 在层次图G_L内用一次DFS过程进行增广每次都向层次增加1的方向增广，每次增广完毕，在层次网络中要去掉因改进流量而导致饱和的弧，DFS执行完毕则该阶段增广完毕。
4. 转步骤2。

复杂度

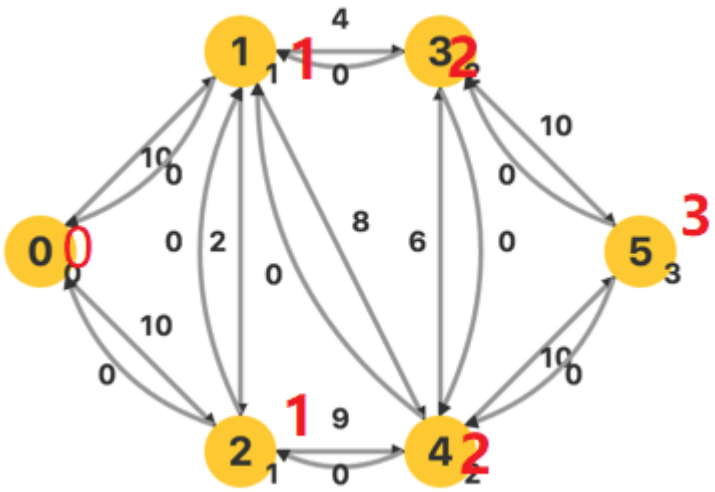
实际复杂度$O(V^2E)$

- 小贴士：在实际运行中远远达不到这个上界，一般可以护理10^4到10^5规模的算法.其中求解二分图时有更好的复杂度$O(m\sqrt{n}$且实际上的复杂度比这个更加优秀（说明匈牙利也是一个没什么用的算法）

Dinic算法演示

首先设maxflow=0

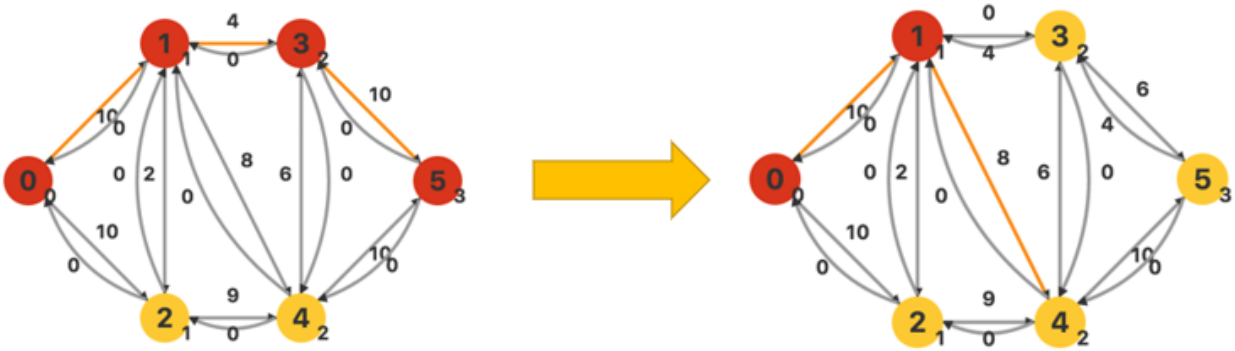
- 如图，源点为0，汇点为5，首先对其建立层次图



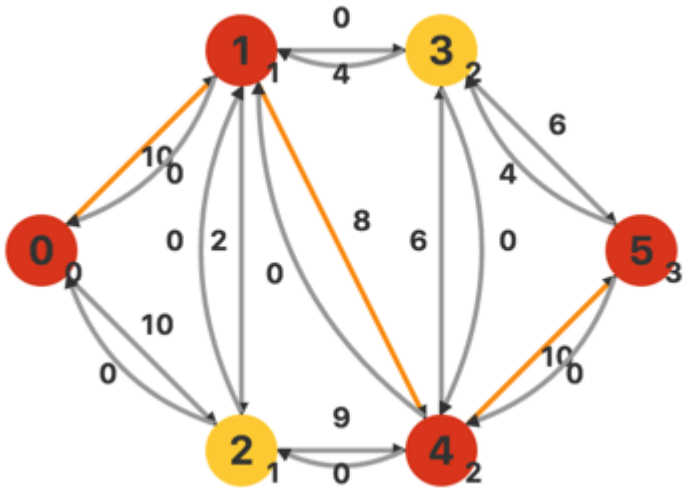
- 每个顶点右下方的数字就是其对应的层次。接下来进行DFS，找到一条流量为4的增广路。

此时maxflow=4

- 回退至顶点3，发现它没有其他的允许弧，继续回退至1。在回退的过程中不断修改流经的弧的容量。接下来，发现顶点1有一个允许弧连至4，继续DFS。此时流量上限已经更新为10-4=6（指<0,1>边）。



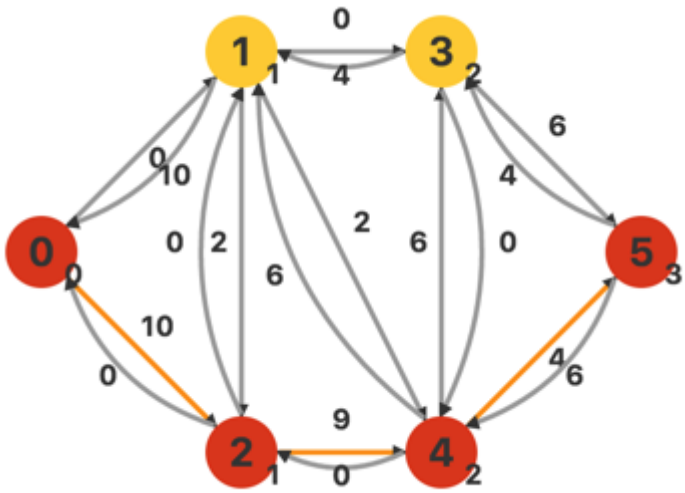
- 又找到一条流量为min(6,8,10)=6的增广路。



此时的最大流量maxflow=4+6=10

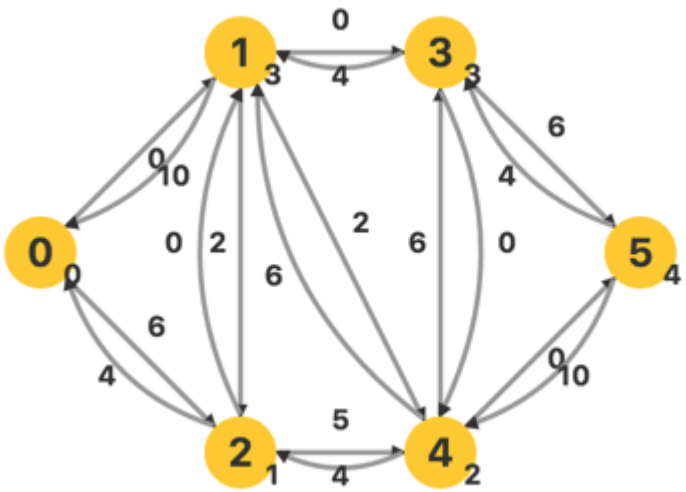
- 此时顶点1的流量上限已经为0，从顶点1回退至源点。修改相关弧的剩余容量。

- 继续搜索源点的相邻顶点2，并找到一条增广路。由于源点的容量上限始终是INF，所以增广路的流量为min(10,9,4)=4。

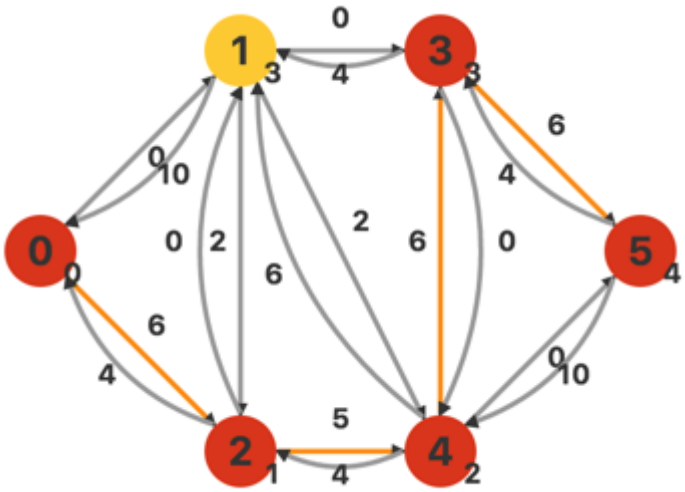


此时的最大流量为maxflow=4+6+4

- 此时节点二已经跑满，回退到源点，并调整相应的弧的容量，发现源点没有其他的相邻顶点，dfs过程结束
- 再次BFS，重新建立层次图



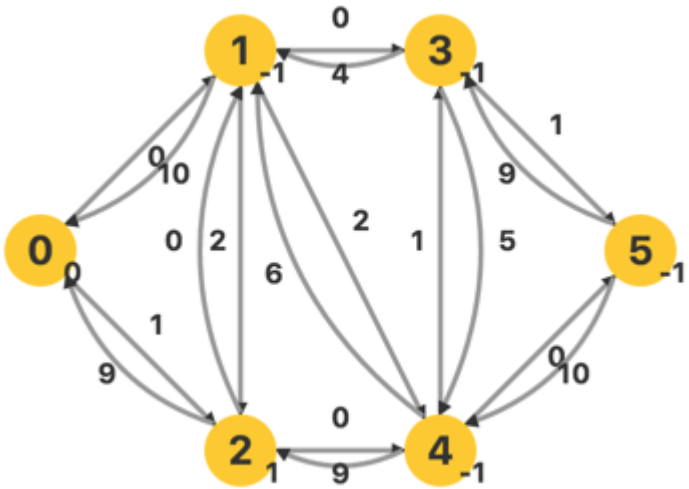
- 找到一条增广路，流量为min(6,5,6,6)=5。



此时的最大流量maxflow=4+6+4+5

- 发现没有其他增广路，回退至源点，更新相关弧的容量。

- 源点没有其他的相邻点，重新BFS建立新的层次网络，发现汇点不在层次网络中，算法结束

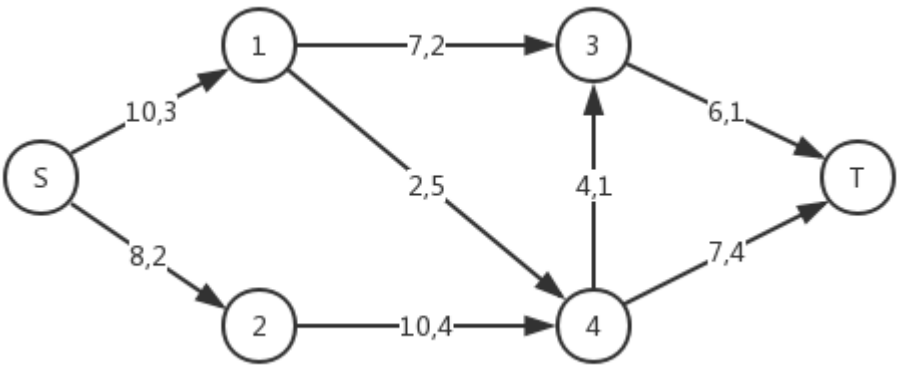


最小割

- 在网络G(V,E)中，设E'⊆E，如果在G中删去E'后，图G不再连通，则称E'是G的割。割将图的顶点集合V划分成两部分：S和T，我们用(S,T)来表示一个割。如果割划分的两个点积满足源点Vs∈S，汇点Vt∈T，那么称这个割为网络的s-t割。如无特殊说明，在网络中的割指的是s-t割。
- 通过最大流的原理（跑至汇点不在层次网络中，即s,t不联通）不难想到将最大流割去即是最小的割

最小费用最大流

- 在一个网络中，可能存在多个总流量相同的最大流f，我们可以在计算流量的基础之上，给网络中的弧增加一个单位流量的费用（简称费用），在确保流量最大的前提下总费用最小，这样的最大流被称为最小费用最大流。
- 在下面这个网络中，弧上有逗号分隔的两个数分别为弧的容量和单位流量费用。例如，一条流量为2、经过S->1->4->T的流的费用为(3+5+4)*2=24。



基本步骤

1. 网络初始流量为 0
2. 在当前的剩余网络上求出从 S 到 T 的最短增广路 min_dist(s,t)，以每条弧的单位流量费用为边权，“距离”即为该路径上的边的单位费用之和。如果不存在，则算法结束
3. 找出这条路径上的边的容量的最小值 f，则当前最大流 max_flow 扩充 f，同时当前最小费用 min_cost 扩充 f * min_dist(s,t) 。
4. 修改增广路上弧的流量（正向边的容量减少 f，反向边的容量增加 f），重复步骤 2

证明 算法采用贪心的思想，每次增加流量的路径花费最小，因为最大流量有限，每次循环流量都会增加，因此算法结束时，流量必定会达到最大流量，而每次都是最小花费，所以最后的总花费最小。

复杂度

时间复杂度$O(\text{寻找最短路} * maxflow)$

建模技巧选讲

最大权闭合子图

什么是最大权闭合子图?

给出一张图，图中的点与点之间通过有向边连接，选出一些点，保证这些选出的点的后继点也属于这些被选出来的点，且权值最大则是最大权闭合图。

如何寻找最大权闭合子图?

方法：设立一个虚源点和一个虚汇点，所有的正权点都向虚源点连接容量为其权值的边，所有的负权点都向汇点连接容量为权值的边，原本有边的点之间连接容量无限大的边即可，最后权值之和减去这幅图的源点与汇点之间的最大流即可。

正确性：我们知道最大流即是最小割，而这个模型中，我们只可能从与源点还有汇点连接的点去割，而其余的边由于容量无限，所以并不可能割，而这种割边的一端是源点或者汇点的割称作为简单割。

而可以证明的是对于一个网络N的简单割[S,T]与图G的闭合图V1方案存在一个对应关系$V_1 \cup \{s\} = S$、$\overline{V_1} \cup \{t\} = T$

证明:

闭合图对应简单割：由于V1闭合，所以V1与V2之间不存在边，s与t之间不存在边，所以S，T之间也就不存在边，所以其也就对应着割，而本图的所有割为简单割，所以闭合图对应简单割

简单割对应闭合图：由于S，T为简单割，故S,T之间不存在边，而s，t之间也不存在边，故$S-\{s\}$与$T-\{t\}$之间也就不存在边，因而$S-\{s\}$图的所有后继点(除了s，t)必然仍然在$S-\{s\}$内，$T-\{t\}$同理，所以简单割对应闭合图

最小割的流量

设点集V中权值为正的点的集合为V^+权值为负的点的集合为V^-

那么最小割[S,T]之间割去的最小流量就是

$$c[S,T] = \sum_{v \in V_w^+} w_v + \sum_{v \in V_1^-} -w_v$$

证明:

$[S,T] = [\{s\}, V_2] \cup [V_2, V_1] \cup [V_1, t]$

因为[S,T]为简单割，所以$[V_2, V_1]$为空集

因为只有正权与V1之间有连边，负权与V2之间有连边

所以

$[\{s\},V_2]=[\{s\},V_2^+]$

$[V_1,\{t\}]=[V_1^-,\{t\}]$

所以$c[S,T]=\sum_{v\in v^+}w_v+\sum_{v\in v^-}-w_v$

得证

如何保证得到的就是最大权闭合图

最大权闭合图的总权值为$w(V_1)=\sum_{v\in V_1^+}w_v-\sum_{v\in V_1^-}(-w_v)$

令w(V1)+c[S,T]则有

$w(V_1)+c[S,T]=\sum_{v\in V_1^+}w_v-\sum_{v\in V_1^-}(-w_v)+\sum_{v\in V_2^+}w_v+\sum_{v\in V_1^-}-w_v$

$=\sum_{v\in V_1^+}w_v+\sum_{v\in V_2^+}w_v$

$=\sum_{v\in V^+}w_v$

经过移项可以得到

$ans=w(V1)=\sum_{v\in V^+}w_v-c[S,T]$

所以使正权和减去最小割既是答案的最大值

带二分图性质的最大权闭合图优化

当选择一条边两端的点时视作该边被选择，选择一条边，可以获得一个价值，选择点则需要付出一定的代价，问如何获得最大的价值。

设E为边，V为点，则问题的本质就变为：

$Maximize(\sum W_E-\sum W_V)$

考虑到选了某条边，那么这条边的点就必选，其本质就变成了一个最大权闭合图问题，我们可以将边也抽象为点，向其两个端点连边，然后在这幅图上跑最大权闭合图即可，复杂度为O(MaxFlow(n+m,n+m))。

考虑上面这个问题，假如原图是一张稠密图（就如例题），那么这个算法的复杂度就会变得非常糟糕了，为此我们需要作出优化。

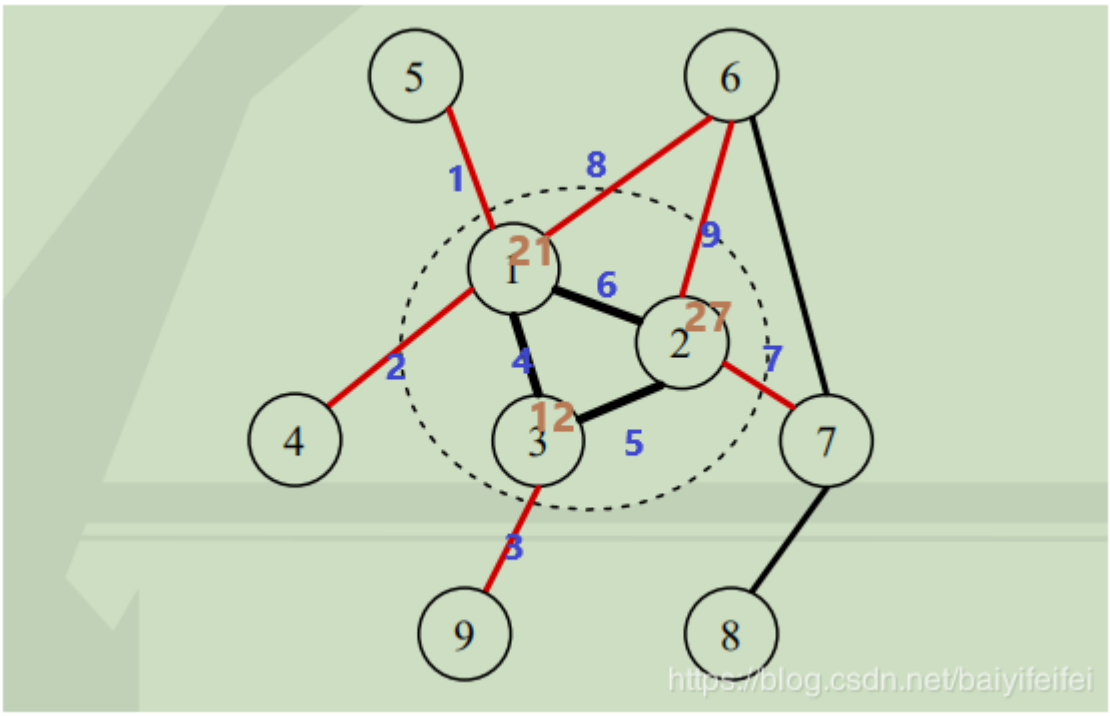
再来看这个式子

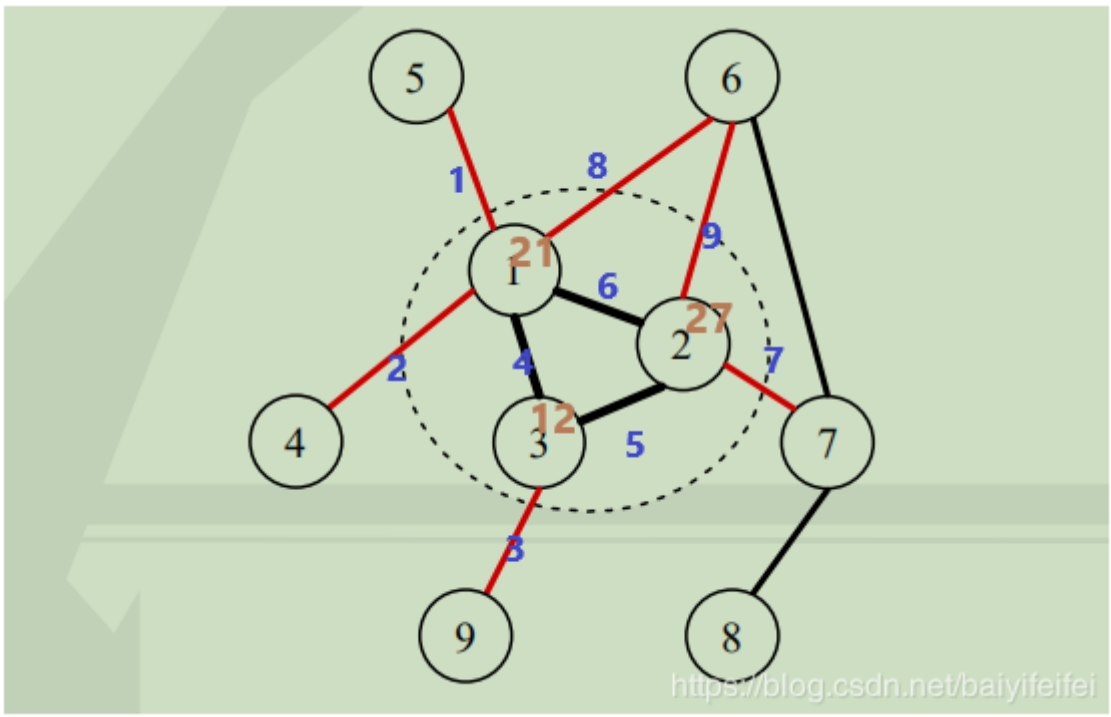
$Maximize(\sum W_E-\sum W_V)$

我们可以通过乘上-1来使这个问题转化为求最小值

$Minimize(\sum W_V-W_E)$

考虑到边的总价值等于所有的点相连的边的总价值减去那些已经选择的点和不选择的点之间的边除去2就是选择的边的总价值。也就是选择的点所连接的边的总价值减去与不选择的点之间的最小割/2，具体上可以看胡伯涛论文中的这幅图：





其中蓝紫色与棕色的数字是我自己标上的，其中蓝紫色的为边的价值，棕色的为与这个端点连接的边的价值之和，黑色加粗的边为选择的边，黑色虚线圆内的点（1,2,3）即是选择的边，我们不难发现图中选择的边的总价值就是

[(21+27+12)-(1+2+8+9+7+3)]/2=15

其中(1+2+8+9+7+3)也就是红色边的价值之和就是已选择的1,2,3与未选择的4,5,6,7.8,9之间的最小割

我们用W_{VE}来表示与一点相连的所有的边的价值总和,已经选择的点集为V未选择的点集为V'，那么原式就可以表示为

$$\sum W_v - \frac{\sum W_{vx} - c[V, V']}{2}$$

进一步转化，原式就变成了

$$\sum(W_v - \frac{W_{vx}}{2}) + \frac{c[V, V_i]}{2}$$

为了使得原式更加整齐，对原式乘2

$$\sum(2W_v - W_{vx}) + c[V, V_i]$$

因此原图中的每个点的权值就是$2W_v - W_{vx}$，考虑最小割只能接受非负的边权，所以要为所有的点权加上一个极大值，令为U，然后答案就是

$$\frac{U * n - C[S, T]}{2}$$

可行性证明

$C[S,T] = \sum_{u\in S, v\in T} c_{u,v}$

$= \sum_{v\in \overline{V'}} C_{s,v} + \sum_{u\in V'} C_{u,t} + \sum_{u\in V', v\in \overline{V'}, (u,v)\in E} C_{u,v}$

$= \sum_{v\in \overline{V'}} U + \sum_{u\in V'} (U + 2W_v - W_{vx}) + \sum_{u\in V', v\in \overline{V'}, (u,v)\in E} W_E$

$= U|V| + \sum_{u\in V'} (2 * W_v - W_{vx} + \sum_{v\in \overline{V'}, (u,v)\in E} W_E)$

$= U|V| + \sum_{u\in V'} (2 * W_V - \sum_{v\in V', (u,v)\in E} W_E)$

$= U * n + 2\sum W_V - 2\sum W_E$

$= U * n - 2(\sum W_E - \sum W_E)$

$= U * n - 2 * ans$

所以$ans = \frac{U*n-C[S,T]}{2}$

K次区间覆盖

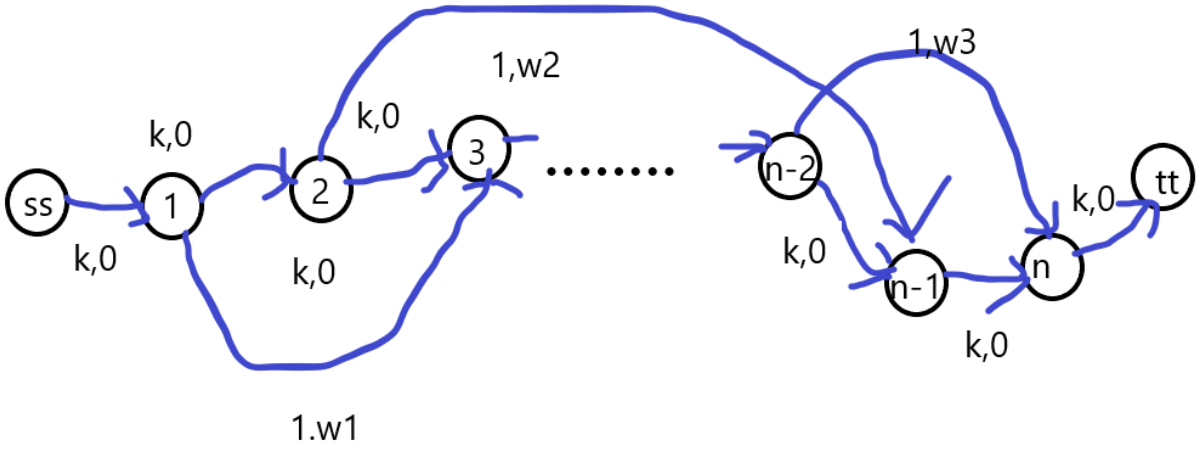
什么是K次区间覆盖

给出n个开区间，区间最多可以选一次，且所有的区间都有一定的权值，与此同时，要保证实轴上的所有数被选择的次数要小于k次。问如何选出最大的权值。

如何找到最大K次区间覆盖

简单地想一下，我们可以每个区间建立一个虚点，超级源点向虚点连接一条容量为区间长度，价值为这个区间价值的负边，然后这个点向区间内所有的点连一条容量为1，价值为0 的边，然后所有的点向超级汇点连接一条容量为1的价值为0 的点。但是这个实际上并不能保证一个区间被选中之后，其中的所有的点都被选中。

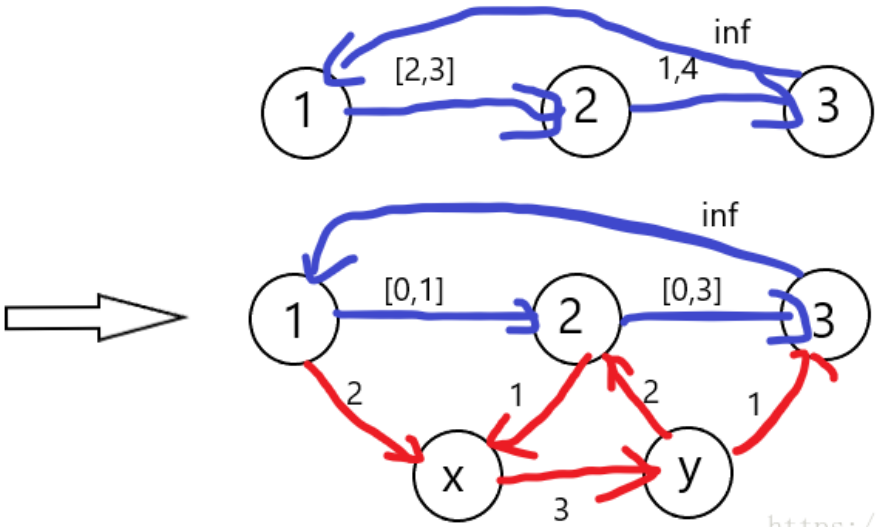
换一个建模方法，设置一个超级源点，一个超级汇点，源点向第一个点连一条容量为k，费用为0的边。再将所有区间内的数之间连上用容量为k费用为0 的边连接上再将最后一个点与汇点用一条容量为k的费用为0 的边，再在区间的两个端点之间连接一条容量为1的，费用为-w的边，形成这样一种结构，如下图：



上下界网络流

上下界可行流

对于一副循环图（即无源点和汇点，起始点和终末点之间存在一条容量无限大的边）来说，假设其中的每条边的流量都介于[l,r]直接，我们不妨将l取出，将原边变为[0,r-l]，与一条容量为l的必行边，并在这条边中间插入两个点，设为x，y，细节见图示



我们不难发现从x到y的流量只有x->y一条，因此其x->y的最大流也就是3，于是我们可以据此推理出假使存在一条从y到x的最大流等于3时就是所有的必行边都被走过了，那么这个图就是可行的。

但是本题是一个有汇源点的图，所以我们要想办法将之转化为循环图，具体方法就是在汇点和源点直接也连上一条无穷大的边即可。需要注意的是：

从汇点向源点连！

上下界最大流

在可行流跑过的残余网络上拆去<t,s>跑最大流加上上下界的流量即为最终答案

上下界最小流

法1：用之前所述的方法求可行流，如果可行流不存在，则算法结束；否则从可行流出发，倒向求解，即保持网络中弧方向不变，将t作为源点，将s作为汇点，把可行流放大，最终最终求得的最大流即为从s到t的最小流。

法2：有源汇的最大流最小流也可以通过二分答案求得，即二分t->s的下界(最大流)和上界(最小流)，复杂度多了一个log

法3：求虚源点到虚汇点的最大流，建<t,s>容量无穷大，再跑一遍虚源点到h虚汇点的最大流，答案即为t到s之间的流量

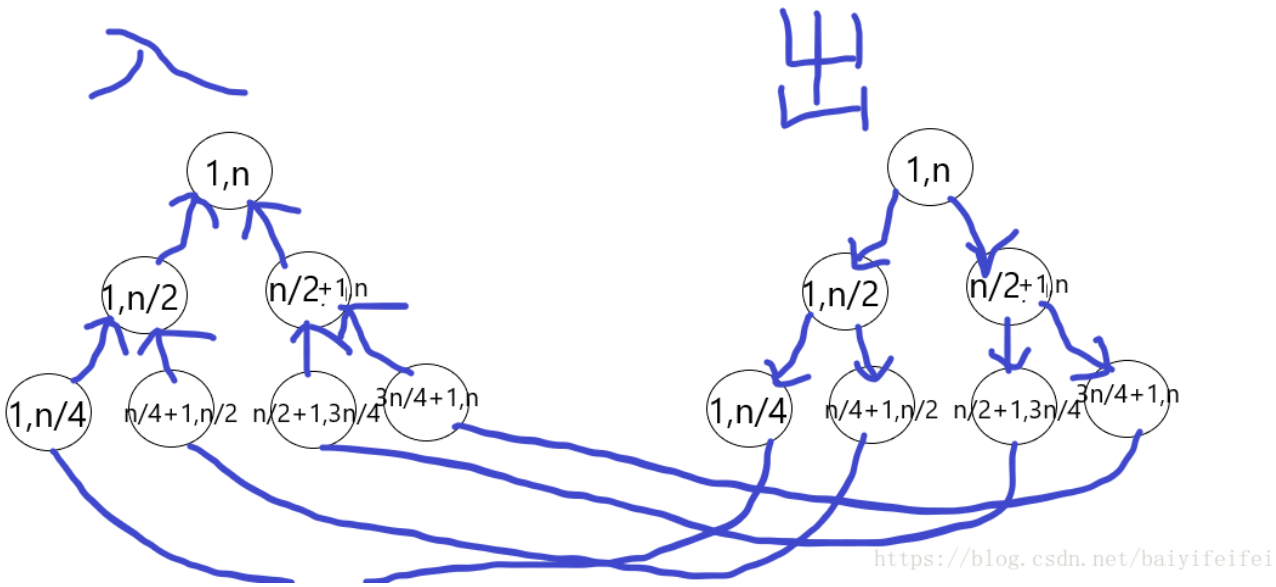
线段树网络流

什么时候需要线段树网络流

边数较大，且为一段区间的权值的点项另一端区间的权值的点连边

如何优化

就像常规线段树优化建图，建两棵线段树，入树上所有的节点向其父节点连边，出树上所有的节点向其子节点连边



然后常规网络流的跑法即可.