图论模型的构建

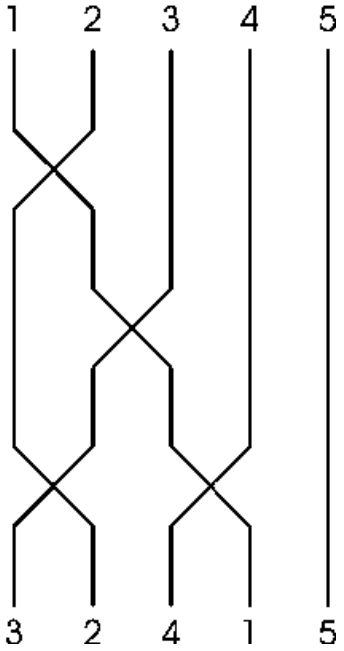
长沙市雅礼中学 朱全民

NOIP若干图论的考题

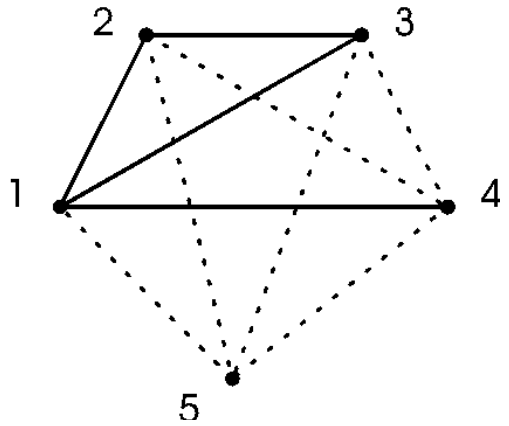
- Core(2007):图的多源最短路算法及其简单处理
- 双栈排序(2008):栈的应用+二分图的搜索
- 最优贸易(2009):基本图论

问题：求网线线序

- 网线从机房连接到办公室
- 在机房，所有网线从左到右编号为1,2,3,…,*N*
- 给出每两条线是否交叉的信息，请计算办公室内从左到右各条线的编号



(a)



(b)

选址问题

- 现准备在 n 个居民点$v_1, v_2, \cdots, v_n$中设置一银行. 问设在哪个点, 可使最大服务距离最小? 若设置两个银行, 问设在哪两个点?
- 模型假设　假设各个居民点都有条件设置银行, 并有路相连, 且路长已知.

模型建立与求解

- 用**Floyd**算法求出任意两个居民点v_i, v_j之间的最短距离, 并用d_{ij}表示.

(1) 设置一个银行, 银行设在 v_i 点的最大服务距离为

$$d_i = \max_{1\le j\le n}\{d_{ij}\}, \quad i = 1,2,\ldots,n.$$

求k, 使 $d_k = \min_{1\le i\le n}\{d_i\}$.

即若设置一个银行, 则银行设在 v_k 点, 可使最大服务距离最小.

(2) 设置两个银行, 假设银行设在v_s, v_t 点使最大服务距离最小.

$$\text{记 } d(i,j) = \max_{1\le k\le n}\{\min\{d_{ik}, d_{jk}\}\}.$$

则s, t 满足: $d(s,t) = \min_{1\le i<j\le n}\{d(i,j)\}$.

进一步, 若设置多个银行呢?

求k, 使 $d_k = \min\limits_{1\le i\le n}\{d_i\}$.

即若设置一个银行, 则银行设在 v_k 点, 可使最大服务距离最小.

(2) 设置两个银行, 假设银行设在v_s, v_t 点使最大服务距离最小.

记 $d(i,j) = \max\limits_{1\le k\le n}\{\min\{d_{ik}, d_{jk}\}\}$.

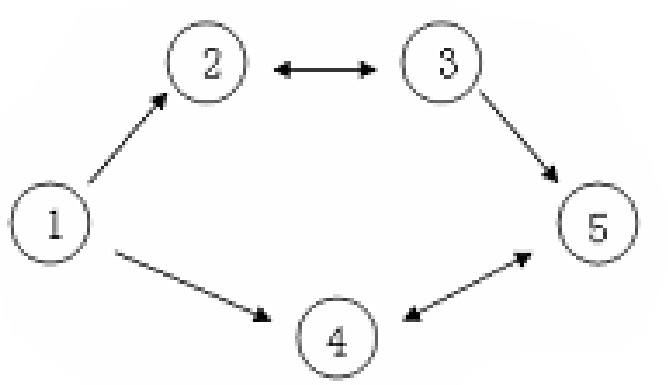
则s, t 满足:

$$d(s,t) = \min_{1\le i<j\le n}\{d(i,j)\}.$$

进一步, 若设置多个银行呢?

最优贸易

- 某国有M个城市N条道路，任意两个城市有道路，有一部分道路为单行线，一部分为双向道路。
- 某人去该国旅游，从城市1出发到城市n结束，他想做水晶球的生意一次挣点旅行费用，每个城市有一个水晶球的价格（买入卖出都一样），他可以经过每个城市多次。
- 问他能挣最多的费用为多少？
- 如下图，假设城市1~5的价格为4，3，5，6，1
- 则选择1-4-5-4-5路线，
  挣得5



分析

- 这是一道非常典型的图论题, 如果有扎实的图论基础解决起来并不困难.
- 解决这道题的关键是发现, 我们可以将原图中的任意一个强连通分量收缩为一个点, 这个新点的买入价格等于该强连通分量中最小的买入价格, 这个新点的卖出价格等于该强连通分量中最大的卖出价格. 这是因为, 这个新点的性质和一个强连通分量是一样的, 如果我们要在一个强连通分量中进行购买操作, 一定会选择买入价格最小的那个点, 如果我们要在一个强连通分量中进行卖出操作, 也一定会选择卖出价格最大的那个点.
-

分析

- 所以算法就非常清晰了. 首先利用DFS将所有的强连通分量收缩, 这样我们就可以得到一个有向无环图G. 由于G中没有环, 我们可以对G进行拓扑排序, 然后利用递推求得到达某个点i时, 可能的最低买入价格best(i)是多少, 以及该点最终能否到达终点. 最后对于所有能够到达终点的点p, 设其卖出价格为sell(p), 在sell(p)-best(p)中取最大值即得到答案. 时间复杂度仅为O(V+E).
-
- 在实现的时候最好使用栈消除DFS中的递归调用, 因为图中的点可以达到100000, 当递归深度达到这么大的时候有相当大的几率发生栈溢出. 参考实现中采用了非递归实现DFS.

奇怪的电梯

- 大楼的每一层楼都可以停电梯，而且第i层楼(1<=i<=N)上有一个数字Ki(0<=Ki<=N)。电梯只有四个按钮：开，关，上，下。上下的层数等于当前楼层上的那个数字。当然，如果不能满足要求，相应的按钮就会失灵。例如：3 3 1 2 5代表了Ki(K1=3,K2=3,……)，从一楼开始。在一楼，按“上”可以到4楼，按“下”是不起作用的，因为没有-2楼。那么，从A楼到B楼至少要按几次按钮呢？
- 输入：

  二行，第一行为三个用空格隔开的正整数，表示N,A,B(1≤N≤200, 1≤A,B≤N)，第二行为N个用空格隔开的正整数，表示Ki。
- 输出：仅一行，即最少按键次数,若无法到达，则输出-1。

构图

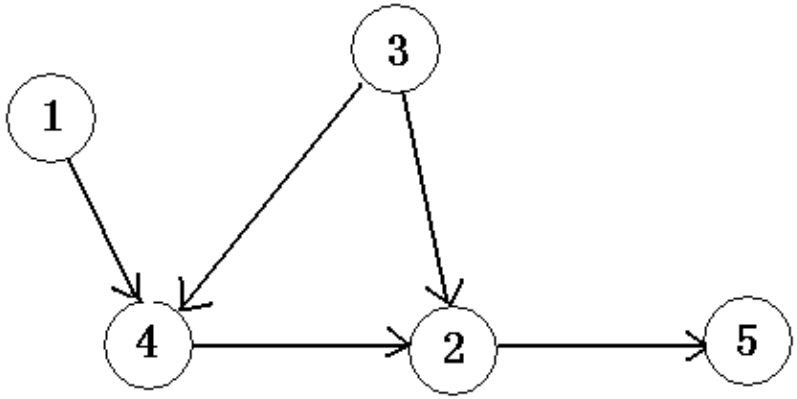
LIFT.IN

5 1 5

3 3 1 2 5

LIFT.OUT

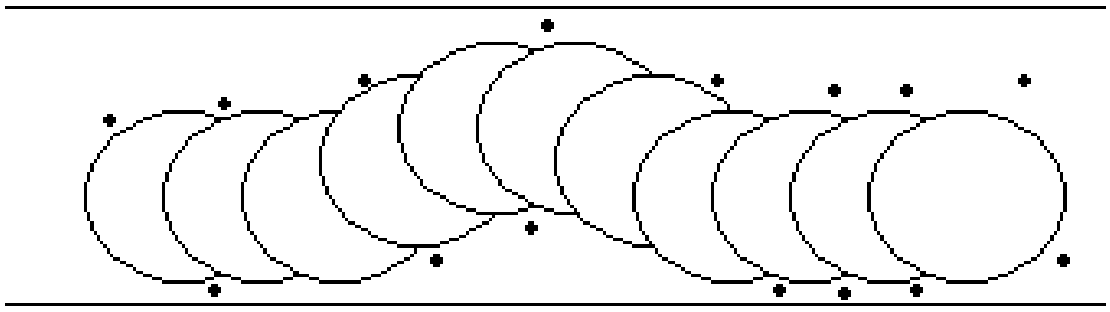
3



- 对于A楼而言，实际上对它最多只能做2个操作，上到A+X层或下到A-X层，当然前提是存在A+X或A-X层。显然，如果把每一层楼看做一个顶点，如果A楼可以到B楼，则从顶点A引一条到顶点B的边，这样一来，问题就变成了图论中的两顶点间最短路径问题了！当然权设为1就行了。

人穿柱子游戏

- 在一个无限长的条形路上，有n(n<=200)个柱子，体积不计，有一个人想从左边走到右边，人近似看成一个半径为R的圆（如下图），问能否实现？



构图

- 每个圆的大小完全相同，不存在包含，相切（如果内切，就是重合了，如果外切，就是中间不连通的）等等复杂的关系，只有相交和相离的关系，而且如果2个圆之间相交的话，那么这2个圆就是相通的，可以在这2个圆的圆心之间连一条边，增加一个源点，与上边有交点的圆和源点连一条边，增加一个汇点，与下边有交点的圆和汇点连一条边，这样就把一道几何题完全转换成了一道图论题，只要判断源点和汇点之间是否有路就可以了

奇怪的数列

- 编程输入3个整数n，p，q，寻找一个由整数组成的数列（a_1，a_2，……，a_n），要求：其中任意连续p项之和为正数，任意连续q项之和为负数。0<n<100，0<p，q<n，若不存在这样的整数数列，则输出NO；否则输出满足条件的一个数列即可。
- 输入格式：

  仅一行分别表示n，p，q，之间用一个空格隔开。
- 输出格式：

  只有一行，有解即输出这个数列，每个数之间用一个空格隔开。否则输出NO。

分析

- 如果我们按常规思想，直接将第i个整数a_i开始的k个整数之和描述成多项式$a_i+a_{i+1}+…+a_{i+k-1}$的话，问题就很难再往下思考和解决了。所以，我们不防换个角度，暂且撇去每一项数究竟为何值的具体细节，而将注意力集中至连续性这一特点上。设s_i表示数列前i个整数之和，即$s_i=a_1+a_2+…+a_i$。其中$s_0=0$ $(0≤i≤n)$。显然根据题意，有：

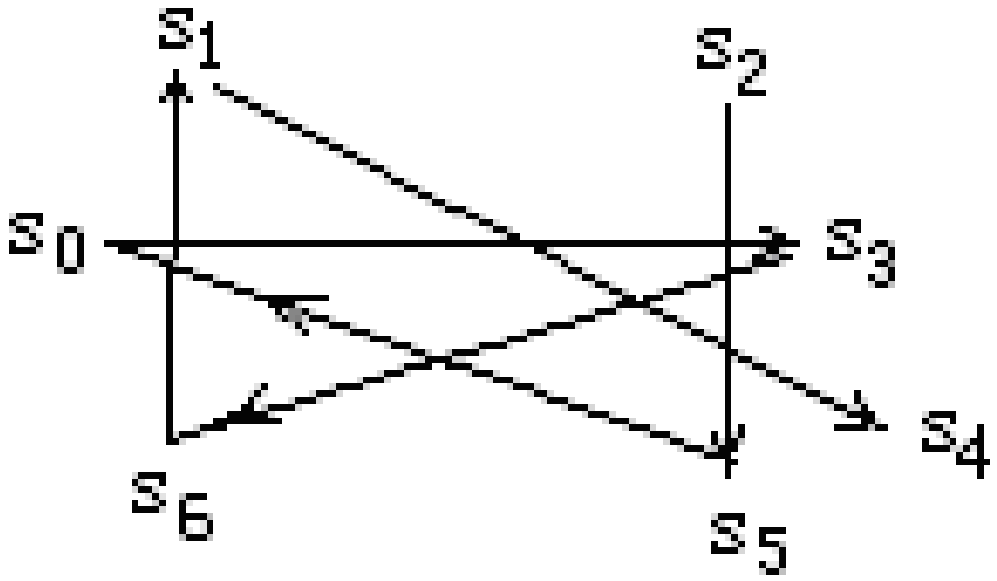
  $s_i<s_{i+p}$ $(0≤i≤n-p)$

  $s_{i+q}<s_i$ $(0≤i≤n-q)$

- 下面，我们把每个s_i抽象成一个点，则根据上述两个不等式可以建立一个有向图，图中共有n+1个顶点，分别为s_0，s_1，……，s_n。若$s_i>s_j$$(0≤i，j≤n)$，则从$s_i$往$s_j$引出一条有向边。

构图

- 对于n=6，p=5，q=3的情况，我们可以建立下图：



- 对图进行拓扑排序；

  if 图有回路 **then** 无解退出

  else 生成拓扑序列**order[0]…order[n]**；
- 那么如果得到了一个拓扑序列，该如何转换成s数组呢？因为拓扑序列中顶点对应的s值是递减的，其中$s_0=0$。
- 如果**order[i]=0**，则依次设定$s_{order[0]}=i$，$s_{order[1]}=i-1$，……，$s_{order[i-1]}=1$，$s_{order[i]}=0$，$s_{order[i+1]}=-1$，……，$s_{order[n]}=i-n$。
- 例如，对于上图所示的有向图，可以得到下表：

i	0	1	2	3	4	5	6
order[i]	2	5	0	3	6	1	4
$s_{order[i]}$	2	1	0	-1	-2	-3	-4

- 所以，得到$s_0=0$，$s_1=-3$，$s_2=2$，$s_3=-1$，$s_4=-4$，$s_5=1$，$s_6=-2$。再根据s的定义，由：　$a_i=(a_0+a_1+…+a_{i-1}+a_i)-(a_0+a_1+…+a_{i-1})=s_i-s_{i-1}$，求出：$a_1=s_1-s_0=-3$，$a_2=s_2-s_1=5$，$a_3=s_3-s_2=-3$，$a_4=s_4-s_3=-3$，$a_5=s_5-s_4=5$，$a_6=s_6-s_5=-3$。显然这个整数数列的任意连续 5 个整数之和为正，任意连续 3 个整数之和为负。

牧场规划

- 小可可的好朋友Sealock最喜欢吃花生了，于是借用了小可可的牧场从事花生选种试验。他以网格的方式，非常规整地把牧场分割成M*N个矩形区域(M*N≤5000)，由于各个区域中地水面、沼泽面积各不相同，因此各区域地实际可种植面积也各不相同，已知区域(i,j)地可种面积使A(i,j)。
- 每个区域种最多只能种植一个品种地花生。可种植面积为零地区域不能被选择用来从事选种试验，同时为了防止花粉传播到相邻区域造成试验结果不正确，任何两个相邻的区域都不可以同时种植花生。这里说的相邻指的是两个区域有公共边，仅仅有公共点的两个区域不算做相邻。
- 小可可准备帮助Sealock规划一下如何选择种植区域，才能使得实际可种植面积总和最大。

构图

- 将试验田转化为点、并连接相邻的试验田后可以发现，我们得到的是一个二分图。通过对原图的黑白染色，可以把其中的一部分称为白点、另一部分称为黑点。由二分图建立网络：加入源点和汇点，从源点向每个白点引一条边，容量为白点对应试验田的面积；从每个黑点向汇点引边，容量为该黑点的对应面积。最后将相邻点之间的边改为网络中的边，由白点指向黑点，容量为正无穷。通过求网络最大流得到它的最小割，即为最优方案。

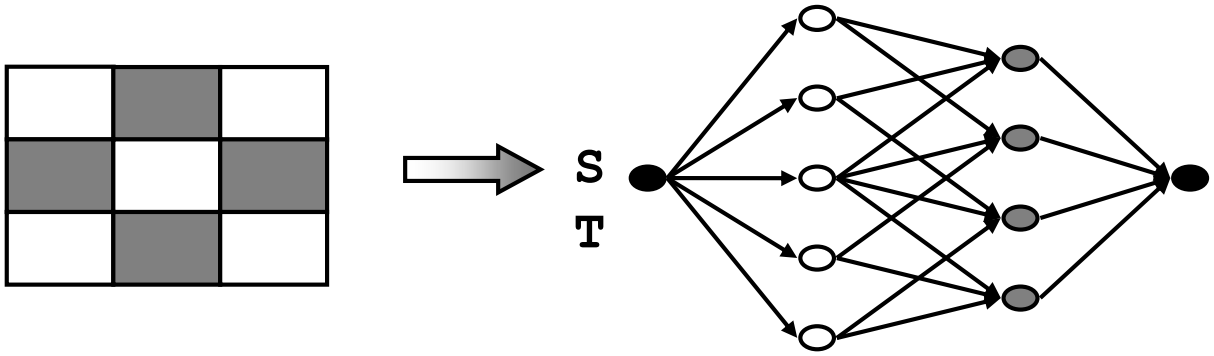


图1　建立网络

和平委员会（SPO）

- 要选一个委员会，但是出现了一个问题，某些代表是有矛盾的，不能同时选到委员会里来。现在有n个政党，每个政党只有两个代表，要从每个政党中选择一个代表，使委员会里的人都是友好的。所有的人用1到2n来编号，$2i-1$ 和 $2i$属于同一个政党。
- 读入政党总数，以及不友好的人的对数。
- 计算是否可以建立委员会。如果可以，给出方案。
- 输入：第一行有两个整数n和m，$1<=n<=8000$，$0<=m<=20000$。分别表示政党的总数以及不友好的关系数。接下来的m行，每行整数a 和 b, $1<=a<b<=2n$，表示这两个人是有矛盾的。
- 输出：无解则输出NIE。否则打印n行，从小到大输出你的方案中各人的编号。任意可行解都是可以的。

分析：

- 原题可描述为：

  有n个组，第i个组里有两个节点A_i, A_i'。需要从每个组中选出一个。而某些点不可以同时选出（称之为不相容）。任务是保证选出的n个点都能两两相容。

  - （在这里把A_i, A_i'的定义稍稍放宽一些，它们同时表示属于同一个组的两个节点。也就是说，如果我们描述A_i，那么描述这个组的另一个节点就可以用A_i'）

初步构图

- 如果$\mathcal{A}_i$与$\mathcal{A}_j$不相容，那么如果选择了$\mathcal{A}_i$，必须选择$\mathcal{A}_j$‘；同样，如果选择了$\mathcal{A}_j$，就必须选择$\mathcal{A}_i$’。

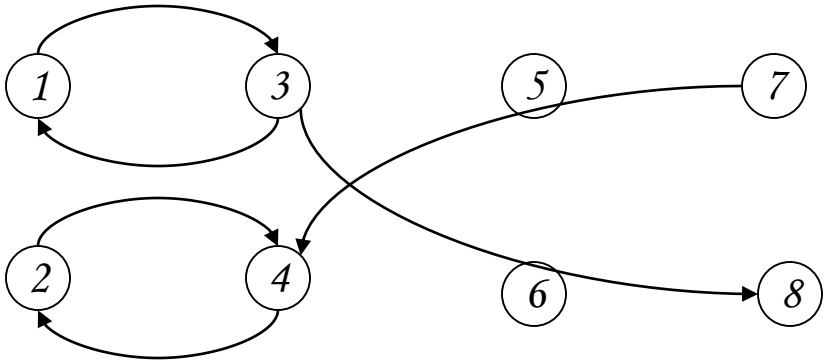
  $\mathcal{A}_i \longrightarrow \mathcal{A}_j'$

  $\mathcal{A}_j \longrightarrow \mathcal{A}_i'$

  这样的两条边**对称**

- 我们从一个例子来看：

- 假设*4*个组，不和的代表为：*1*和*4*，*2*和*3*，*7*和*3*，那么构图：



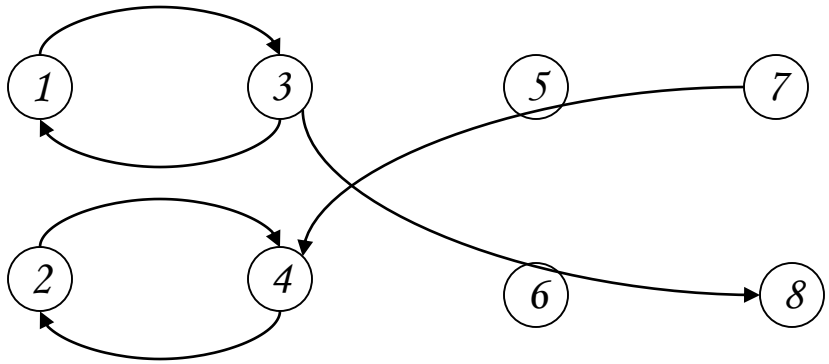
假设：

首先选*1*

→*3*必须选，*2*不可选

→*8*必须选，*4*、*7*不可选

5、*6*可以任选一个



- **矛盾**的情况为：

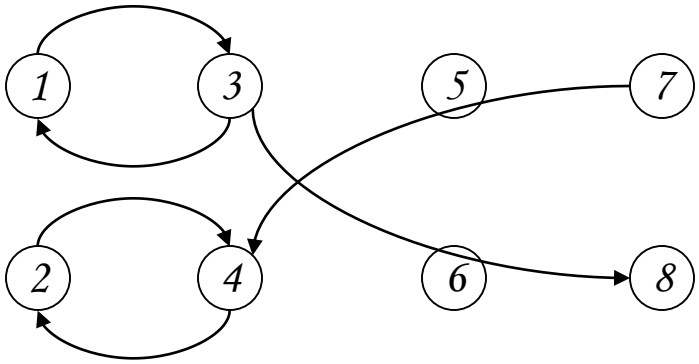
存在$\mathcal{A}_i$，使得$\mathcal{A}_i$既必须被选又不可选。

- 得到**算法*1***：
- 枚举每一对尚未确定的$\mathcal{A}_i, \mathcal{A}_i'$，任选*1*个，推导出相关的组，若不矛盾，则可选择；否则选另*1*个，同样推导。若矛盾，问题必定无解。

- 此算法正确性简要说明：
- 由于$\mathcal{A}_i$,$\mathcal{A}_i'$都是尚未确定的，它们不与之前的组相关联，前面的选择不会影响$\mathcal{A}_i$, $\mathcal{A}_i'$。

- 算法的时间复杂度在最坏的情况下为$O(nm)$。
- 在这个算法中，并没有很好的利用图中边的**对称**性

- 先看这样一个结构：



此图中*1*和*3*构成一个**环**，这样*1*和*3*要么都被选择，要么都不被选。

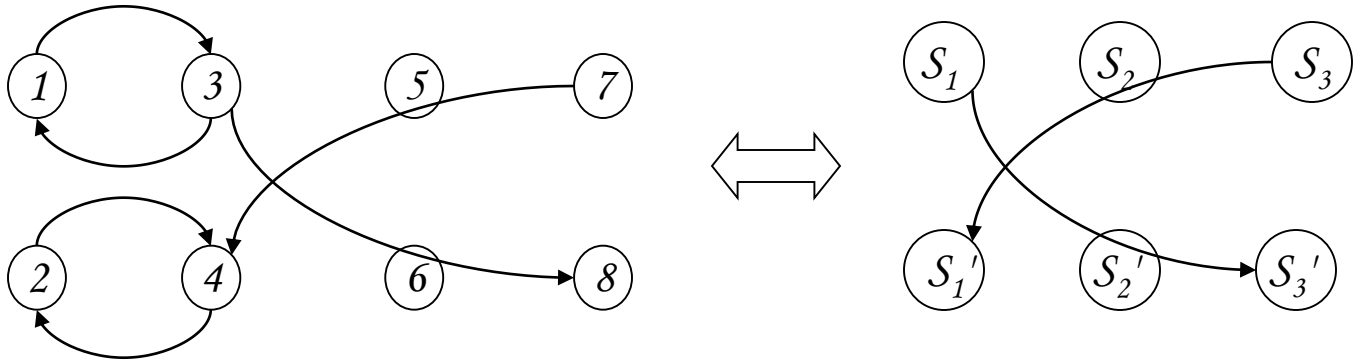
*2*和*4*同样如此。

- 更一般的说：
- 在每个一个环里，任意一个点的选择代表将要选择此环里的每一个点。不妨把环收缩成一个子节点（规定这样的环是**极大强连通子图**）。新节点的选择表示选择这个节点所对应的环中的每一个节点。

图的收缩

- 对于原图中的每条边$A_i \longrightarrow A_j$（设A_i属于环S_i，A_j属于环S_j）如果$S_i \neq S_j$，则在新图中连边：

$S_i \longrightarrow S_j$



- 这样构造出一个新的**有向无环图**。
- 此图与原图**等价**。

图的收缩

算法2

- 通过求强连通分量，可以把图转换成新的有向无环图，在这个基础上，介绍一个新的算法。

- 新算法中，如果存在一对A_i, A_i'属于同一个环，则判无解，否则将采用拓扑排序，以自底向上的顺序进行推导，一定能找到可行解。

算法2的流程：

- *1．构图*
- *2．求图的极大强连通子图*
- *3．把每个子图收缩成单个节点，根据原图关系构造一个有向无环图*
- *4．判断是否有解，无解则输出（退出）*
- *5．对新图进行拓扑排序*
- *6．自底向上进行选择、删除*
- *7．输出*

瘦陀陀和胖陀陀

- 一场可怕的战争后，瘦陀陀和他的好朋友胖陀陀将要凯旋。
  - 瘦陀陀处在城市***A***
  - 胖陀陀处在另外一个未知的城市
  - 他们打算选一个城市***X***（这个由瘦陀陀来决定）
  - 胖陀陀会赶在瘦陀陀之前到达城市***X***
  - 然后等待瘦陀陀也赶到城市***X***与他汇合，并举办一次庆祝宴会（由瘦陀陀请客）
  - 接着一起回到他们的家乡城市***B***
  - 由于胖陀陀嘴馋，他要求举办宴会的城市必须是瘦陀陀回家的路线中举办宴会最贵的一个城市。

一个例子（续）

- 瘦陀陀正专注地看回家的地图
  - 地图上标有n（$n \leq 200$）个城市和某些城市间直达的道路
  - 以及每条道路的过路费
  - 瘦陀陀还知道在每一座城市举办宴会的花费。
- 给出地图和A、B的位置
  - 请你告诉瘦陀陀回家的最小费用
  - 你的程序会接收到多次询问
  - 即对于每对城市$(c1,c2)$，你的程序应该立刻给出瘦陀陀从$c1$到$c2$的最小花费。

分析

- 胖陀陀规定必须在最贵的城市举办宴会
  - 因此不能简单地选择一条最短路走
  - 若路上有一个花费特别贵的城市…
- 对于每个点X，如果在那里办宴会…
  - 如何求最短路？
  - 多个询问怎么处理？
    - floyd计算每两点的距离？
    - SSSP就可以胜任吗？
      - AB = AX + XB…

树网的核

- 给出一棵无根树，边上有权。称树的最长路径为直径，定义路径的偏心距为：点到路径的上的点的最小值的最大值，给出一个s，找出直径上的某段长度不超过s的路径，使得偏心距最小。

分析

- 考虑到树的性质，对于任意两点，最短路=联通路=最长路。
  首先用floyd算法求出任意两点之间最短路。同时可以求出最长路径上都有哪些点。由于这是一棵树，最短路必然唯一。设mid[a,b]是a,b之间的联通路上的一个中间点。考虑问题的解，构造一个函数F(k,a,b)为K到ab间的最短路的长度。则
- f(k,a,b)=min{d[k,mid[a,b],f[k,a,mid[a,b]],f[k,mid[a,b],b]}
- 写出了这个方程，便不难得出一个三次方的算法。在实际做的时候，可以把k放在最外层枚举，这样内层实际上只用到了f的后面2维，用2维数组记录即可。

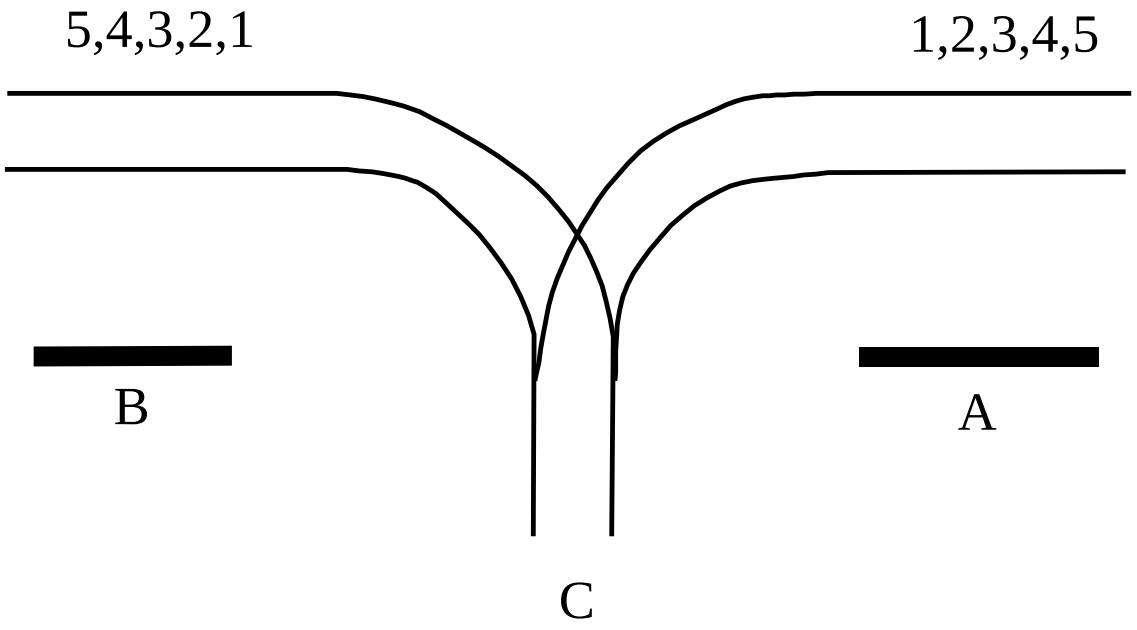
双栈排序

- 有两个队列和两个栈,分别命名为队列1(q),队列2(q2),栈1(s1)和栈2(s2).最初的时候,q2,s1和s2都为空,而q中有n个数(n<=1000),为1~n的某个排列.
  现在支持如下四种操作:
  - a操作,将 q的首元素提取出并加入s1的栈顶.
  - b操作,将s1的栈顶元素弹出并加入q2的队列尾.
  - c操作,将 q的首元素提取出并加入s2的栈顶.
  - d操作,将s2的栈顶元素弹出并加入q2的队列尾.
- 请判断,是否可以经过一系列操作之后,使得q2中依次存储着1,2,3,...,n.如果可以,求出字典序最小的一个操作序列.

考虑单栈

例1：1,2,3,4,5 yes例2：5,4,3,2,1 yes;

例3：3,2,4,5,1 yes;例4：3,1,4,5,2 no;



定理

- 定理：对于任意两个数q[i]和q[j]来说,它们不能压入同一个栈中的充要条件p是:存在一个k,使得i<j<k且q[k]<q[i]<q[j].
- 充分性：即如果满足条件p,那么这两个数一定不能压入同一个栈.这个结论很显然,使用反证法可证.
  假设这两个数压入了同一个栈,那么在压入q[k]的时候栈内情况如下:
  ...q[i]...q[j]...
  因为q [k]比q [i]和q [j]都小,所以很显然,当q[k]没有被弹出的时候,另外两个数也都不能被弹出
  而之后,无论其它的数字在什么时候被弹出,q [j]总是会在q [i]之前弹出.而q [j]>q [i],这显然是不正确的.

证明

- 必要性：也就是,如果两个数不可以压入同一个栈,那么它们一定满足条件p.这里我们来证明它的逆否命题,也就是"如果不满足条件p,那么这两个数一定可以压入同一个栈."
- 不满足条件p有两种情况:一种是对于任意i<j<k且q[i]<q[j],q[k]>q[i];另一种是对于任意i<j,q[i]>q[j].
- 第一种情况下,很显然,在q[k]被压入栈的时候,q[i]已经被弹出栈.那么,q[k]不会对q[j]产生任何影响(这里可能有点乱,因为看起来,当q[j]<q[K]的时候,是会有影响的,但实际上,这还需要另一个数R,满足J<K<R且 q[r]<q[j]<q[k],也就是证明充分性的时候所说的情况...而事实上我们现在并不考虑这个r,所以说q[k]对q[j]没有影响).
- 第二种情况下,我们可以发现这其实就是一个降序序列,所以所有数字都可以压入同一个栈.
  这样,原命题的逆否命题得证,所以原命题得证.
- 此时,条件p为q[i]和q[j]不能压入同一个栈的充要条件也得证.

构图

- 这样,我们对所有的数对(i,j)满足1<=i<j<=n,检查是否存在i<j<k满足q[k]< q [i]<q [j].如果存在,那么在点i和点j之间连一条无向边,表示q [i]和q[j]不能压入同一个栈.
- 二分图的两部分看作两个栈,因为二分图的同一部分内不会出现任何连边,也就相当于不能压入同一个栈的所有结点都分到了两个栈中.
- 此时我们只考虑检查是否有解,所以只要O(n)检查出这个图是不是二分图,就可以得知是否有解.

深度优先搜索求解

- 检查有解的问题已经解决.接下来的问题是,如何找到字典序最小的解.
  实际上,可以发现,如果把二分图染成1和2两种颜色,那么结点染色为1对应当前结点被压入s1,为2对应被压入s2.为了字典序尽量小,我们希望让编号小的结点优先压入s1.
- 又发现二分图的不同连通分量之间的染色是互不影响的,所以可以每次选取一个未染色的编号最小的结点,将它染色为1并从它开始DFS染色,直到所有结点都被染色为止.这样,我们就得到了每个结点应该压入哪个栈中 。
- 还有一点小问题,就是如果对于数对(i,j),都去枚举检查是否存在k，使得q[k]<q[I]<> M

最优乘车（NOI97）

- 一名旅客最近到H城旅游，他很想去S公园游玩，但如果从他所在的饭店没有一路巴士可以直接到达S公园，则他可能要先乘某一路巴士坐几站，再下来换乘同一站台的另一路巴士，这样换乘几次后到达S公园。
- 现在用整数1，2，…… N 给H城的所有的巴士站编号，约定这名旅客所在饭店的巴士站编号为1，2，…… S，公园巴士站的编号为N。
- 写一个程序，帮助这名旅客寻找一个最优乘车方案，使他在从饭店乘车到S公园的过程中换车的次数最少。
- 输入N和M条公交线路信息
- 求最少换车的次数

模型的构建

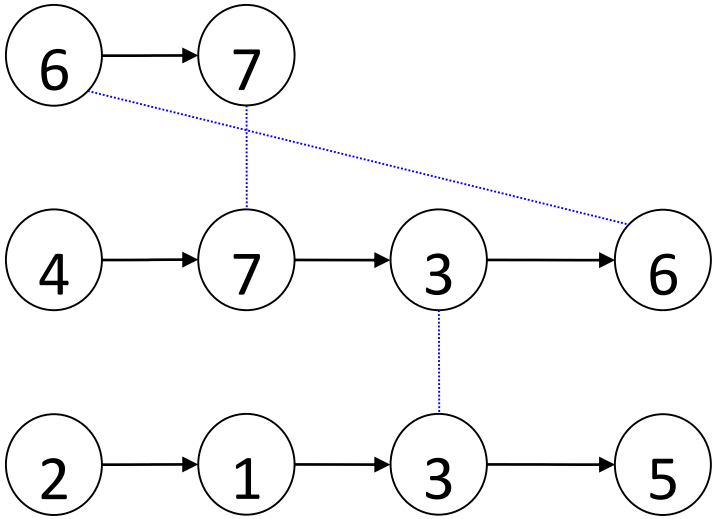
我们来分析样例

3 7

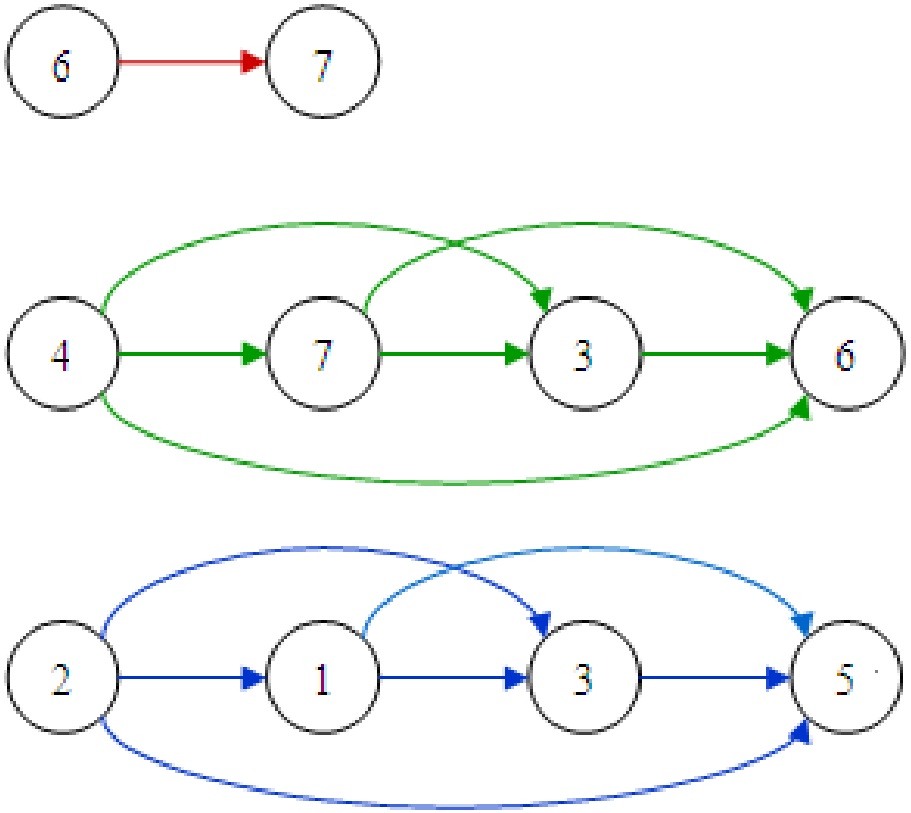
6 7

4 7 3 6

2 1 3 5



- 考察4 ——> 7 ——> 3 ——> 6这条线路。由于巴士在同一线路上行走不需换车，我们可设4 ——> 7，4 ——> 3，4 ——> 6，7 ——> 3，7 ——> 6，3 ——> 6这些边的权值都为1。对每条线路我们都这样构图，然后问题就转化成求起点1到终点n的最短路。



01串问题（NOI99试题）

- 给定7个整数$N,A_0,B_0,L_0,A_1,B_1,L_1$，要求设计一个01串$S=s_1s_2\ldots s_i\ldots s_N$，满足：

（1）$s_i=0$或$s_i=1$，$1<=i<=N$；

（2）对于S的任何连续的长度为L_0的子串$s_js_{j+1}\ldots s_{j+L0-1}(1<=j<=N-L_0+1)$，0的个数大于等于$A_0$且小于等于$B_0$；

（3）对于S的任何连续的长度为L_1的子串$s_js_{j+1}\ldots s_{j+L1-1}(1<=j<=N-L_1+1)$，1的个数大于等于$A_1$且小于等于$B_1$；

- 例如，$N=6,A_0=1,B_0=2,L_0=3,A_1=1,B_1=1,L_1=2$，则存在一个满足上述所有条件的01串S=010101。

输入：$N,A_0,B_0,L_0,A_1,B_1,L_1$（$3<=N<=1000$）

输出：一个满足所有条件的01串。

构图

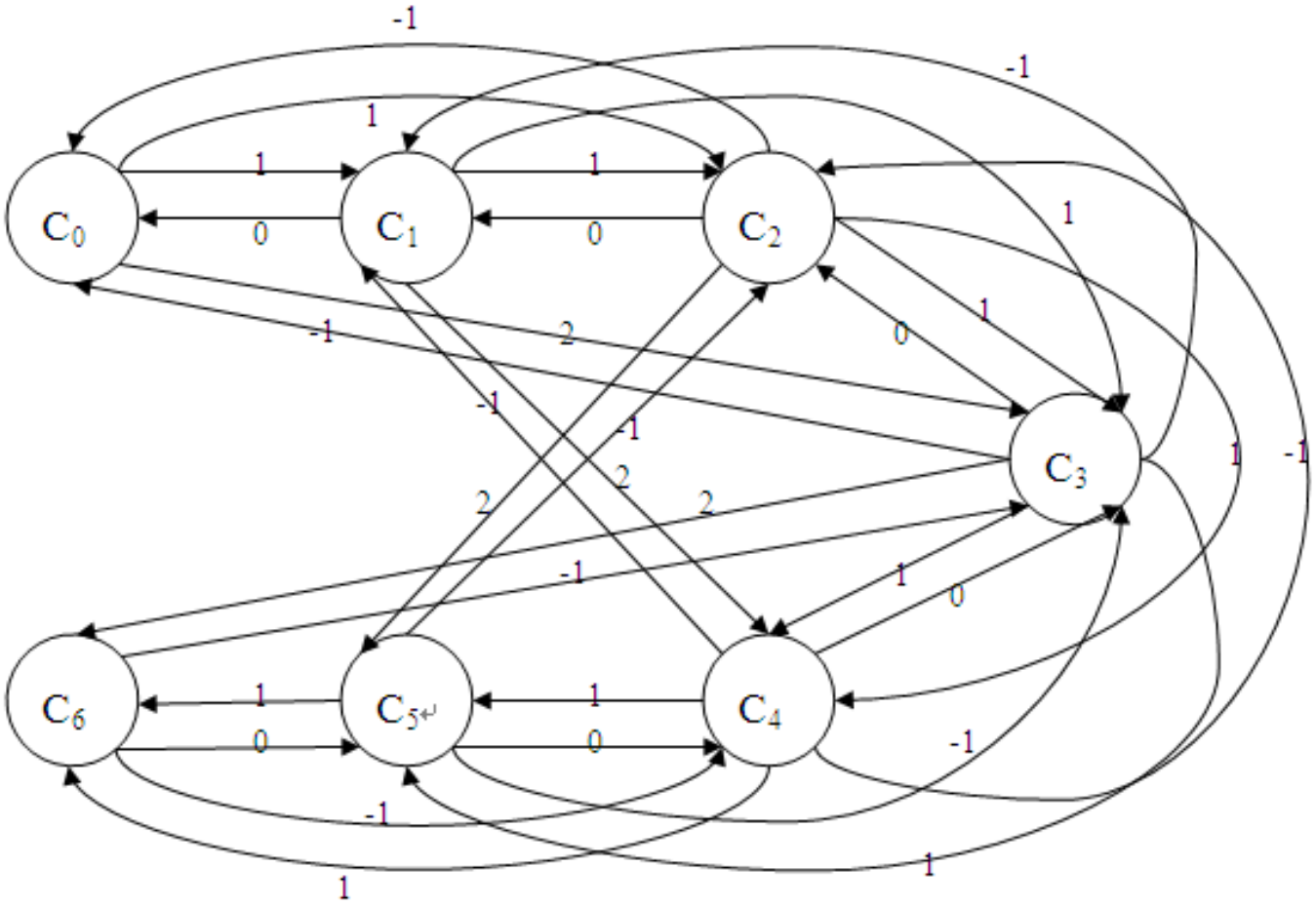
（1） 图的节点

用C_i表示$s_1,s_2....s_i$中1的个数，我们可以得到$C_1,C_2....C_n$，用他们作图的节点，将C_0也考虑进去（显然：$C_0=0$），我们就得到了N+1个节点。

（2）图的边与权。若我们已找到一个串S，则这个串S必须满足如下性质：对任意的i，C_i一定符合下面的不等式：

$$\begin{cases} C_i + 0 \geq C_{i-1} \\ C_{i-1} + 1 \geq C_i \\ A_1 \leq C_{i+L1} - C_i \leq B_1 \\ A_0 \leq L_0 - (C_{i+L0} - C_i) \leq B_0 \end{cases} \Rightarrow \begin{cases} C_i + 0 \geq C_{i-1} \\ C_i + 1 \geq C_{i+1} \\ C_i + (-A_1) \geq C_{i-L1} \\ C_i + B_1 \geq C_{i+L1} \\ C_i + (L_0 - A_0) \geq C_{i+L0} \\ C_i + (B_0 - L_{0)} \geq C_{i-L0} \end{cases}$$

- 样例构图
- 输入：**6 1 2 3 1 1 2**
- 输出：**010101**



模型求解

（1） 判断是否有解

- 考虑下面这样一种情况：$\begin{cases} C_i + x \geq C_j \\ C_j + y \geq C_k \Rightarrow C_i + (x+y+z) \geq C_i \\ C_k + z \geq C_i \end{cases}$
- 若**x+y+z < 0** 则 $C_i < C_i$
- 这显然是不可能得到的，所以若是出现了这种情况，则说明无解。这种情况就是构建的图中出现负权回路的情况！

（2）求出S序列

- 要求出S序列，我们只要求出C数组就可以了。因为C数组和S序列是一一对应的关系。而C数组的值，又可以通过构建的图论模型来求。图中C_0节点到各个节点的最短距离，就是各个节点的值。由于有负权，我们可以选择Bellman-Ford算法。