图论算法

【基本知识】

一．基本概念

在计算机科学与技术领域中，常常需要表示事物之间的关系。而客观事物之间的关系往往是千变万化，错综复杂的。图是用点和边来描述事物与事物之间的关系，是对客观实际问题的一种抽象，在奥赛中之所以用图来解决问题，是因为图能够把纷杂的信息变的有序、直观、清晰。

如果数据元素集合 D 中的各元素之间存在任意的前后件关系 R，则此数据结构 G=（D，R）称为图。如果将数据元素抽象为结点，元素之间的前后件关系用边表示，则图亦可以表示为 G=（V，E），其中 V 是结点的有穷（非空）集合，E 为边的集合。如果元素 a 是元素 b 的前件，这种前后件关系对应的边用(a，b)表示，即(a，b)∈E。

二．学习内容

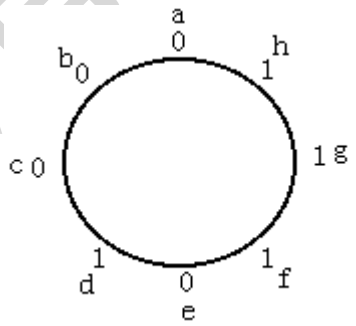
1. 图的表示法及存储结构
2. 图的遍历法
3. 图的基本算法
4. 图的应用

【常见试题分类解析】

一．排列方案

【例 1】

将 2n 个 0 和 2n 个 1 排成一圈。从任一位置开始，每次按逆时针方向以长度为 n+1 的单位数二进制数。要求给出一种排法，用上面方法产生出的 2n+1 个二进制数互不相同。当 n=2 时，有 4 个 0 和 4 个 1，排列如下图：



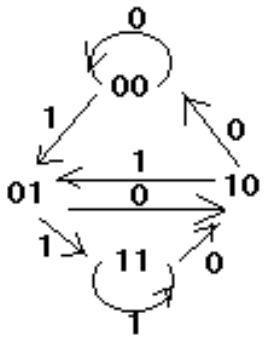
从位置 a 开始，逆时针方向取三个数 000；然后从位置 b 开始取三个数 001，接着取 010、101、011、111、110、100，共 8 个互不相同的二进制数。

【输入】n　(1≤n≤100)；

【输出】2n+1 位二进制数（排列方案不唯一，但一定能组成 2n+1 个互不相同的 n+1 位二进制数）

【分析】

以 n 位二进制数排列之间的重叠部分为点，以 01 排列为边，构成有向图。例如 n=2



从该图中找一条欧拉回路（不重复地遍历图中的每一条边），边上的 01 序列构成问题的解。设 vt[i]—十进制数 i 的访问标志（0≤i≤2n-1）。

$$\begin{cases} 0 & \text{十进制数}i\text{未访问，对应的二进制数尾添‘1’，即}i=(i*2+1)\bmod 2^n \\ 1 & \text{十进制数}i\text{未访问，对应的二进制数尾添‘0’，即}i=(i*2)\bmod 2^n \\ 2 & \text{十进制数}i\text{已访问，即欧拉回路已遍历} \end{cases}$$

首先 vt 清零，然后从 i=0 开始，按照上述规则计算，直至 vt[k]=2 为止。

```
var
 n，i：integer；
 mx，nw：longint；  {mx=2n；nw 为 n 位二进制数对应的十进制数}
 vt：array[0..10500] of byte；   {访问序列}
begin
 readln(n)；  {输入 0 和 1 的个数}
 mx：=1；                                                    {mx=2n }
 for i：=1 to n do mx：=mx*2；
 nw：=0；i：=0；                                             {从 0 开始访问}
 while vt[nw]<2 do     {若十进制数 nw  未访问，则增加一条边}
 begin
  inc(vt[nw])；
  if vt[nw]=1 {增加一条权为 1 的边}
   then begin write(1)；  nw：=(nw*2+1)mod mx；  end
   else begin   write(0)；  nw：=(nw*2)mod mx；  end；  {增加一条权为 0 的边}
  i：=(i+1)mod 70；          {每行 01 数的上限为 70}
  if i=0 then writeln；
 end；{while}
end.{main}
```

二．无向图的传递闭包

【例 2】连通性问题

输入一张无向图，指出该图中哪些顶点对之间有路。

【输入】

n　(顶点数，1≤n≤20)

e　(边数 1≤e≤210)

以下 e 行，每行为有边连接的一对顶点

【输出】

k 行，每行两个数，为存在通路的顶点对序号 i、j(i<j)

【分析】

var

link，longlink：array[1..20，1..20] of boolean；{无向图和无向图的传递闭包。其中 $longlink[i,j]=\begin{cases} true & i至j有路 \\ false & i至j无路 \end{cases}$ }

我们递推产生 longlink(0)、longlink(1)、…longlink(n)。在递推过程中，路径长度的’+’运算和比较大小的运算用相应的逻辑运算符’and’和’or’代替。对于 i，j 和 k=1‥n，如果图中结点 i 至结点 j 间存在通路且通路上所有结点的序号均属于{1‥k}，则定义 longlinkij(k)=1；否则 longlinkij(k)=0。

longlinkij(k)=longlinkij(k-1)or(longlinkik(k-1)andlonglinkkj(k-1))

传递闭包 longlink 的计算过程如下：

```
longlink←link;
for k←1 to n do   {枚举中间顶点}
 for i←1 to n do{枚举所有顶点对}
   for j←1 to n do {计算顶点 i 和顶点 j 的连通情况}
     longlink[i，j]←longlink[i，j]or(longlink[i，k]andlonglink[k，j]);
```

主程序

```
fillchar(longlink，sizeof(longlink)，0)；{传递闭包初始化}
fillchar(link，sizeof(link)，0)；        {无向图初始化}
readln(n)；readln(e)；                   {读顶点数和边数}
for k←1 to e do                          {输入无向图的信息}
 begin
  readln(i，j)；link[i，j]←true；link[j，i]←true；
 end；{for}
计算传递闭包 longlink;
for i←1 to n-1 do                        {输出所有存在通路的顶点对}
 for j←i+1 to n do if longlink[i，j] then 输出 i 和 j；
```

三．最小生成树

【知识点】

对于一张图进行深度优先搜索或宽度优先搜索，可生成一棵深度优先搜索树或宽度优先搜索树。搜索的出发点不同，生成树的形态亦不同。在一张有权连通图中，如何寻找一棵各边权的总和为最小的生成树，就是本章节所要讨论的问题，即 prim 算法。

【例 3】机器蛇

在未来的某次战争中，我军计划了一次军事行动，目的是劫持敌人的航母。由于这个计划高度保密，你只知道你所负责的一部分：机器蛇的通信网络。计划中要将数百条机器蛇投放到航母的各个角落里。由于航母内部舱室、管线错综复杂，且大部分由金属构成，因此屏蔽效应十分强烈，况

且还要考虑敌人的大强度电子干扰，如何保持机器蛇间的联系，成了一大难题。每条机器蛇的战斗位置由作战计划部门制定，将会及时通知你。每条机器蛇上都带有接收、发射系统，可以同时与多条机器蛇通讯。由于整个系统承载的数据量庞大，需要一个固定的通讯网络。情报部门提供了极其详尽的敌方航母图纸，使你对什么地方有屏蔽了如指掌。

请你设计一个程序，根据以上信息构造通讯网络，要求信息可以在任意两条机器蛇间传递，同时为了避免干扰，通讯网络的总长度要尽可能的短。

【输入】

输入数据的第一行是一个整数 n（n≤200）表示参战的机器蛇总数。

以下 n 行，每行两个整数 xi，yi，为第 i 支机器蛇的战斗位置。

接下来一行是一个整数 m（m≤100）表示航母内部可能产生屏蔽的位置。

最后 m 行，每行四个整数 ai，bi，ci，di，表示线段(ai，bi)-(ci，di)处可能有屏蔽，也就是说通讯网络不能跨越这条线段。

【输出】

输出数据应仅包括一个实数，表示建立的通讯网的最短长度，保留 3 位小数。

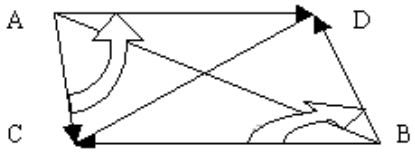
如果不能成功建立通讯网，请输出-1.000。

【分析】

题目中要求信息可以在任意两条机器蛇间传递、通讯网络的总长度要尽可能的短，显然这是一个求图的最小生成树问题。这道题在构造图的过程中还涉及到一点计算几何的知识。

1、判断线段相交

两条线段 AB、CD，相交的充要条件是：A、B 在直线 CD 的异侧且 C、D 在直线 AB 的异侧。也就是说从 AC 到 AD 的方向与从 BC 到 BD 的方向不同，从 CA 到 CB 的方向也与从 DA 到 DB 的方向不同。



这可以通过矢量的叉积来判断：$(\overline{AC}*\overline{AD})*(\overline{BC}*\overline{BD})<0$且$(\overline{CA}*\overline{CB})*(\overline{DA}*\overline{DB})<0$

其中用坐标计算矢量叉积的公式是 $\overline{OP_1}*\overline{OP_2}=x_1*y_2-x_2*y_1$

2、套用最小生成树的经典算法求解

以机器蛇为顶点，以不受屏蔽的通信线路为边构建图，就可以直接套用最小生成树的经典算法求解。由于几乎每两条机器蛇间都会有一条边，因此应选用 Prim 算法。

设

```
const
  maxn=200 ；  oo=2000000000；{ 机器蛇数的上限和无穷大}
type
  TPoint=record     {坐标}
    x，y：longint；
    end；
var
  s，w1，w2：array[1..maxn] of TPoint；  { 机器蛇的坐标和屏蔽线的坐标 }
```

```
    n，m，i，j，k：integer；
    ba：array[1..maxn] of boolean；  { 机器蛇的访问标志}
    d：array[1..maxn] of longint；  {d[i]以机器蛇 i 为头的最短边长,即 i 不在生成树,d[i]为 i 点与生成树连接的最短边长}
    min：longint；
    ans：double；
  function cp(p1，p2，p：TPoint)：integer；  { 计算矢量 PP1*PP2 }
  var
    v：longint；
  begin
    v：=(p1.x-p.x)*(p2.y-p.y)-(p1.y-p.y)*(p2.x-p.x)；
    if v=0 then cp：=0 else if v>0 then cp：=1 else cp：=-1；
  end；{cp}
  function dist(a，b：integer)：longint；{ 计算第 a 条机器蛇和第 b 条机器蛇间的距离，若 ab 之间有屏蔽，则距离设为无穷大 }
  var
    i：integer；
  begin
    dist：=oo；
    for i：=1 to m do { 如果 a 到 b 穿过第 i 个屏蔽，则返回无穷大 }
      if (cp(w1[i]，w2[i]，s[a])*cp(w1[i]，w2[i]，s[b])=-1) and
         (cp(s[a]，s[b]，w1[i])*cp(s[a]，s[b]，w2[i])=-1) then exit；
    dist：=sqr(s[a].x-s[b].x)+sqr(s[a].y-s[b].y)；
  end；{ dist }
  begin
    read(n)；{ 读入数据 }
    for i：=1 to n do with s[i] do read(x，y)；
    read(m)；
    for i：=1 to m do read(w1[i].x，w1[i].y，w2[i].x，w2[i].y)；
  {用 Prim 算法求最小生成树 }
    fillchar(ba，sizeof(ba)，0)；  {所有机器蛇未访问}
    for i：=2 to n do d[i]：=oo；  {最短边长序列初始化}
    d[1]：=0 ；ans：=0；  {从机器蛇 1 出发,通信网的最短长度初始化}
     for i：=1 to n do begin {访问 n 条机器蛇}
     min：=oo；{在所有未访问的机器蛇中寻找与已访问的机器蛇相连且具有最短边长的机器蛇 k}
       for j：=1 to n do
        if not ba[j] and (d[j]<min) then begin k：=j；  min：=d[j]；end；{then}
    if min=oo then begin ans：=-1；  break；  end；{then}{若这样的机器蛇不存在，则无解退出}
    ans：=ans+sqrt(min)；  ba[k]：=true；  {最短边长计入通信网,机器蛇 k 已访问}
    for j：=1 to n do {机器蛇 k 出发的所有不受屏蔽的边中，寻找边长最短的（k，j）}
      begin min：=dist(k，j)；  if min<d[j] then d[j]：=min；  end；{for}
  end；{for}
   writeln(ans：0：3)；  {输出通信网的最短长度}
```

end.{main}

四．单源最短路问题

【知识点】

所谓单源是指一个出发顶点，单源最短路问题指的是该顶点至所有可达顶点的最短路径问题，即 Dijkstra 算法。

【例 4】设计公共汽车线路

现有一张县城的城镇地图，图中的顶点为城镇，无向边代表两个城镇间的连通关系，边上的权为公路造价，县城所在的城镇为 v0。由于该县的经济比较落后，因此公路建设只能从县城开始规划。规划的要求是所有可达县城的城镇必须建设一条通往县城的汽车线路，该线路的工程的总造价必须最少。

【输入】

n　(城市数，1≤n≤20)

县城所在的城镇序号 v0

e　(有向边数 1≤e≤210)

以下 e 行，每行为 3 个整数，两个城镇的序号和它们间的公路造价

【输出】

k 行，每行为一条通往县城的汽车线路的总造价和该条线路途径的城镇序号

【分析】

设 G=（V，E）是一个有向图，它的每一条边（U，V）∈E 都有一个权 W（U，V），在 G 中指定一个结点 V0，要求把从 V0 到 G 的每一个结点 Vj(V J ∈ V)的最短有向路找出来（或者指出不存在从 V0 到 Vj 的有向路，即 V0 不可达 Vj）。这个问题即为单源最短路问题。解决单源最短路径的基本思想是把图中所有结点分为两组，每一个结点对应一个距离值

第一组：包括已确定最短路径的结点，结点对应的距离值是由 v0 到此结点的最短路径长度；

第二组：包括尚未确定最短路径的结点，结点对应的距离值是 v0 经由第一组结点（中间结点）至此结点的最短路径长度；

我们按最短路径长度递增的顺序把第二组的结点加到第一组中去，直至 v0 可达的所有结点都包含于第一组。在这个过程中，总保持从 v0 到第一组各结点的最短路径长度都不大于从 v0 至第二组任何结点的路径长度。

初始时 v0 进入第一组，v0 的距离值为 0；第二组包含其它所有结点，这些结点对应的距离值这样确定（设 vi 为第二组中的结点）

然后每次从第二组的结点中选一个其距离值为最小的结点 vm 加到第一组中。每往第一组加入一个结点 vm，就要对第二组的各结点的距离值作一次修正（设 vi 为第二组的结点）：

若加进 vm 做中间结点使得 v0 至 vi 的路径长度更短（即 vi 的距离值>vm 的距离值+Wmi），则要修改 vi 的距离（vi 的距离值←vm 的距离值+Wmi）。修改后再选距离值最小的一个结点加入到第一组中，…。如此进行下去，直至第一组囊括图的所有结点或再无可加入第一组的结点存在。显然，这种从第二组的结点中选距离值最小的结点扩展是一种贪心策略

设

n—图的结点数；

adj—有向图的相邻矩阵。其中

$$adj[i,i]=\begin{cases}0 & \text{结点}i\text{在第二组}\\ 1 & \text{结点}i\text{在第一组}\end{cases}$$

$$adj[i,j]=\begin{cases}w_{ij} & (i,j)\in E\\ \infty & (i,j)\notin E\end{cases}$$

(1≤i，j≤n，i≠j)

dist—最短路径集合。其中

dist[i]．pre—在 v0 至 vi 的最短路径上，vi 的前趋结点序号；
dist[i]．length—v0 至 vI 的最短路径长度，即 vi 的距离值；
1≤i≤n）

```
Const n=图的结点数；
  Type
    path=record                                  {路径集合的结点类型}
           length：real；                        {距离值}
            pre：0‥n；                           {前趋结点序号}
            end；
  var
      adj：array[1‥n，1‥n] of real               {相邻矩阵}
      dist：array[1‥n] of path；                 {路径集合}
```

计算单源最短路径的过程如下：

```
  fillchar(adj，sizeof(adj)，0)；{建立相邻矩阵 adj}
  for i←1 to n do
    for j←1 to n do
    if（i，j）∈E then adj[i，j]←wij
                          else adj[i，j]←∞；
    for i←1 to n do {路径集合初始化}
      begin
      dist[i]．length←adj[v0，i]；
      if dist[i]．length<>∞
        then dist[i]．pre←v0
      else dist[i]．pre←0；
      end；{for}
    adj[v0，v0]←1；{源结点 v0 进入第一组}
repeat
   min←∞；u←0；
   for i←1 to n do {从第二组中找距离值最小的结点 u}

if (adj[i，i]=0)and(dist[i]．length<min)
 then begin u←i；min←dist[i]．length；end；{then}
if u<>0  {第二组中存在一个距离值最小的结点}
 then begin
        adj[u，u]←1；{结点 u 进入第一组}
        for i←1 to n do  {修正第二组中 u 可达的结点距离值}
        if(adj[i，i]=0)and(dist[i]．length>dist[u]．length+adj[u，i])
            then begin
              dist[i]．length←dist[u]．length+adj[u，i]；
               dist[i]．pre←u；
```

```
                    end；{then}
                end；{then}
            until u=0；
```

算法结束时，沿着结点 vi 的 pre 指针回溯，便可确定 v0 至 vi 的最短路径：

```
        procedure print(i：integer)；
          begin
            if i=v0 then 输出结点 v0
                    else begin
                         print(dist[i]．pre)；
                         输出结点 i 和 v0 至 vi 的最短路径长度 dist[i]．length；
                         end；{else}
            end；{print}
```

显然递归调用 print[1]，…，print[n]后，可得知 v0 至所有结点的最短路径。由此得出主程序：

```
输入城镇数 n；
输入出发城镇序号 v0；
输入城镇间的距离矩阵 w；
计算单源最短路径方案 dist；
for i←1 to n do                                          {枚举除 v0 外的其它城镇}
begin
   if (i<>v0 )and(dist[i]．length<>∞){若城镇 i 与城镇 v0 间有通路，则输出它们之间的最短距离和路径方案}
then begin wrtiln(dist[i]．length)；print(i)；end；{then}
wrteln；
  end；{for}
```

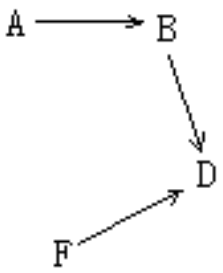
【注意】

在计算多源无权图的最短路时,与其采用 Dijkstra 算法(k*n2),不如采用宽度优先搜索(n2)。初始时，所有源点入队列。

五．拓扑序列

【例 5】士兵排队

有 n 个士兵(1≤n≤26)，编号依次为 A、B、C、……。队列训练时，指挥官要把一些士兵从高到矮依次排成一行。但现在指挥官不能直接获得每个人的身高信息，只能获得“p1 比 p2 高”这样的比较结果（p1，p2∈{ ’A’ ‥’Z’}），记为 p1>p2。例如 A>B，B>D，F>D。士兵的身高关系下图所示



对应的排队方案有三个：AFBD、FABD、ABFD。

输入 k 行，每行为 a b，表示 a>b

输出排队方案

【分析】

士兵的身高关系对应一张有向图，图中的顶点对应一个士兵，有向边<vi，vj>表示士兵 i 高于士兵 j。我们按照从高到矮将士兵排出一个线性的顺序关系，即为对有向图的顶点进行拓扑排序。

⑴拓扑序列的定义：拓扑排序是有向图的另一种重要运算。给出有向图 G=(V，E)。若顶点的线性序列 v1’，…，vn’(vi’∈V，1≤i≤n)满足如下条件：

vk’至 vk+1’有一条路径（ 1 ≤k<n-1）

则称该序列为拓扑序列。 一个有向图结点的拓扑序列不是唯一的。有环图不能拓扑排序。

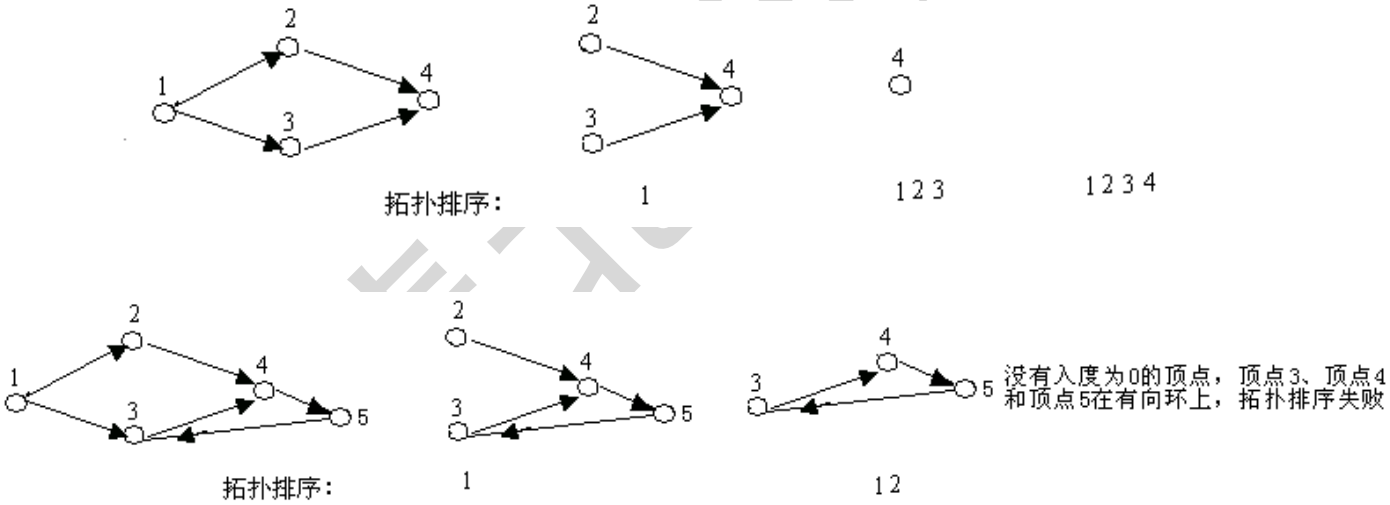
⑵拓扑序列的计算

下面给出拓扑排序的方法

⑴从图中选择一个入度为 0 的结点且输出之；

⑵从图中删除该结点及其所有出边（即与之相邻的所有结点的入度减一）；

反复执行这两个步骤，直至所有结点都输出了，也就是整个拓扑排序完成了。或者直至剩图中再没有入度为 0 的结点，这就说明此图中有环，拓扑排序不能再进行下去了。例如




```
var
        g：array[’A’‥’Z’，’A’‥’Z’] of 0‥1；  {图的相邻矩阵}
        d：array[’A’‥’Z’] of integer；{结点的度数序列}
        s：set of  ’A’‥’Z’；{士兵名集合}
```

图中的结点为士兵。若 a>b，则 g[a，b]←1 （即 a 向 b 引入一条有向边）；计算结点 b 的入度（即比士兵 b 高的人数）d(b)←d(b)+1。显然最高士兵对应的结点入度为 0：

```
        s←[]；{士兵名集合初始化}
        while    文件未输入完  do
         begin
            读 a>b 信息；
            if (a∈{ ’A’‥’Z’})and (b∈{ ’A’‥’Z’})
```

```
                then begin
                    s←s+[a，b]；{计算士兵名集合}
                    g[a，b]←1；{构造有向边（a，b）}
                    d[b]←d[b]+1；{累计结点 b 的入度}
                    end；{then}
            end；{while}
```

然后通过下述方法计算士兵名字符集 m 和士兵人数 k

```
m←’’；
for a= ’A’ to    ’Z’ do   if a∈s then m←m+a；
k←length(m)；
```

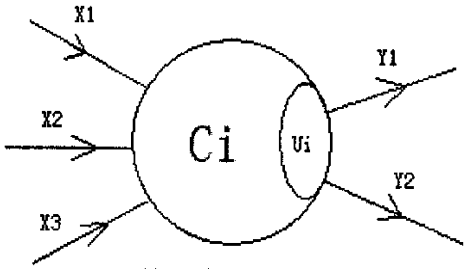
接下来对有向图G作拓扑排序。若图 G 中有回路，则排队方案不存在；否则拓扑序列 n 即为一个排队方案。拓扑排序的过程如下：

```
n←’’；    {拓扑序列 n 初始化为空}
for i：=1 to k do {依次搜索每一个出列士兵}
 begin
  j←1；
  while(d[m[j]]>0)and(j≤k)doj←j+1；{搜索第 i 个入度为 0 的结点序号 j}
  if j>k then {若入度为 0 的结点不存在，则队列不存在最高的士兵，无解退出}
   then begin   输出失败信息；halt；end；{then}
        n←n+m[j] {入度为 0 的结点 j 进入拓扑序列 n}
        a←m[j]；d[a]←∞；
        for j：=1 to k do{删去结点 j}
          if g [a，m[j]]>0 then d[m[j]]←d[m[j]]-1；
    end；{for}
输出拓扑序列 n；
```

六．神经网络

【例 6】

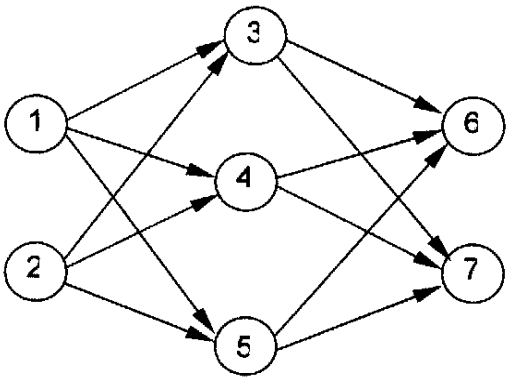
在兰兰的模型中，神经网络就是一张有向图，图中的节点称为神经元，而且两个神经元之间至多有一条边相连，下图是一个神经元的例子：



图中，X1—X3 是信息输入渠道，Y1－Y2 是信息输出渠道，C1 表示神经元目前的状态，Ui 是阈值，可视为神经元的一个内在参数。

神经元按一定的顺序排列，构成整个神经网络。在兰兰的模型之中，神经网络中的神经无分为几层；称为输入层、输出层，和若干个中间层。每层神经元只向下一层的神经元输出信息，只从上

一层神经元接受信息。



兰兰规定，Ci 服从公式：（其中 n 是网络中所有神经元的数目）

$$C_i = \sum_{(j,i)\in E} W_{ji}C_j - U_i$$

公式中的 Wji（可能为负值）表示连接 j 号神经元和 i 号神经元的边的权值。当 Ci 大于 0 时，该神经元处于兴奋状态，否则就处于平静状态。当神经元处于兴奋状态时，下一秒它会向其他神经元传送信号，信号的强度为 Ci。 如此．在输入层神经元被激发之后，整个网络系统就在信息传输的推动下进行运作。现在，给定一个神经网络，及当前输入层神经元的状态（Ci），要求你的程序运算出最后网络输出层的状态。

【输入格式】

第一行是两个整数 n（1≤n≤20）和 p。接下来 n 行，每行两个整数，第 i＋1 行是神经元 i 最初状态和其阈值（Ui），非输入层的神经元开始时状态必然为 0。再下面 P 行，每行由两个整数 i，j 及一个整数 Wij，表示连接神经元 i、j 的边权值为 Wij。

【输出格式】

输出包含若干行，每行有两个整数，分别对应一个神经元的编号，及其最后的状态，两个整数间以空格分隔。仅输出最后状态非零的输出层神经元状态，并且按照编号由小到大顺序输出！

若输出层的神经元最后状态均为 0，则输出 NULL。

【分析】

理解问题的第一步就是认真“读题”。那么我们先来看一看这个题目涉及的问题。研究一下题目中所给的图的一些性质，可以发现如下特点：

1．图中所有的节点都有一个确定的等级，我们记作 Level(i)

2．图中所有的边都是有向的，并且从 Level 值为 i-1 的节点指向 Level 值为 i 的节点

我们不妨将其抽象为“阶段图”。

更一般地，我们可以发现所有的阶段图都是有向无环的，这样我们可以通过拓扑排序来得到期望的处理节点的顺序。

由于阶段图的性质使得该图的所有边所连接节点的等级都是相邻的，因此就可以设计出一个基于宽度优先搜索（即 BFS）的算法：

1．初始时将所有输入层的节点放入队列；

2．取出队列中的一个元素，不重复地扩展并处理该节点所发出的边的目标节点；

3．如果队列非空，则转向 2；

4．输出输出层中所有满足条件的节点。

但是由于本题在问题描述中并没有明确的给出判断一个节点是否是输入节点，因此需要在算法进行的过程当中额外地考虑一些边界情况的数据（这个过程即便是真实数据没有这样出也是要有的），下面给出的更一般的算法可能会更好的跳过这些边界情况。

1．对原图中所有的节点进行一次拓扑排序；

2．按照拓扑顺序处理每一个节点；

3．输出输出层中所有满足条件的节点。

七．最大流问题

【知识点】

堆的定义；堆的操作；堆的性质；存储方式；删除最小值元素；向下调整；插入元素和向上调整；堆的建立；时间复杂度分析。

【例 7】餐巾问题

某软件公司正在规划一项 n 天的软件开发计划，根据开发计划第 i 天需要 ni 个软件开发人员，为了提高工作效率，公司给软件人员提供了很多的服务，其中一项服务就是要为每个开发人员每天提供一块消毒毛巾，这种毛巾使用一天后必须再做消毒处理后才能使用。消毒方式有两种，A 种方式的消毒时间需要 a 天时间，B 中方式的消毒需要 b 天时间（b>a），A 种消毒方式的费用为每块毛巾 fA，B 种消毒方式的费用为每块毛巾 fB，而买一块新毛巾的费用为 f（新毛巾是已消毒的，当天可以使用）；而且 f>fA>fB。公司经理正在规划在这 n 天中，每天买多少块新毛巾、每天送多少块毛巾进行 A 种消毒和每天送多少块毛巾进行 B 种消毒。当然，公司经理希望费用最低。

你的任务就是：求出提供毛巾服务的最低总费用。

【输入文件】第 1 行为 n, a, b, f, fA, fB.

第 2 行为 n1, n2, …, nn（注：1≤f, fA, fB≤60, 1≤n≤1000）

【输出文件】最少费用

【输入输出示例】

input.txt	output.txt
4 1 2 3 2 1 8 2 1 6	38

【分析】

公司第 i 天需要 ni 块毛巾，可以把这 ni 块毛巾的来源列举如下：

新买的毛巾。

第 i － a － 1 天之前通过 A 种方式消毒的毛巾。

第 i － b － 1 天之前通过 B 种方式消毒的毛巾。

我们构造带费用的网络流图 G = (V, E, C)。V = {S, V1, V2, …, Vn, V1’, V2’, …, Vn’, T}，其中 S 是源点、T 是汇点。E 中包含如下几类弧：

<S, Vi>（1≤i≤n），容量为 ni，费用为 0。表示第 i 天需要 ni 块毛巾。

<Vi, T>（1≤i≤n），容量为正无穷大，费用为 f。该弧的流量表示第 i 天通过方式 a 得到的毛巾数量。

<Vi, Vi-a-1’>（a+2≤i≤n），容量为正无穷大，费用为 fA。该弧的流量表示第 i 天通过方式 b 得到的毛巾数量。

<Vi, Vi-b-1’>（b+2≤i≤n），容量为正无穷大，费用为 fB。该弧的流量表示第 i 天通过方式 c 得到的毛巾数量。

<Vi’, Vi-1’>（2≤i≤n），容量为正无穷大，费用为 0。由于题目中没有说消毒完的毛巾要马

上使用，所以第 3 天就消毒好的毛巾可以暂且留着，到第 7 天、第 8 天再使用也可以。因此这里增设此弧。

<Vi’, T>（1≤i≤n），容量为 ni，费用为 0。

然后对这个网络流图以 S 为源点、T 为汇点的求最小费用最大流即可。求得的最大流费用就是原问题的答案。