南京信息工程大学 藕舫学院数学建模竞赛实验班

图论模型

主讲：费文龙

答疑：feiwl@126.com

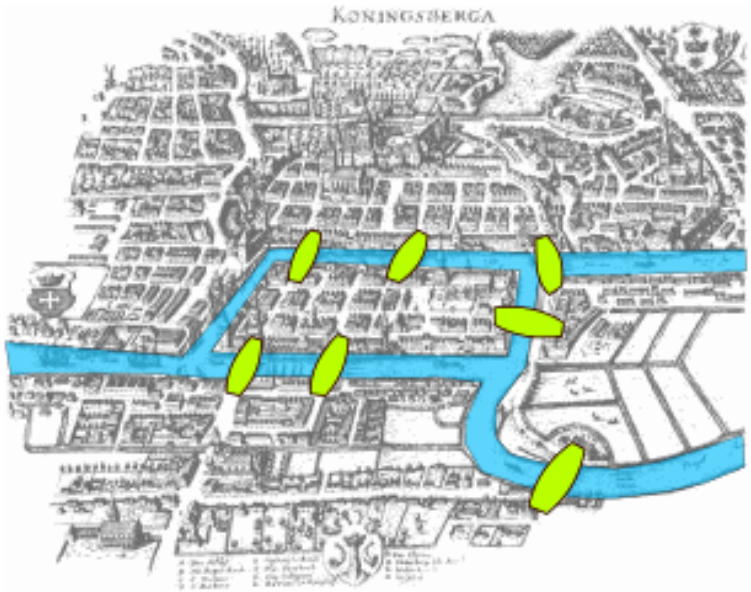
本学期数模培训总体要求

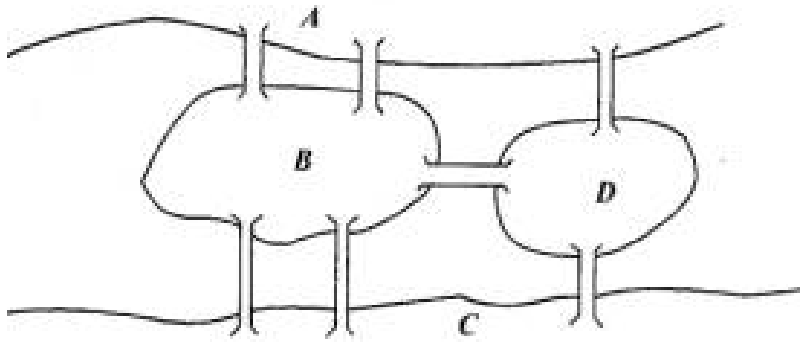
- 认真学习每一个模型和算法的基本思想；
- 通过对算法的学习，训练每一位同学的编程动手能力；
- 对培训课程中学过的各类算法，建立自己的程序库，尽可能构造通用函数，以便后续使用过程中调用；

图论模型

1. 图论基本概念
2. 最短路径算法
3. 最小生成树算法
4. 遍历性问题
5. 二分图与匹配
6. 网络流问题
7. 关键路径问题
8. 系统监控模型
9. 着色模型

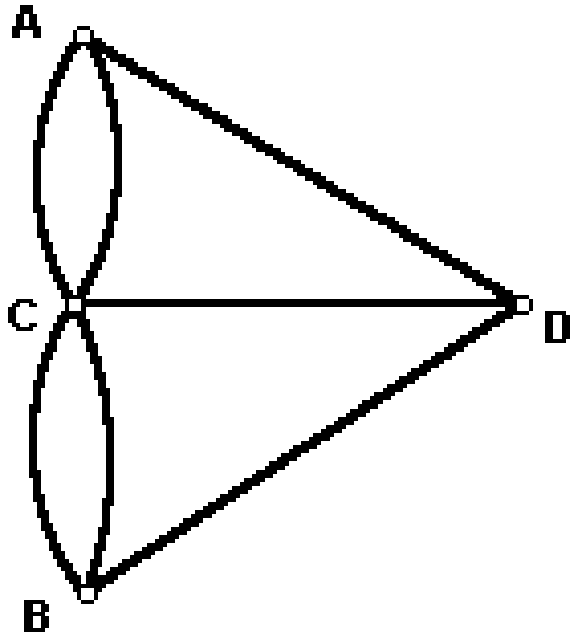
1、图论的基本概念





问题1(哥尼斯堡七桥问题)：

能否从任一陆地出发通过每座桥恰好一次而回到出发点？

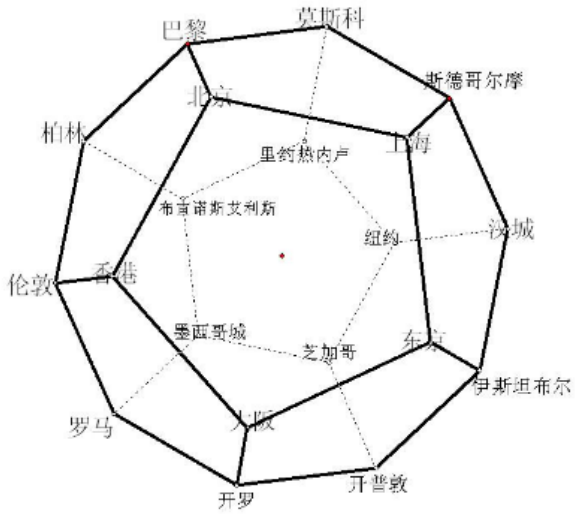


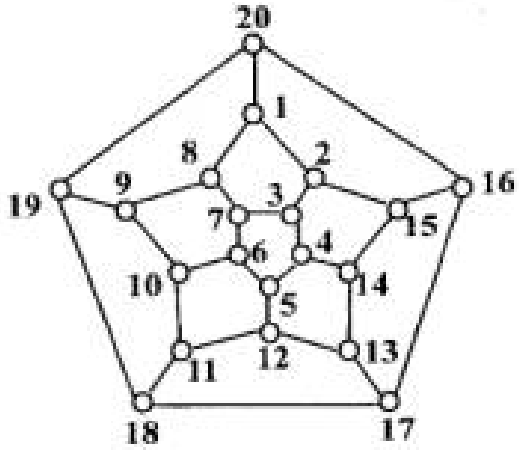
欧拉指出：

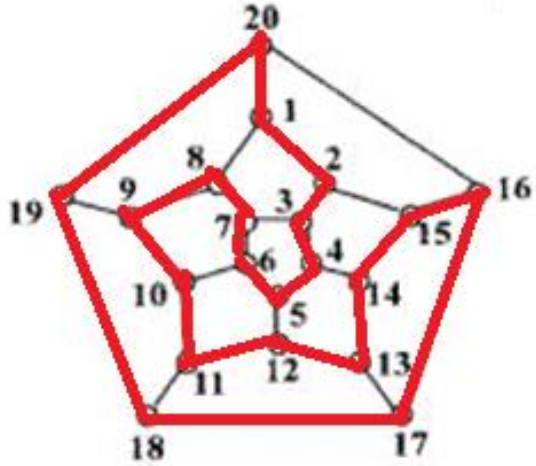
如果每块陆地所连接的桥都是偶数座，则从任一陆地出发，必能通过每座桥恰好一次而回到出发地.

问题2(哈密顿环球旅行问题):

十二面体的20个顶点代表世界上20个城市，能否从某个城市出发在十二面体上依次经过每个城市恰好一次最后回到出发点？







欧拉问题是“边遍历”，哈密尔顿问题是“点遍历”

问题3(四色问题):

对任何一张地图进行着色，两个共同边界的国家染不同的颜色，则只需要四种颜色就够了.

问题4(关键路径问题):

一项工程任务,大到建造一座大坝,一座体育中心,小至组装一台机床,一架电视机，都要包括许多工序.这些工序相互约束,只有在某些工序完成之后，一个工序才能开始．即它们之间存在完成的先后次序关系,一般认为这些关系是预知的，而且也能够预计完成每个工序所需要的时间.

这时工程领导人员迫切希望了解最少需要多少时间才能够完成整个工程项目，影响工程进度的要害工序是哪几个?

图的定义

图论中的“图”并不是通常意义下的几何图形或物体的形状图，而是以一种抽象的形式来表达一些确定的事物之间的联系的一个数学系统.

定义1 一个有序二元组(V, E)称为一个**图**，记为$G = (V, E)$，其中

① V称为G的顶点集，$V \neq \phi$，其元素称为**顶点**或**结点**，简称**点**；

② E称为G的边集，其元素称为**边**，它联结V中的两个点，如果这两个点是无序的，则称该边为**无向边**，否则，称为**有向边**.

如果$V = \{v_1, v_2, \cdots, v_n\}$是有限非空点集，则称$G$为**有限图**或**$n$阶图**.

如果E的每一条边都是无向边,则称G为**无向图**(如图1)；如果E的每一条边都是有向边,则称G为**有向图**(如图2)；否则,称G为**混合图**.

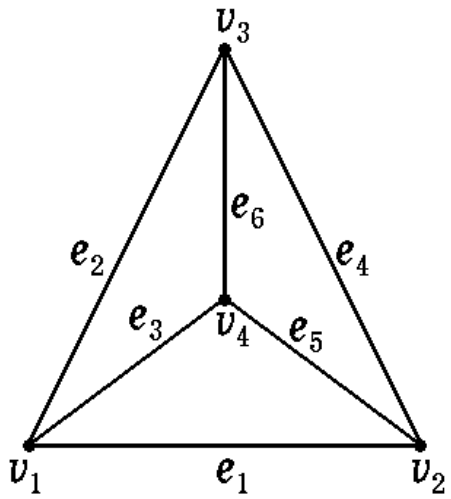


图1

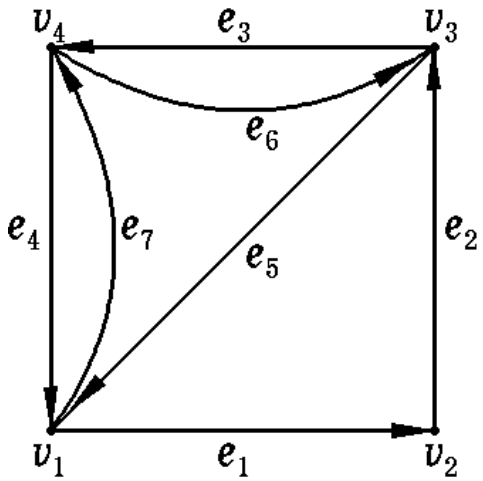


图2

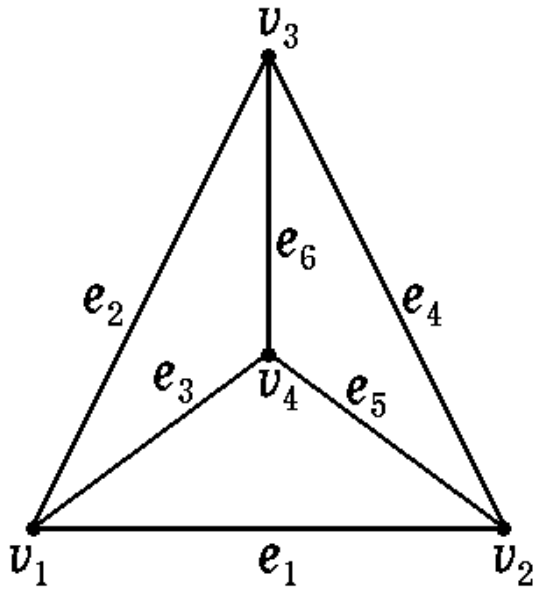
并且常记： $V=\{v_1, v_2, \cdots, v_n\}$, $|V|=n$；

$E=\{e_1, e_2, \cdots, e_m\}(e_k=v_iv_j)$, $|E|=m$.

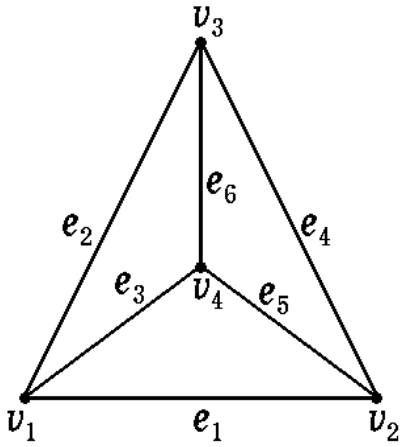
称点v_i, v_j为边v_iv_j的**端点**.在有向图中,称点v_i, v_j分别为边v_iv_j的**始点**和**终点**.该图称为**(n,m)图**

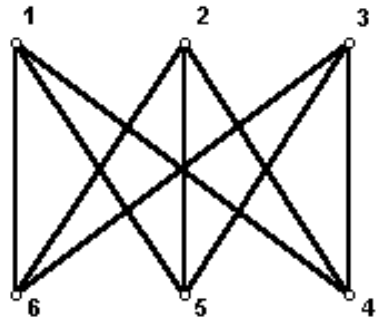
对于一个图$G=(V, E)$, 人们常用图形来表示它, 称其为**图解**. 凡是有向边, 在图解上都用箭头标明其方向.

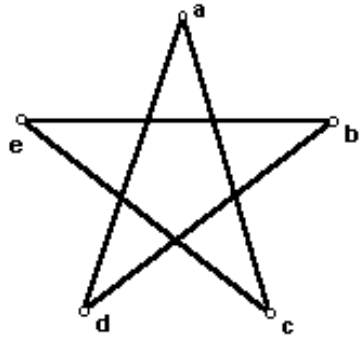
例如, 设$V=\{v_1, v_2, v_3, v_4\}$, $E=\{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4\}$, 则$G=(V, E)$是一个有4个顶点和6条边的图, G的图解如下图所示.

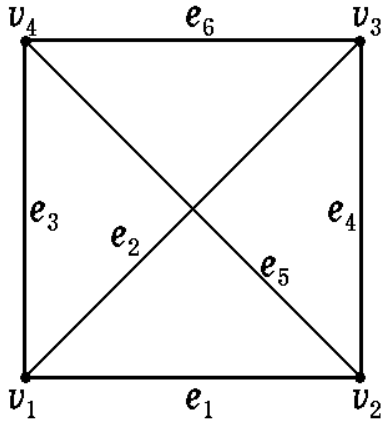


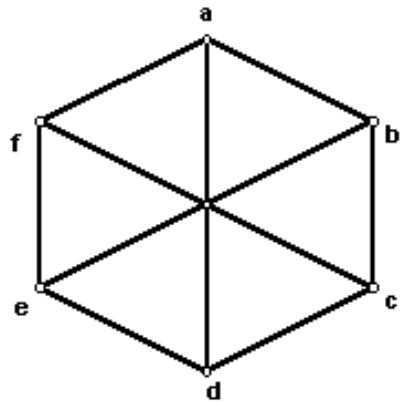
一个图会有许多外形不同的图解，下面两个图表示同一个图 $G = (V, E)$ 的图解. 这两个图互为同构图, 今后将不计较这种外形上的差别, 而用一个容易理解的、确定的图解去表示一个图.

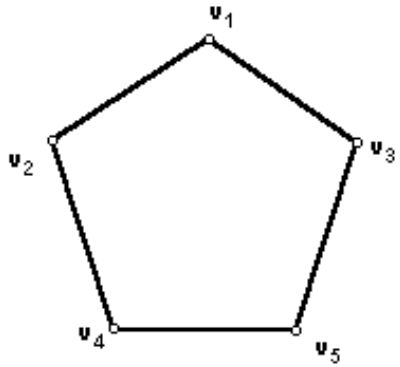






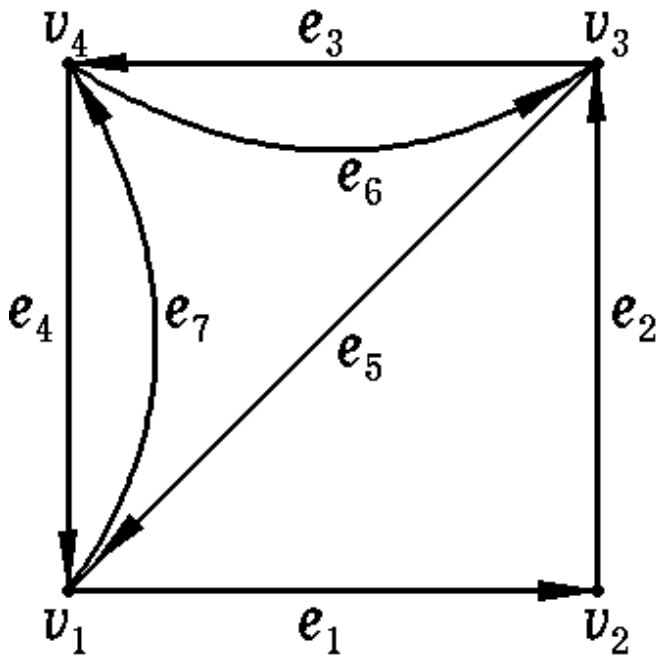






有边联结的两个点称为**相邻的点**,有一个公共端点的边称为**相邻边**.边和它的端点称为**互相关联**.常用$d(v)$表示图G中与顶点v关联的边的数目, $d(v)$称为顶点v的**度数**.对于有向图,还有**出度**和**入度**之分.

用$N(v)$表示图G中所有与顶点v相邻的顶点的集合.



$d(v_1)= d(v_3)= d(v_4)=4,\ d(v_2)=2$

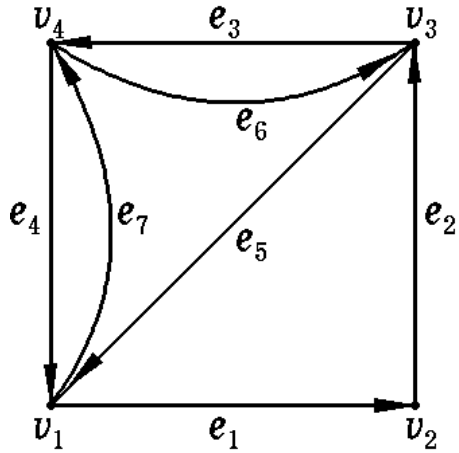
$d_{out}(v_1)= d_{out}(v_3)= d_{out}(v_4)=2,\ d_{out}(v_2)=1$

$d_{in}(v_1)= d_{in}(v_3)= d_{in}(v_4)=2,\ d_{in}(v_2)=1$

握手定理

- **握手定理**：无向图中，所有结点的度数之和等于$2m$。

$$\sum_{i=1}^{n} d(v_i) = 2m$$



- 右图：

$$\sum_{i=1}^{n} d(v_i) = 2*7 = 14$$

$d(v_1)= d(v_3)= d(v_4)=4,$
$d(v_2)=2$

- **推论1**：无向图中必有偶数个度数为奇数的结点。
- **推论2**：有向图中所有结点的出度之和等于入度之和。

我们今后只讨论有限简单图：

（1）顶点个数是有限的;

（2）任意一条边有且只有两个不同的点与它相互关联;

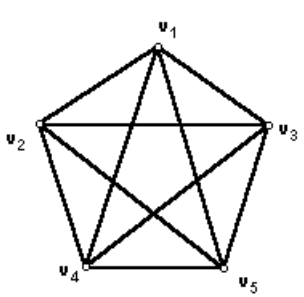
（3）若是无向图，则任意两个顶点最多只有一条边与之相联结;

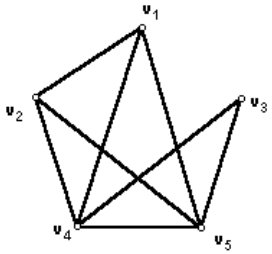
（4）若是有向图，则任意两个顶点最多只有两条边与之相联结. 当两个顶点有两条边与之相联结时，这两条边的方向相反.

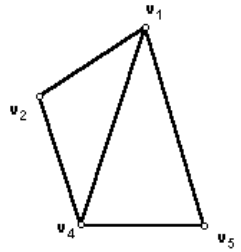
如果某个有限图不满足(2)(3)(4)，可在某条边上增设顶点使之满足.

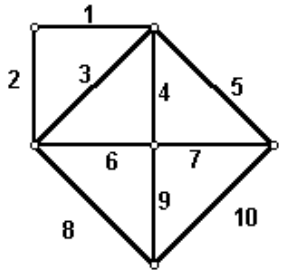
常用的图

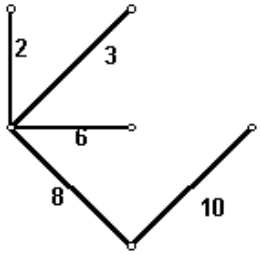
- 给定图$G=<V,E>$ 和 $G'=<V',E'>$ 是两个图，如果有 $V' \subseteq V$ 和 $E' \subseteq E$,则称图G'是图 G 的**子图。**若$V'=V$ 称图G'是图 G 的**生成子图；**

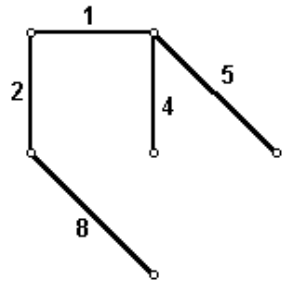








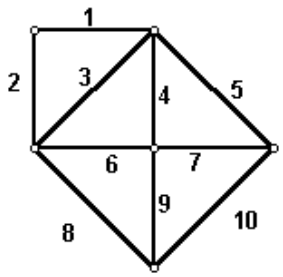


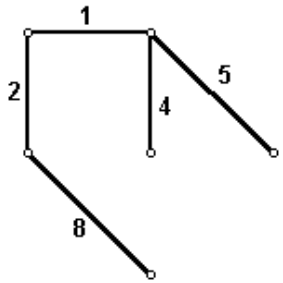


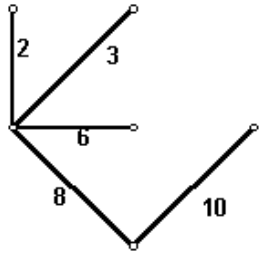
- 若将图G的每一条边e都对应一个实数$F(e)$，则称$F(e)$为该边的权，并称图G为**赋权图(网络),** 记为$G = <V, E, F>$

常用的图

- 任意两点均有通路的图称为**连通图。**
- 连通而无圈的图称为**树，**常用$T=<V,E>$表示树。
- 若图G'是图 G 的生成子图，且G'又是一棵树，则称G'是图G 的**生成树。**

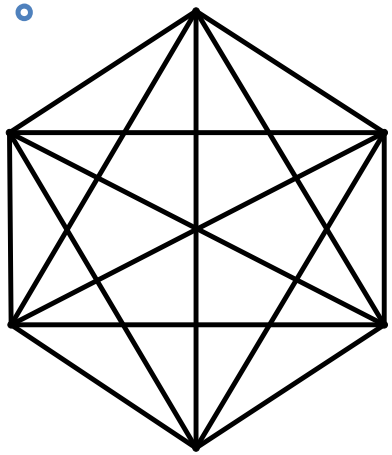


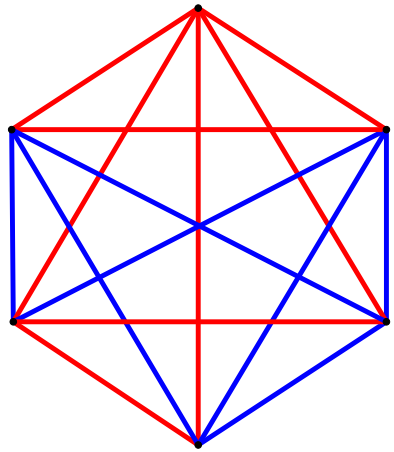




例 Ramsey问题

- **问题：**任何6个人的聚会，其中总会有3个互相认识或3人互相不认识。
- **图论模型：**用红、蓝两种颜色对6个顶点的完全图K_6的**边**进行任意着色，则不论如何着色必然都存在一个红色的K_3或一个蓝色的K_3。

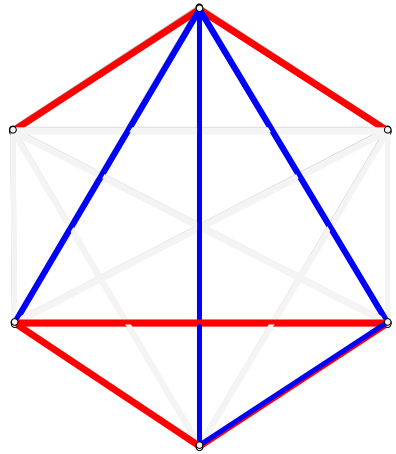




- **对应关系：每个人即为一个结点；人与人之间的关系即为一条边**

例 Ramsey问题

- **图论证明：**
- 用红、蓝两种颜色对K_6的**边**进行着色，
- K_6的任意一个顶点均有5条边与之相连接，这5条边必有3条边的颜色是相同的，不妨设为蓝色（如图）
- 与这3条边相关联的另外3个节点之间的3条边，若都为红色，则形成**红色的K_3**；
- 若另外3个节点之间的3条边有一条为蓝色，则与上面的蓝色边形成**蓝色的K_3**；
- 因此必然存在一个红色的K_3或一个蓝色的K_3。



例 Ramsey问题

- **Ramsey数：**R (3,3) =6 ; R (3,4) =9 ; 。 。 。 。 。 。

r, s	3	4	5	6	7	8	9	10
3	6	9	14	18	23	28	36	40 - 43
4	9	18	25	35 - 41	49 - 61	56 - 84	73 - 115	92 - 149
5	14	25	43 - 49	58 - 87	80 - 143	101 - 216	125 - 316	143 - 442
6	18	35 - 41	58 - 87	102 - 165	113 - 298	127 - 495	169 - 780	179 - 1171
7	23	49 - 61	80 - 143	113 - 298	205 - 540	216 - 1031	233 - 1713	289 - 2826
8	28	56 - 84	101 - 216	127 - 495	216 - 1031	282 - 1870	317 - 3583	317 - 6090
9	36	73 - 115	125 - 316	169 - 780	233 - 1713	317 - 3583	565 - 6588	580 - 12677
10	40 - 43	92 - 149	143 - 442	179 - 1171	289 - 2826	317 - 6090	580 - 12677	798 - 23556

例 过河问题

- **问题**：一摆渡人欲将一只狼、一头羊、一篮菜从河西渡过河到河东。由于船小，一次只能带一物过河，并且狼与羊，羊与菜不能独处，给出渡河方法。
- 这里显然不能用一个节点表示一个物体。一个物体可能在河东，也可能在河西，也可能在船上，状态表示不清楚。
- 另外，问题也可以分成几个小问题，
  如：问题是否能解？
  有几种不同的解法？
  最快的解决方案是什么？

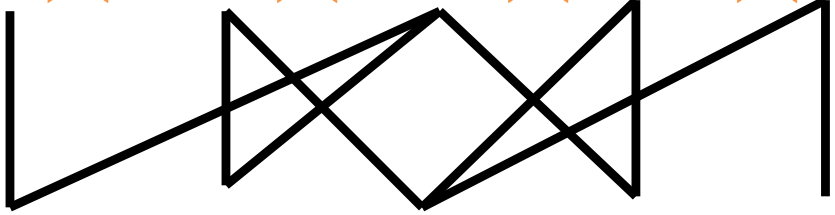
例 过河问题

- **解：** 用四维0-1向量表示(人,狼,羊,菜)在河西岸的状态(在河西岸则分量取1，否则取0)，共有2^4 =16 种状态。在河东岸的状态类似记作。
- 由题设，状态(0,1,1,0)，(0,0,1,1)，(0,1,1,1)是不允许的；
- 其对应状态：(1,0,0,1)， (1,1,0,0)，(1,0,0,0)也是不允许的。
- **以可允许的10个状态向量作为顶点，将可能互相转移的状态用边连接起来构成一个图。**
- 利用图论的相关知识即可回答原问题。

例 过河问题

(1,1,1,1) (1,1,1,0) (1,1,0,1) (1,0,1,1) (1,0,1,0)

(0,0,0,0) (0,0,0,1) (0,0,1,0) (0,1,0,0) (0,1,0,1)



(0,1,0,1) (0,1,0,0) (0,0,1,0) (0,0,0,1) (0,0,0,0)

(1,0,1,0) (1,0,1,1) (1,1,0,1) (1,1,1,0) (1,1,1,1)

河西=(人,狼,羊,菜) 河东=(人,狼,羊,菜)

将10个顶点分别记为$A_1, A_2, \cdots, A_{10}$，

从图中易得到两条路：

$A_1A_6A_3A_7A_2A_8A_5A_{10}$;

$A_1A_6A_3A_9A_4A_8A_5A_{10}$.

问题的转换：

➢过河问题是否能解？

即：图中A_1到A_{10}是否连通？

➢有几种不同的解法？

即：A_1到A_{10}之间有多少条不同的路径？

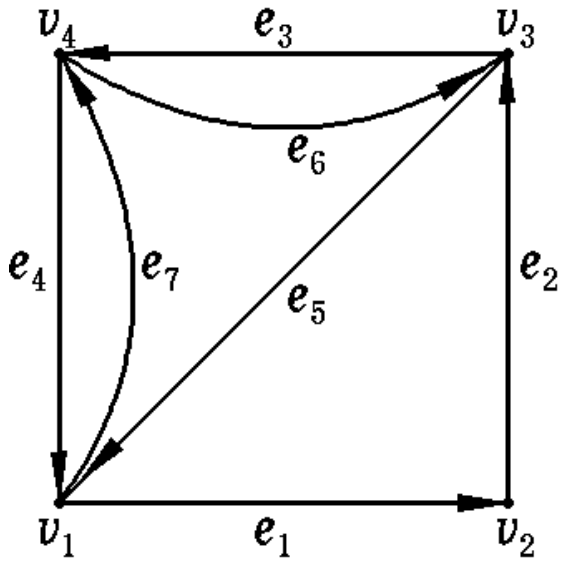
➢最快的解决方案是什么？

即：A_1到A_{10}最短路径有哪些？

图的矩阵表示

(1) 邻接矩阵：邻接矩阵表示了点与点之间的邻接关系. 一个n阶图G的邻接矩阵$A=(a_{ij})_{n\times n}$，其中

$$a_{ij}=\begin{cases}1, & v_{ij}\in E;\\ 0, & v_{ij}\notin E.\end{cases}$$

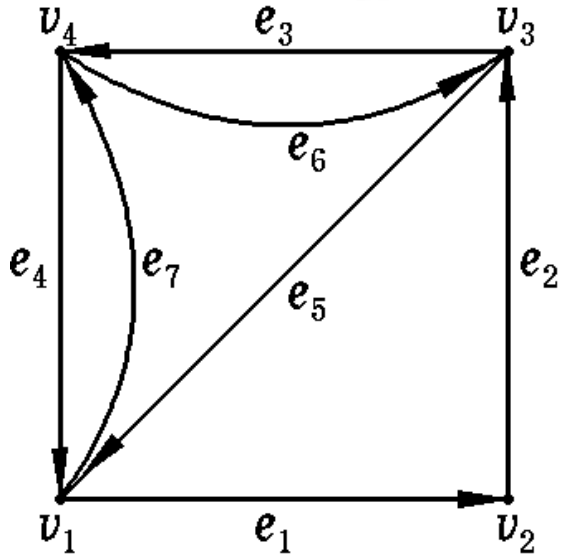


$$\mathbf{A}=\begin{pmatrix}0&1&0&1\\0&0&1&0\\1&0&0&1\\1&0&1&0\end{pmatrix}$$

$$d_{out}(v_i)=\sum_{j=1}^{n}a_{ij}$$

$$d_{in}(v_j)=\sum_{i=1}^{n}a_{ij}$$

$$\mathbf{A}=\begin{pmatrix}0&1&0&1\\0&0&1&0\\1&0&0&1\\1&0&1&0\end{pmatrix}$$



令$A^k=(a^{(k)}_{ij})$,

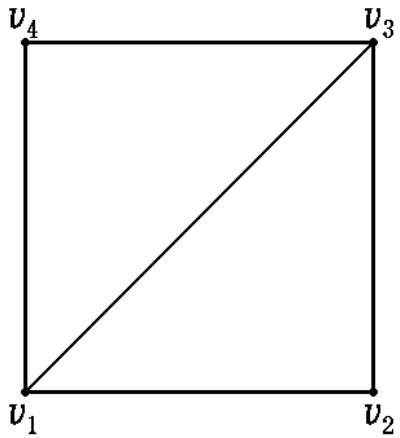
若$a^{(k)}_{ij}=t$,

$$\mathbf{A}^2=\begin{pmatrix}1&0&2&0\\1&0&0&1\\1&1&1&1\\1&1&0&2\end{pmatrix}$$

$$\mathbf{A}^3=\begin{pmatrix}2&1&0&3\\1&1&1&1\\2&1&2&2\\2&1&3&1\end{pmatrix}$$

则表示：图中v_i到v_j有t条长度为k的路径.

无向图G的邻接矩阵A是一个对称矩阵.



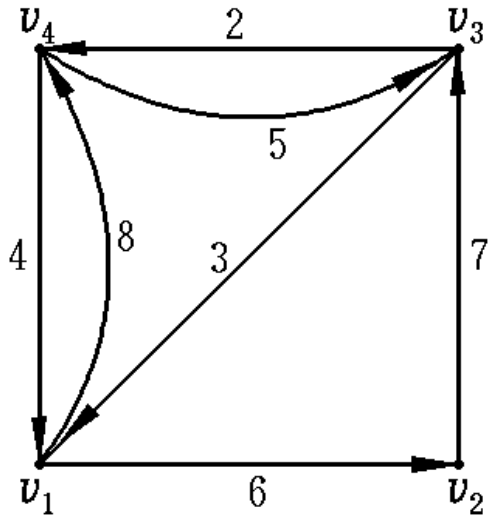
$$\mathbf{A}=\begin{pmatrix}0&1&1&1\\1&0&1&0\\1&1&0&1\\1&0&1&0\end{pmatrix}$$

$$d(v_i)=\sum_{j=1}^{n}a_{ij}=\sum_{j=1}^{n}a_{ji}$$

(2) 权矩阵　一个n阶赋权图$G=(V,E,F)$的权矩阵$A=(a_{ij})_{n\times n}$，其中

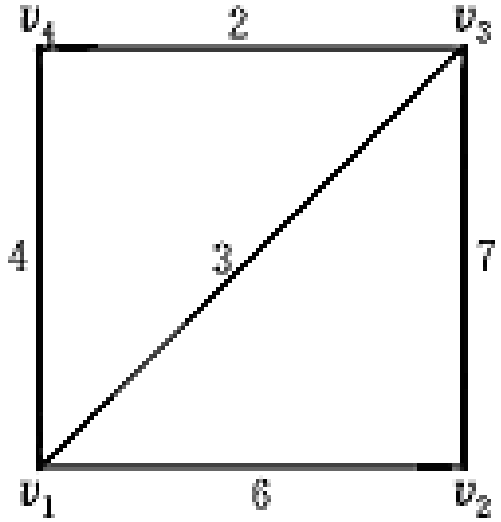
$$a_{ij}=\begin{cases}F(v_iv_j), & v_{ij}\in E;\\ 0, & i=j;\\ \infty, & v_{ij}\notin E.\end{cases}$$

有限简单图基本上可用权矩阵来表示.



$$\mathbf{A}=\begin{pmatrix} 0 & 6 & \infty & 8 \\ \infty & 0 & 7 & \infty \\ 3 & \infty & 0 & 2 \\ 4 & \infty & 5 & 0 \end{pmatrix}$$

无向图*G*的权矩阵*A*是一个对称矩阵.

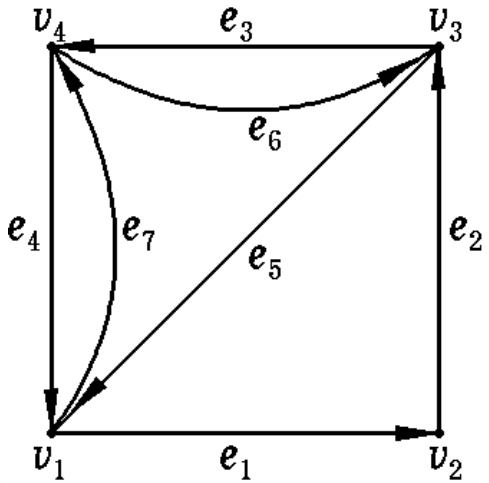


$$\mathbf{A}=\begin{pmatrix} 0 & 6 & 3 & 4 \\ 6 & 0 & 7 & \infty \\ 3 & 7 & 0 & 2 \\ 4 & \infty & 2 & 0 \end{pmatrix}$$

(3) 关联矩阵：一个有m条边的n阶有向图G的关联矩阵$A=(a_{ij})_{n\times m}$，其中

$$a_{ij}=\begin{cases}1, & \text{若}v_i\text{是}e_j\text{的始点;}\\ -1, & \text{若}v_i\text{是}e_j\text{的终点;}\\ 0, & \text{若}v_i\text{与}e_j\text{不关联.}\end{cases}$$

有向图的关联矩阵每列的元素中有且仅有一个1, 有且仅有一个－1.

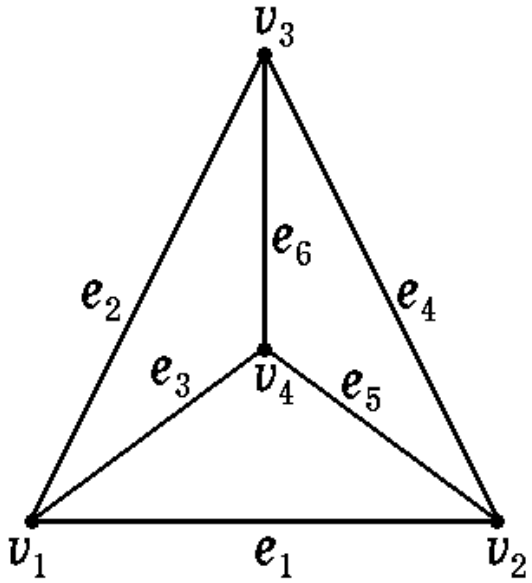


$$\mathbf{A}=\begin{pmatrix}1 & 0 & 0 & -1 & -1 & 0 & 1\\ -1 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & -1 & 1 & 0 & 1 & -1 & 0\\ 0 & 0 & -1 & 1 & 0 & 1 & -1\end{pmatrix}$$

一个有m条边的n阶无向图G的关联矩阵$A=(a_{ij})_{n\times m}$，其中

$$a_{ij}=\begin{cases}1, & \text{若}v_i\text{与}e_j\text{关联;}\\ 0, & \text{若}v_i\text{与}e_j\text{不关联.}\end{cases}$$

无向图的关联矩阵每列的元素中有且仅有两个1.



$$\mathbf{A}=\begin{pmatrix}1&1&1&0&0&0\\1&0&0&1&1&0\\0&1&0&1&0&1\\0&0&1&0&1&1\end{pmatrix}$$

例 锁具装箱问题

1994年全国大学生数学建模竞赛B题(锁具装箱)中关于锁具总数的问题可叙述如下：

某厂生产一种弹子锁具，每个锁具的钥匙有5个槽，每个槽的高度从{1,2,3,4,5,6}中任取一数。由于工艺及其它原因，制造锁具时对5个槽的高度有两个限制：

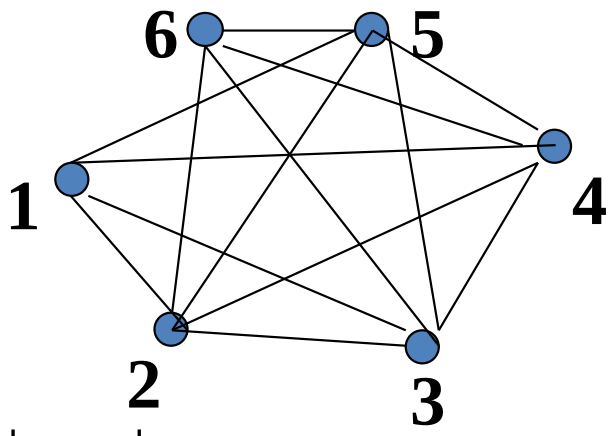
(1)至少有3个不同的数；

(2)相邻两槽的高度之差不能为5。

满足以上条件制造出来的所有互不相同的锁具称为一批．我们的问题是如何确定每一批锁具的个数?

该问题用图论中的邻接矩阵解决较为简单

例 锁具装箱问题



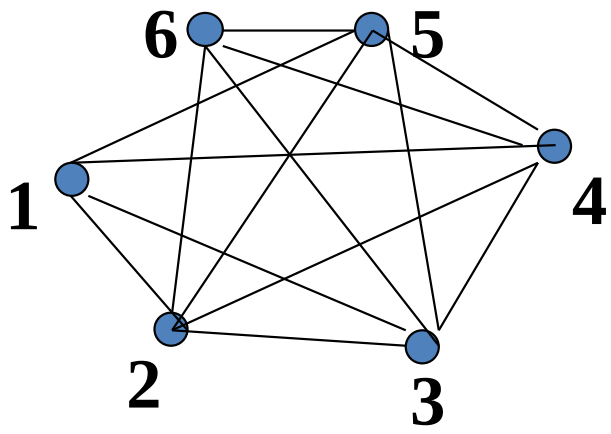
构造无向图

$$G=(V,\ E),V=\{1,2,3,4,5,6\},E=\{ij\,|\,i,\ j\in V\text{且}|i-j|\neq 5\}$$

在G中每一条长度为4的道路对应于一个相邻两槽高度之差不超过5的锁具，即满足限制条件（2）的锁具，反之亦然，于是可以通过G的邻接矩阵A来计算t的值，具体步骤如下：

例 锁具装箱问题

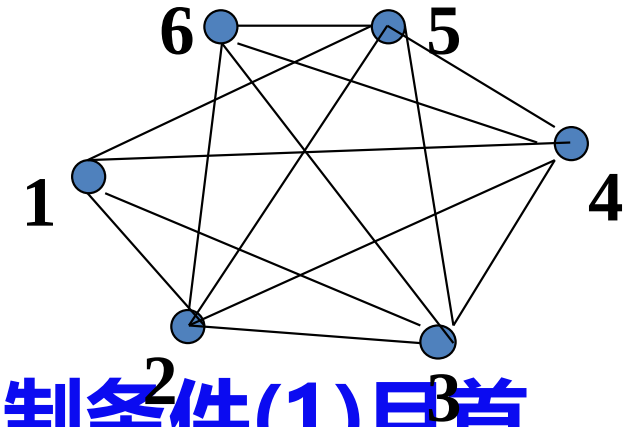
$$A=\begin{bmatrix}1&1&1&1&1&0\\1&1&1&1&1&1\\1&1&1&1&1&1\\1&1&1&1&1&1\\1&1&1&1&1&1\\0&1&1&1&1&1\end{bmatrix}\qquad A^2=\begin{bmatrix}5&5&5&5&5&4\\5&5&5&5&5&5\\5&5&5&5&5&5\\5&5&5&5&5&5\\5&5&5&5&5&5\\4&5&5&5&5&5\end{bmatrix}$$



$$A^4=\begin{bmatrix}141&165&165&165&165&140\\165&194&194&194&194&165\\165&194&194&194&194&165\\165&194&194&194&194&165\\165&194&194&194&194&165\\140&165&165&165&165&141\end{bmatrix}$$

因此， $t=\sum_{i=1}^{6}\sum_{j=1}^{6}a^{(4)}{}_{ij}=6306$.

例 锁具装箱问题



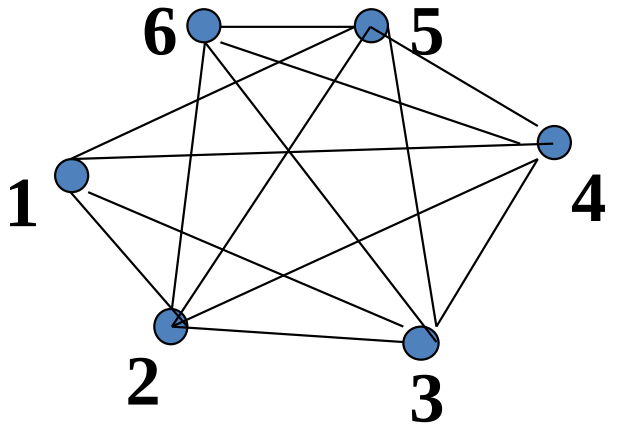
又令 $s = y_1 + y_2 + y_3 + y_4 + y_5 + y_6$,

其中y_i表示满足限制条件(2)但不满足限制条件(1)且首位为i的锁具数, (i=1,2,3,4,5,6),显然$y_1=y_6$, $y_2=y_4=y_3=y_5$,于是我们只需要计算y_1和y_2.

计算y_1可分别考虑槽高只有1，12，13，14，15的情形．若只有1，这样的锁具效只有1个，

若只有1和i(i=2,3,4,5)，这样的锁具数=G中以1和i为顶点，长度为3的道路数，此数可通过A的子矩阵A_{1i}计算得到．

例 锁具装箱问题



事实上，因为

$$A_{1i}=\begin{matrix}\text{第1行}\\ \text{第i行}\end{matrix}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix},\quad A_{1i}^{2}=\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix},\quad A_{1i}^{3}=\begin{bmatrix}4 & 4\\ 4 & 4\end{bmatrix}$$

其中i=2,3,4,5,显然y_1=1+(4+4+4+4-1) 4=61.

同理，计算y_2时应考虑槽高只有2,21,23,24,25,26时的情形，类似计算可得

y_2=1+(4+4+4+4-1)×5=76 .

于是，s=61×2+76×4=426，x=6306-426=5880.

该算法既易理解又易操作并且又可扩展 .

2、最短路径算法

定义1 设$P(u,v)$是赋权图$G=(V,E,F)$中从点u到v的路径，用$E(P)$表示路径$P(u,v)$中全部边的集合，记

$$F(P)=\sum_{e\in E(P)}F(e)$$

则称$F(P)$为路径$P(u,v)$的**权**或**长度**（距离）.

定义2 若$P_0(u,v)$是G中连接u,v的路径，且对任意在G中连接u,v的路径$P(u,v)$都有

$$F(P_0)\leqslant F(P),$$

则称$P_0(u,v)$是G中连接u,v的**最短路**.

重要性质：

若$v_0v_1\ldots v_m$是图G中从v_0到v_m的最短路，则$\forall 1\le k\le m$，$v_0v_1\ldots v_k$必为G中从v_0到v_k的最短路.

即：最短路是一条路，且最短路的任一段也是最短路.

求非负赋权图G中某一点到其它各点最短路，一般用**Dijkstra**标号算法；求非负赋权图上任意两点间的最短路，一般用**Floyd**算法.

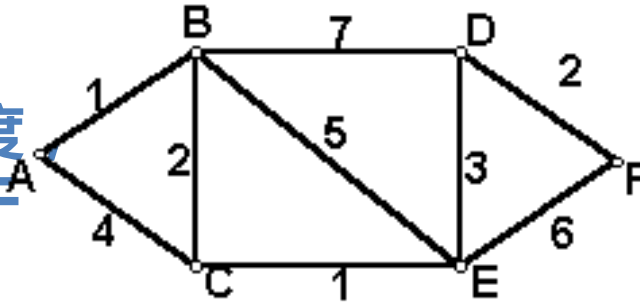
这两种算法均适用于有向非负赋权图.

下面分别介绍两种算法的基本过程.

Dijkstra算法

- **指标（a为起点）**

  设T为V的子集，P=V-T且a∈T，对所有t∈T，设 l(t)表示从a到t的所有通路中距离最短的一条的长度且这条路径中不包含T中其他的结点，则称l(t)为t关于集合P的指标，若不存在这样的路径，这记l(t)=∞



- **注：l(t)不一定是从a到t的最短路径，因为最短路径中可能包含T中其他的节点。**

- **定理1** 若t是T中关于P由最小指标的结点，则l(t)是a和t之间的最短距离。

- **定理2** 设 T为V的子集，P=V-T，设

  (1)对P中的任一点p,存在一条从a到p的最短路径,这条路径仅有P中的点构成,

  (2)对于每一点t,它关于P的指标为l(t),令x为最小指标所在的点, 即:

$$l(x)=\min\{l(t)\},t\in T$$

  (3)令P'=P U{x},T'=T-{x},l'(t)表示T'中结点t关于P'的指标,则

$$l'(t)=\min\{l(t),l(x)+w(x,t)\}$$

Dijkstra算法（求a点到其他点的最短路径）

1、初始化，P={a}，T=V-{a}，对每个结点t计算指标

l(t)=w(a,t)

2、设x为T中关于P有最小指标的点,

即:l(x)=min(l(t)) (t∈T),

3、若T=Φ,则算法结束;

否则,令P'=P U{x},T'=T-{x}

按照公式l'(t)=min{l(t),l(x)+w(x,t)},

计算T'中每一个结点t'关于P'的指标.

4、P'代替P,T'代替T,重复步骤2,3

（ 其中：w(x,y)为图的权矩阵）

改进Dijkstra算法（求a点到其他点的最短路径）

1、初始化，P={a}，T=V-{a}，对每个结点t计算指标

l(t)=w(a,t) ,pro(t)=a;

2、设x为T中关于P有最小指标的点,

即:l(x)=min(l(t)) (t∈T);

3、若T=Φ,则算法结束;

否则,令P‘=P U{x},T’=T-{x}

按照公式l‘(t)=min{l(t),l(x)+w(x,t)},

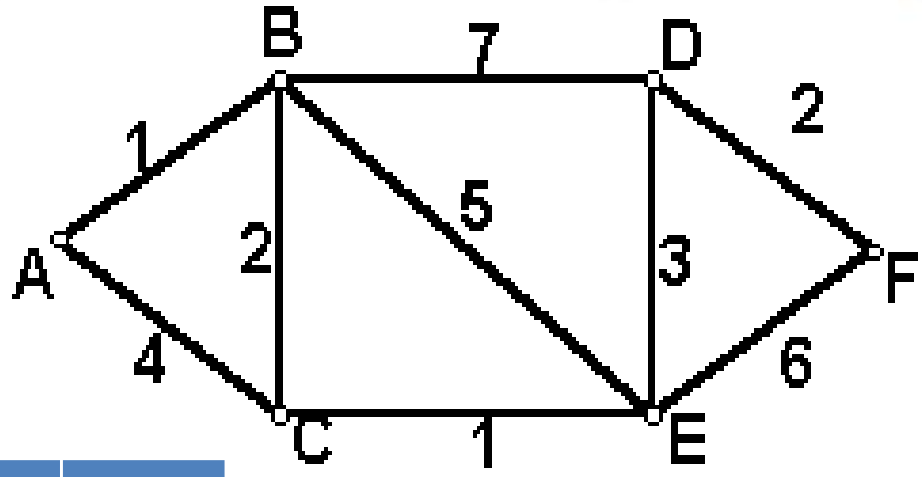
计算T’中每一个结点t‘关于P’的指标.

若 l(x)+w(x,t) < l(t), pro(t)=x;

4、P'代替P,T'代替T,重复步骤2,3

（ 其中：w(x,y)为图的权矩阵，pro(t)中保存了到达t的前一个结点）

例　求下图中A点到其他点的最短路.



V_i	P/T	Pro	L(t)				
A	1						
B	2	A	1				
C	3	AB	4	3			
D	5	BE	∞	8	8	7	
E	4	BC	∞	6	4		
F	6	ED	∞	∞	∞	10	9

A到F的最短路径为：9

路径依次为：
A-B-C-E-D-F

Floyd算法（求任意两点的最短路径）

设$A=(a_{ij})_{n\times n}$为赋权图$G=(V, E, F)$的权矩阵，d_{ij}表示从v_i到v_j点的距离，r_{ij}表示从v_i到v_j点的最短路中下一个点的编号.

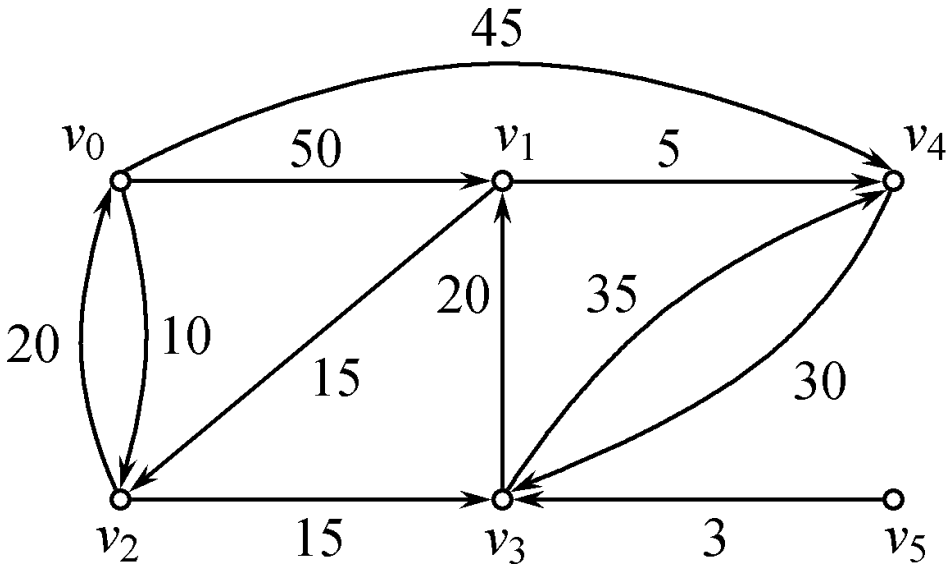
① 赋初值. 对所有i, j, $d_{ij}=a_{ij}$, $r_{ij}=j$. $k=1$. 转向②.

② 更新d_{ij}, r_{ij}. 对所有i, j, 若$d_{ik}+d_{kj}<d_{ij}$, 则令$d_{ij}=d_{ik}+d_{kj}$, $r_{ij}=r_{ik}$, 转向③;

③ 终止判断. 若$k=n$终止; 否则令$k=k+1$, 转向②.

最短路线可由r_{ij}得到.

例　求下图中任意两点间的最短路.



$$A=\begin{bmatrix} 0 & 50 & 10 & \infty & 45 & \infty \\ \infty & 0 & 15 & \infty & 5 & \infty \\ 20 & \infty & 0 & 15 & \infty & \infty \\ \infty & 20 & \infty & 0 & 35 & \infty \\ \infty & \infty & \infty & 30 & 0 & \infty \\ \infty & \infty & \infty & 3 & \infty & 0 \end{bmatrix}$$

$$D=\begin{bmatrix} 0 & 45 & 10 & 25 & 45 & \infty \\ 35 & 0 & 15 & 30 & 5 & \infty \\ 20 & 35 & 0 & 15 & 40 & \infty \\ 55 & 20 & 35 & 0 & 25 & \infty \\ 85 & 50 & 65 & 30 & 0 & \infty \\ 58 & 23 & 38 & 3 & 28 & 0 \end{bmatrix}$$

$$R=\begin{bmatrix} 0 & 2 & 2 & 2 & 4 & -1 \\ 2 & 1 & 2 & 2 & 4 & -1 \\ 0 & 3 & 2 & 3 & 3 & -1 \\ 1 & 1 & 1 & 3 & 1 & -1 \\ 3 & 3 & 3 & 3 & 4 & -1 \\ 3 & 3 & 3 & 3 & 3 & 5 \end{bmatrix}$$

例　设备更新问题

某企业每年年初, 都要作出决定, 如果继续使用旧设备, 要付维修费；若购买一台新设备, 要付购买费. 试制定一个5年更新计划, 使总支出最少.

已知设备在每年年初的购买费分别为11, 11, 12, 12, 13. 使用不同时间设备所需的维修费分别为5, 6, 8, 11, 18.

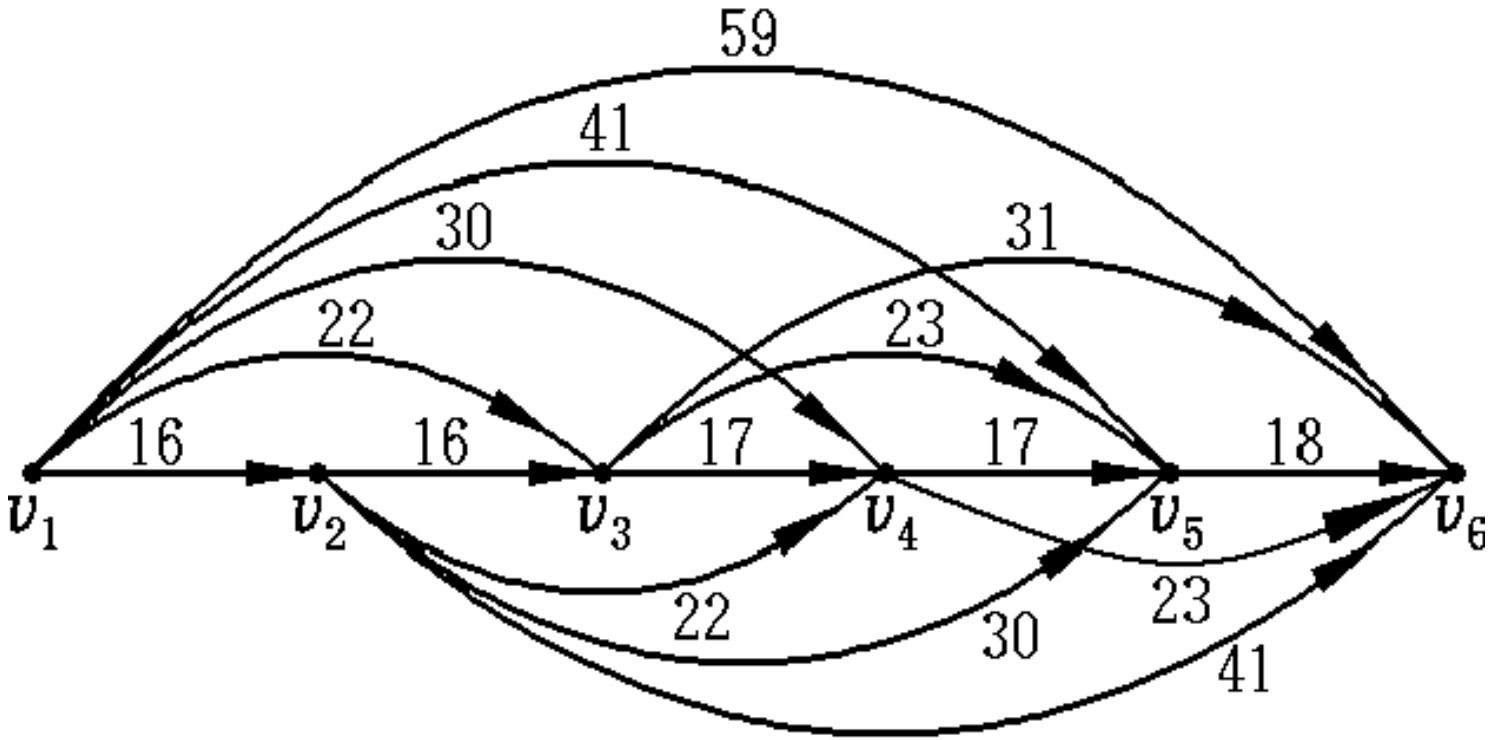
解 设b_i表示设备在第i年年初的购买费, c_i表示设备使用i年后的维修费,

$V=\{v_1, v_2, \cdots, v_6\}$, 点$v_i$表示第$i$年年初购进一台新设备, 虚设一个点$v_6$表示第5年年底.

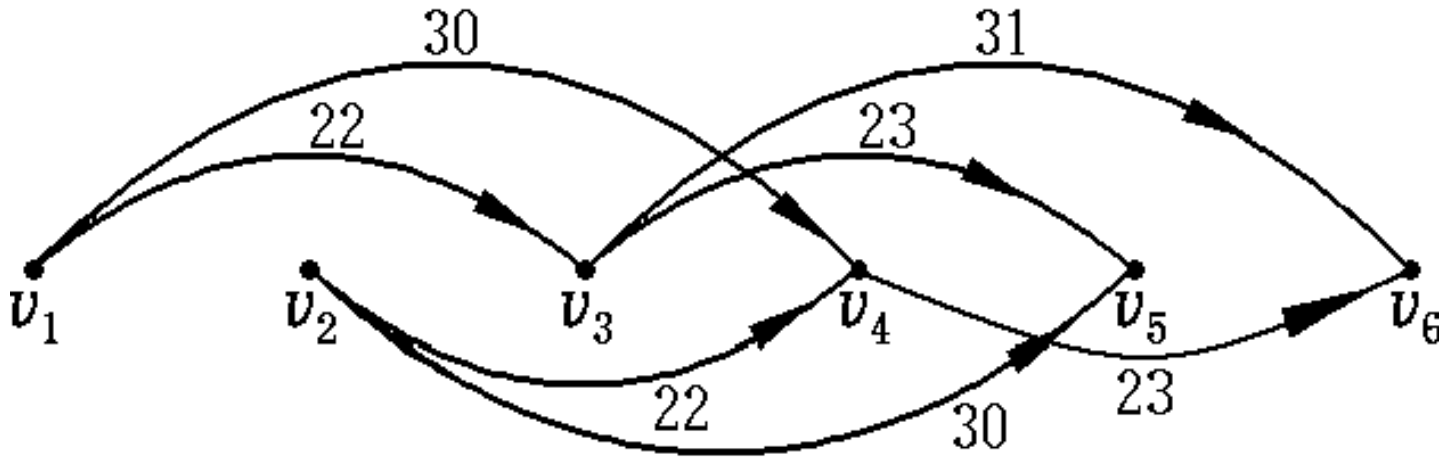
$E=\{v_iv_j \mid 1\le i<j\le 6\}$, 每条边$v_iv_j$表一台设备，从第$i$年年初购买使用到第$j$年年初报废.

$$F(v_iv_j)=b_i+\sum_{k=1}^{j-i}c_k$$

这样上述设备更新问题就变为：在有向赋权图$G=(V, E, F)$(图解如下）中求v_1到v_6的最短路问题.



由实际问题可知, 设备使用三年后应当更新, 因此删除该图中v_1到v_5, v_1到v_6, v_2到v_6的连线；又设备使用一年后就更新则不划算, 因此再删除该图中
v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_6五条连线后得到



从上图中容易得到v_1到v_6只有两条路：

$$v_1v_3v_6\text{和}v_1v_4v_6.$$

而这两条路都是v_1到v_6的最短路.

3、最小生成树

由树的定义不难知道,任意一个连通的(n,m)图G适当去掉$m-n+1$条边后,都可以变成树,这棵树称为图G的**生成树**.

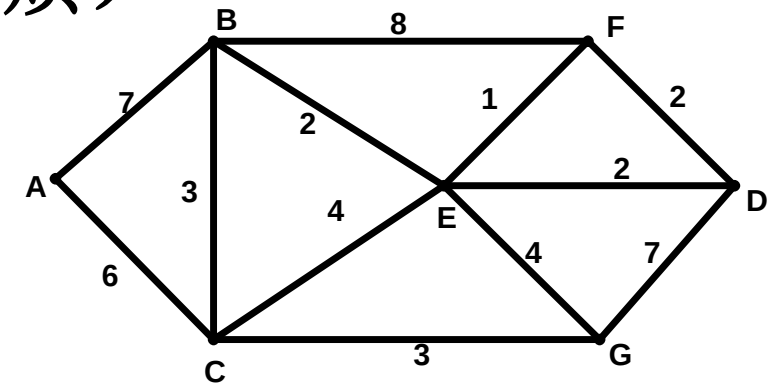
设T是图G的一棵生成树,用$F(T)$表示树T中所有边的权数之和,$F(T)$称为**树T的权**.

一个连通图G的生成树一般不止一棵,图G的所有生成树中权数最小的生成树称为图G的**最小生成树-Minimum-cost Spanning Tree (MST)**.

求最小生成树问题有很广泛的实际应用.例如,把n个乡镇用高压电缆连接起来建立一个电网,使所用的电缆长度之和最短,即费用最小,就是一个求最小生成树问题.

Kruskal算法

从未选入树中的边中选取权重最小的且不构成回路的边加入到树中.（判断回路比较麻烦）



算法过程：

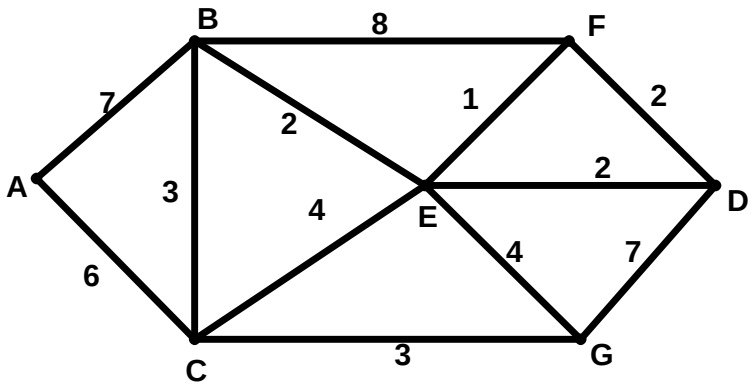
1.将各边按权值从小到大进行排序

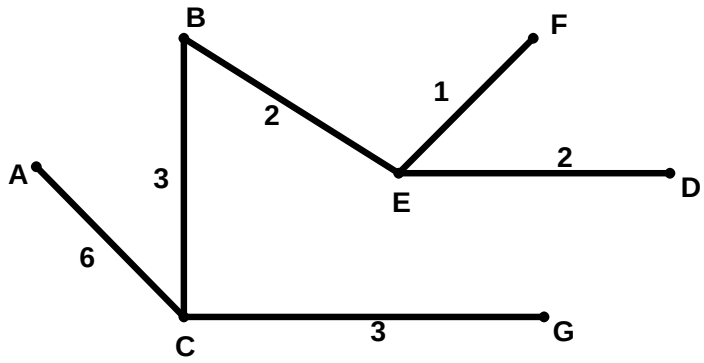
2.找出权值最小且不与已加入最小生成树集合的边构成回路的边。若符合条件，则加入最小生成树的集合中。不符合条件则继续遍历图，寻找下一个最小权值的边。

3.递归重复步骤1，直到找出n-1条边为止，得到最小生成树。

时间复杂度只和边有关，可以证明其时间复杂度为O(mlogm)

求最小生成树





Prim算法

把结点分成两个集合，已加入树中的(*P*)和未加入树中的(*Q*)，然后每次选择一个点在*P*中一个点在*Q*中的权重最小的边，加入树中，并把相应点加入*P*中。

算法过程：

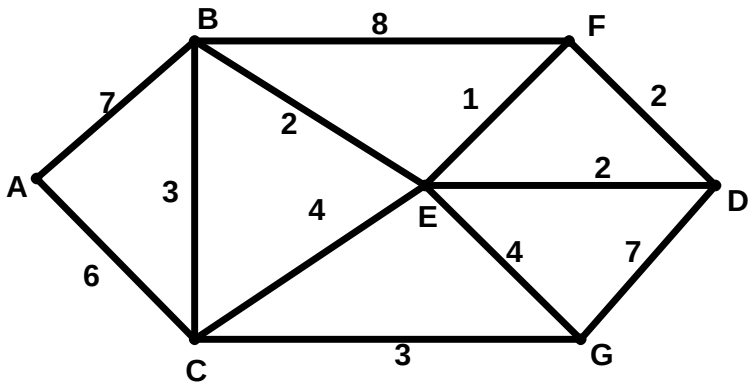
1：初始化：任取一个节点*u*加入生成树,$P=\{u_0\}$, $Q=V-\{u_0\}$,生成树的边集$TE=\Phi$。

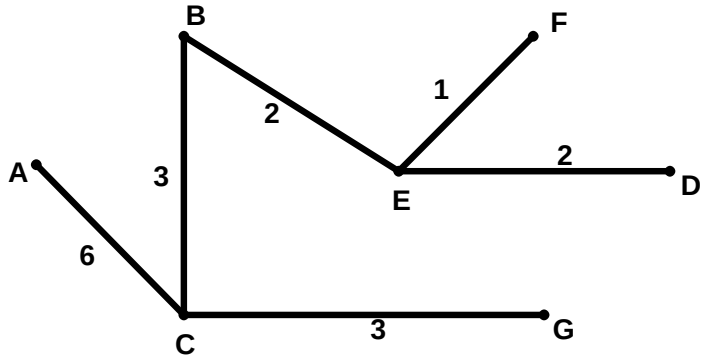
2：在所有$u\in P,v\in Q$的边(u,v)（称为可选边集）中，找一条权重最小的边(u_1,v_1)，将此边加进集合*TE*中，并将*v*加入*P*中。同时对与*v*关联的边，若另一个端点已经在P中，则从可选边集中删除，否则加入可选边集。

3：如果*P=V*，则算法结束；否则重复步骤2。

我们可以算出当*U=V*时，步骤2共执行了$n-1$次，*TE*中也增加了$n-1$条边，这$n-1$条边就是需要求出的最小生成树的边。

求最小生成树





改进的Kruskal算法

参考Prim算法中P、Q集合的划分以及构造方法，我们可以基于分类合并的思想提出改进的Kruskal算法，

算法过程：

1.将各边按权值从小到大进行排序;

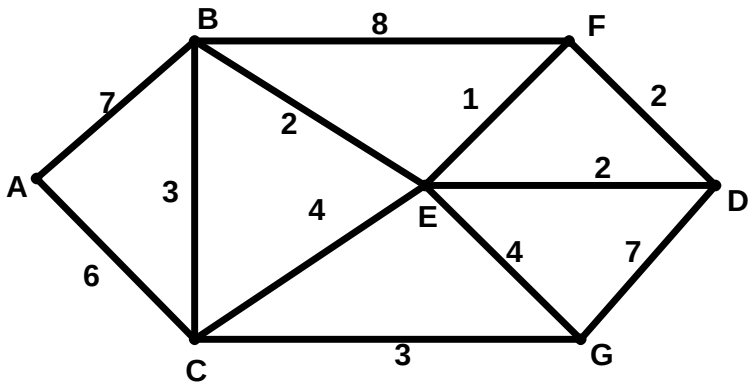
2.将每个节点分为一个类别，并编号1,2,…,n;

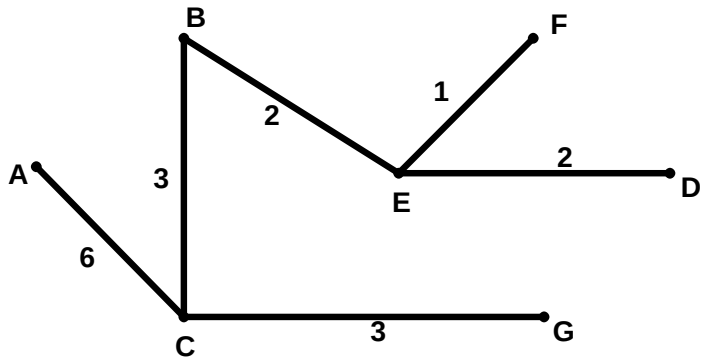
3.找出权值最小的边，若该边的两个端点属于两个不同的类别，则加入最小生成树的集合中，并把这两个类别合并（修改为同一个编号）。若不符合条件则继续遍历图，寻找下一个最小权值的边。

3.递归重复步骤1，直到找出n-1条边为止，得到最小生成树。

类似地, 可定义连通图G的最大生成树.

求最小生成树





例　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

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?

4、遍历性问题

一、欧拉图

- G=(V,E)为一连通无向图
- 经过G中每条边至少一次的回路称为**巡回**；
- 经过G中每条边正好一次的巡回称为**欧拉巡回**；
- 存在欧拉巡回的图称为**欧拉图。**

二、中国邮递员问题（CPP - chinese postman problem）

- 一名邮递员负责投递某个街区的邮件。如何为他（她）设计一条最短的投递路线（从邮局出发，经过投递区内每条街道至少一次，最后返回邮局）？
- 这一问题是我国管梅谷教授1962年首先提出，国际上称之为中国邮递员问题。

- **解法：**

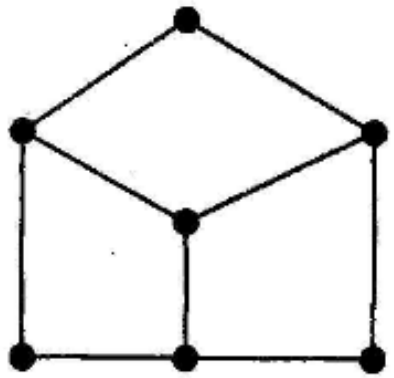
  若本身就是欧拉图，则直接可以找到一条欧拉巡回就是本问题的解。

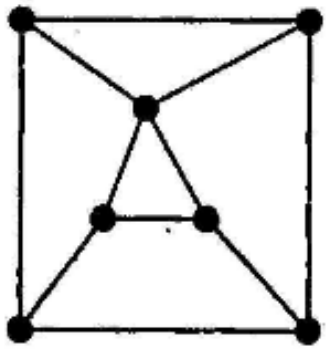
  若不是欧拉图，必定有偶数个奇度数结点，在这些奇度数点之间添加一些边，使之变成欧拉图，再找出一个欧拉巡回。

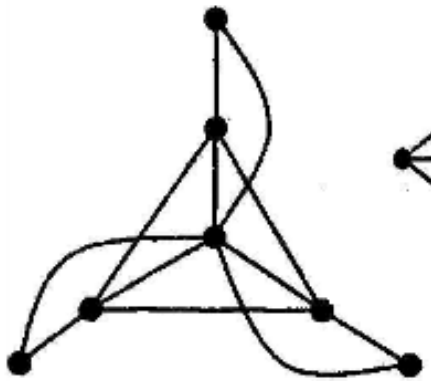
- **具体解法：Fleury算法+Edmonds最小对集算法**

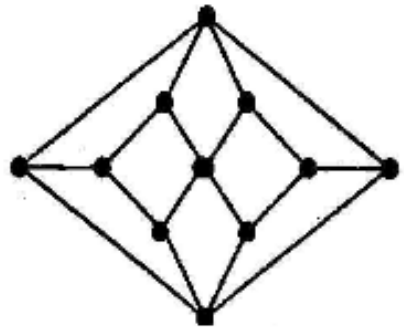
三、哈密尔顿图

- G=(V,E)为一连通无向图
- 经过G中每点一次且正好一次的路径称为**哈密尔顿路径**；
- 经过G中每点一次且正好一次的回路称为**哈密尔顿回路**；
- 存在哈密尔顿回路的图称为**哈密尔顿图。**









四、旅行商问题（TSP - Traveling Salesman Problem）

- 一名推销员准备前往若干城市推销产品。如何为他（她）设计一条最短的旅行路线？ 即：从驻地出发，经过每个城市恰好一次，最后返回驻地**（最小哈密尔顿回路）**
- 对于n个节点的旅行商问题，n个节点的任意一个全排列都是问题的一个可能解（假设任意两个点之间都有边）。G个节点的全排列有(n-1)!个，因此间题归结为在(n-1)!个回路中选取最小回路。
- TSP问题的解法属于NP**完全问题**，一般只研究其近似解法

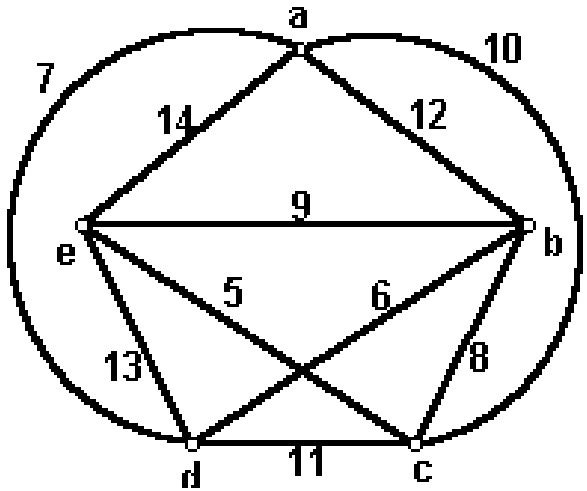
最邻近算法

(1) 选取任意一个点作为起始点，找出与该点相关联的权重最小的边，形成一条初始路径.

(2) 找出与最新加入到路径中的点相关联的权重最小的边加入到路径中，且要求不再路径中产生回路.

(3) 重复(2)直到所有的结点都加入到路径中.

(4) 将起点和最后加入的结点之间的边加入到路径中，形成Hamilton回路.



其他数值算法：

人工神经元方法，
遗传算法等等

5、二分图与匹配

定义1 设X, Y都是非空有限集, 且$X \cap Y = \phi$, $E \subset \{xy \mid x \in X, y \in Y\}$, 称$G = (X, Y, E)$为**二部图**.

二部图可认为是有限简单无向图.

如果X中的每个点都与Y中的每个点邻接, 则称$G = (X, Y, E)$为**完备二部图**.

若$F: E \to R^+$, 则称$G = (X, Y, E, F)$为**二部赋权图**.

二部赋权图的权矩阵一般记作

$$A = (a_{ij})_{|X| \times |Y|},$$

其中$a_{ij} = F(x_i y_j)$.

定义2 设$G=(X, Y, E)$为二部图, 且$M \subset E$. 若M中任意两条边在G中均不邻接, 则称M是二部图G的一个**匹配**.

定义3 设M是二部图G的一个匹配, 如果G的每一个点都是M中边的顶点, 则称M是二部图G的**完美匹配**;

如果G中没有另外的匹配M_0, 使$|M_0|>|M|$, 则称M是二部图G的**最大匹配**.

在二部赋权图$G=(X, Y, E, F)$中, 权数最大的最大匹配M称为二部赋权图G的**最佳匹配**.

显然, 每个完美匹配都是最大匹配, 反之不一定成立.

工作安排问题之一

给n个工作人员$x_1, x_2, \cdots, x_n$安排n项工作$y_1, y_2, \cdots, y_n$. n个工作人员中每个人能胜任一项或几项工作,但并不是所有工作人员都能从事任何一项工作.比如x_1能做y_1, y_2工作,x_2能做y_2, y_3, y_4工作等.

这样便提出一个问题,对所有的工作人员能不能都分配一件他所能胜任的工作?

我们构造一个二部图$G = (X, Y, E)$,这里$X = \{x_1, x_2, \cdots, x_n\}$,$Y = \{y_1, y_2, \cdots, y_n\}$,并且当且仅当工作人员$x_i$胜任工作$y_j$时,$x_i$与$y_j$才相邻.

于是,问题转化为求二部图的一个完美匹配.因为$|X|=|Y|$,所以完美匹配即为最大匹配.

求二部图$G=(X,Y,E)$的最大匹配算法（匈牙利算法，交替链算法）迭代步骤：

从G的任意匹配M开始.

① 将X中M的所有非饱和点都给以标号0和标记*，转向②.

M的非饱和点即非M的某条边的顶点.

② 若X中所有有标号的点都已去掉了标记*，则M是G的最大匹配. 否则任取X中一个既有标号又有标记*的点x_i，去掉x_i的标记*，转向③.

③ 找出在G中所有与x_i邻接的点y_j，若所有这样的y_j都已有标号，则转向②，否则转向④.

④ 对与x_i邻接且尚未给标号的y_j都给定标号i.

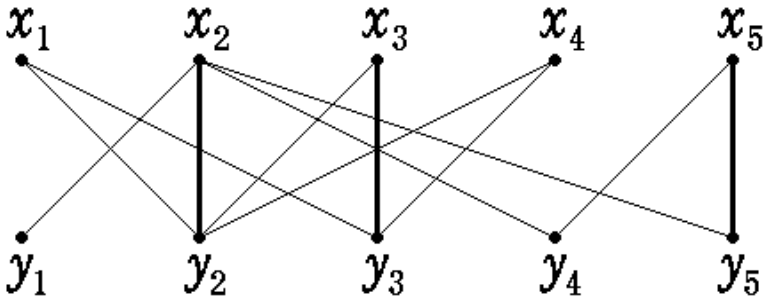
若所有的y_j都是M的饱和点, 则转向⑤, 否则逆向返回. 即由其中M的任一个非饱和点y_j的标号i找到x_i, 再由x_i的标号k找到y_k, …, 最后由y_t的标号s找到标号为0的x_s时结束, 获得M–增广路 $x_s y_t \dots x_i y_j$, 记$P=\{x_s y_t, \dots, x_i y_j\}$, 重新记$M$为$M \oplus P$, 转向①.

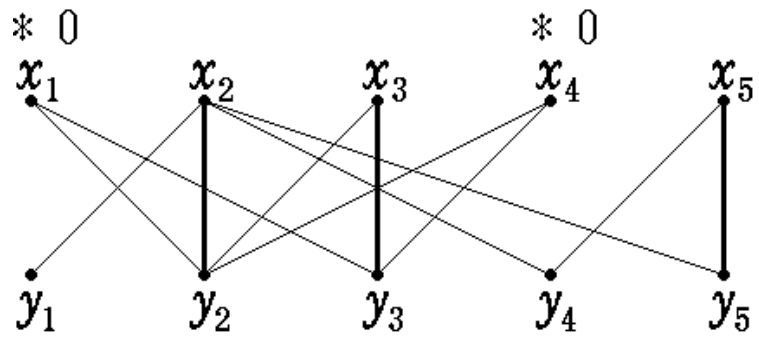
不必理会M–增广路的定义.

$M \oplus P = M \cup P \setminus M \cap P$, 是对称差.

⑤ 将y_j在M中与之邻接的点x_k, 给以标号 j 和标记*, 转向②.

例 求下图所示二部图G的最大匹配

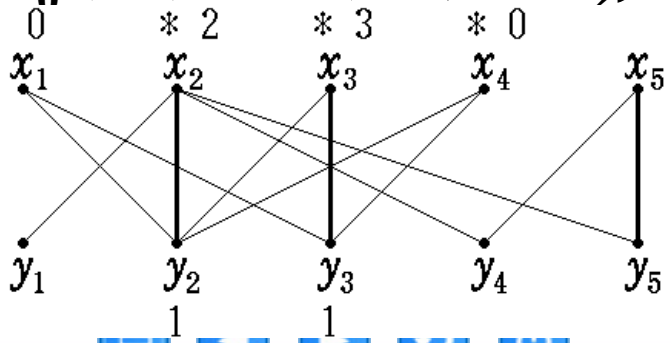




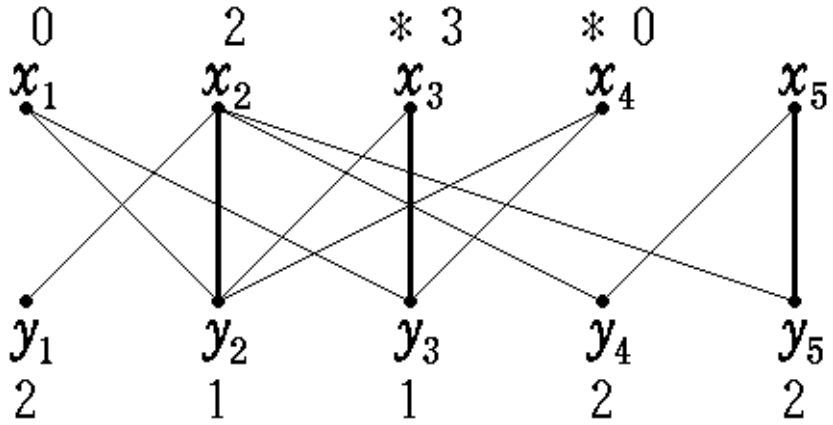
解 ① 取初始匹配$M_0=\{x_2y_2, x_3y_3, x_5y_5\}$（上图粗线所示）.

② 给X中M_0的两个非饱和点x_1, x_4都给以标号0和标记*（如下图所示）.

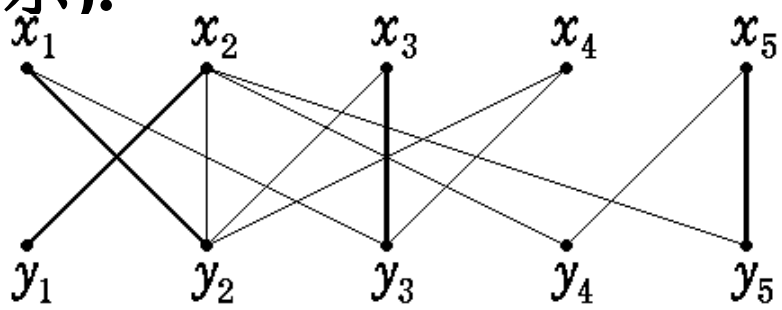
③ 去掉x_1的标记*,将与x_1邻接的两个点y_2,y_3都给以标号1. 因为y_2, y_3都是M_0的两个饱和点, 所以将它们在M_0中邻接的两个点x_2, x_3都给以相应的标号和标记* (如下图所示).



④ 去掉x_2的标记*,将与x_2邻接且尚未给标号的三个点y_1, y_4, y_5都给以标号2(如下图所示).

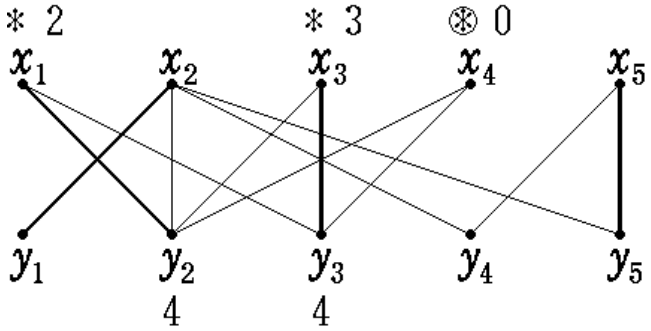


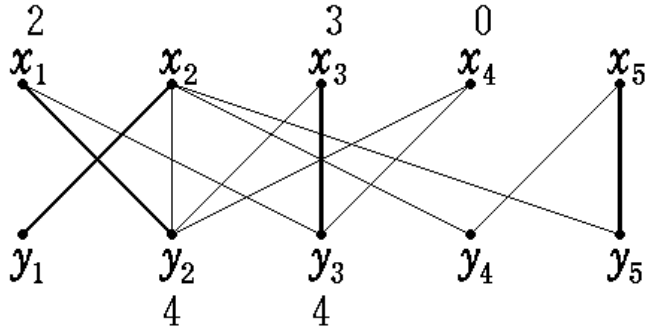
⑤ 因为y_1是M_0的非饱和点,所以顺着标号逆向返回依次得到x_2, y_2,直到x_1为0为止. 于是得到M_0的增广路$x_1y_2x_2y_1$, 记$P = \{x_1y_2, y_2x_2, x_2y_1\}$. 取$M_1 = M_0 \oplus P = \{x_1y_2, x_2y_1, x_3y_3, x_5y_5\}$, 则$M_1$是比$M$多一边的匹配(如下图所示).



⑥ 再给X中M_1的非饱和点x_4给以标号0和标记*, 然后去掉x_4的标记*, 将与x_4邻接的两个点y_2, y_3都给以标号4.

因为y_2, y_3都是M_1的两个饱和点，所以将它们在M_1中邻接的两个点x_1, x_3都给以相应的标号和标记* (如下图所示).





⑦ 去掉x_1的标记*, 因为与x_1邻接的两个点y_2, y_3都有标号4, 所以去掉x_3的标记*.

而与x_3邻接的两个点y_2, y_3也都有标号4, 此时X中所有有标号的点都已去掉了标记*(如下图所示), 因此M_1是G的最大匹配.

G不存在饱和X的每个点的匹配, 当然也不存在完美匹配.

工作安排问题之二

给n个工作人员$x_1, x_2, \cdots, x_n$安排n项工作$y_1, y_2, \cdots, y_n$. 如果每个工作人员工作效率不同, 要求工作分配的同时考虑总效率最高.

我们构造一个二部赋权图$G = (X, Y, E, F)$, 这里$X = \{x_1, x_2, \cdots, x_n\}$, $Y = \{y_1, y_2, \cdots, y_n\}$, $F(x_i y_j)$为工作人员x_i胜任工作y_j时的工作效率.

则问题转化为：求二部赋权图G的最佳匹配.

在求G的最佳匹配时, 总可以假设G为完备二部赋权图. 若x_i与y_j不相邻, 可令$F(x_i y_j)=0$. 同样地, 还可虚设点x或y, 使$|X|=|Y|$. 如此就将G转化为完备二部赋权图, 而且不会影响结果.

定义 设$G=(X, Y, E, F)$为完备的二部赋权图,
若L：$X \cup Y \to R^+$满足：

$$\forall x \in X, y \in Y, L(x) + L(y) \geq F(xy),$$

则称L为G的一个**可行点标记**,
记相应的生成子图为$G_L=(X, Y, E_L, F)$, 这里

$$E_L=\{xy \in E \mid L(x) + L(y) = F(xy)\}.$$

求完备二部赋权图$G = (X, Y, E, F)$的最佳匹配算法迭代步骤：

设$G=(X, Y, E, F)$为完备的二部赋权图, L是其一个初始可行点标记, 通常取

$$L(x) = \max\{F(xy) \mid y \in Y\},\ x \in X,$$

$$L(y) = 0,\quad y \in Y.$$

M是G_L的一个匹配.

① 若X的每个点都是饱和的,则M是最佳匹配.否则取M的非饱和点$u \in X$,令$S=\{u\}$, $T=\phi$,转向②.

② 记$N_L(S)=\{v|u \in S, uv \in G_L\}$.
若$N_L(S)=T$, 则G_L没有完美匹配, 转向③. 否则转向④.

③ 调整标记, 计算

$$a_L=\min\{L(x)+L(y)-F(xy)|x \in S, y \in Y \backslash T\}.$$

由此得新的可行点标记

$$H(v)=\begin{cases} L(v)-a_L, v \in S, \\ L(v)+a_L, v \in T, \\ L(v), v \in S^c \cap T^c. \end{cases}$$

令$L=H, G_L=G_H$, 重新给出G_L的一个匹配M, 转向①.

④ 取$y \in N_L(S) \backslash T$, 若$y$是$M$的饱和点, 转向⑤. 否则, 转向⑥.

⑤ 设$x y \in M$, 则令$S=S \cup \{x\}$, $T=T \cup \{y\}$, 转向②.

⑥ 在G_L中的$u-y$路是$M-$增广路, 设为P, 并令 $M=M \oplus P$, 转向①.

6、网络流问题

定义1 设$G=(V,E)$为有向图, 在V中指定一点称为**发点**(记为v_s), 和另一点称为**收点**(记为v_t), 其余点叫做中间点. 对每一条边$v_iv_j \in E$, 对应一个非负实数C_{ij}, 称为它的**容量**. 这样的G称为**容量网络**, 简称**网络**, 记作$G=(V,E,C)$.

定义2 网络$G=(V,E,C)$中任一条边v_iv_j有流量f_{ij}, 称集合$f=\{f_{ij}\}$为网络G上的一个**流**.

满足下述条件的流f称为**可行流**:

①(限制条件)对每一边v_iv_j, 有$0 \le f_{ij} \le C_{ij}$;

②(平衡条件)对于中间点v_k有$\sum f_{ik}=\sum f_{kj}$,

即中间点v_k的输入量=输出量.

如果 f 是可行流, 则对收、发点v_t、v_s有

$$\sum f_{si} = \sum f_{jt} = W_f,$$

即从v_s点发出的物质总量 = v_t点输入的量.

W_f称为网络流 f 的总流量.

上述概念可以这样来理解, 如G是一个运输网络, 则发点v_s表示发送站, 收点v_t表示接收站, 中间点v_k表示中间转运站, 可行流f_{ij}表示某条运输线上通过的运输量, 容量C_{ij}表示某条运输线能承担的最大运输量, W_f表示运输总量.

可行流总是存在的. 比如所有边的流量$f_{ij} = 0$就是一个可行流(称为零流).

所谓**最大流问题**就是在容量网络中, 寻找流量最大的可行流.

实际问题中, 一个网络会出现下面两种情况:

(1) 发点和收点都不止一个.

解决的方法是再虚设一个发点v_s和一个收点v_t, 发点v_s到所有原发点边的容量都设为无穷大, 所有原收点到收点v_t边的容量都设为无穷大.

(2) 网络中除了边有容量外, 点也有容量.

解决的方法是将所有有容量的点分成两个点, 如点v有容量C_v, 将点v分成两个点v'和v'', 令

$$C(v'v'') = C_v .$$

这里我们要进一步探讨不仅要使网上的流达到最大, 或者达到要求的预定值, 而且还要使运输流的费用是最小的, 这就是最小费用流问题.

最小费用流问题的一般提法：

已知网络$G=(V, E, C)$, 每条边$v_iv_j \in E$ 除了已给容量C_{ij}外, 还给出了单位流量的费用$b_{ij}(\geq 0)$.

所谓最小费用流问题就是求一个总流量已知的可行流 $f=\{f_{ij}\}$ 使得总费用

$$b(f)=\sum_{v_iv_j\in E} b_{ij}f_{ij} \quad \text{最小.}$$

特别地, 当要求 f 为最大流时, 即为最小费用最大流问题.

设网络 $G=(V, E, C)$, 取初始可行流 f 为零流, 求解最小费用流问题的迭代步骤:

① 构造有向赋权图 $G_f=(V, E_f, F)$, 对于任意的 $v_iv_j\in E$, E_f, F 的定义如下:

当 $f_{ij}=0$ 时, $v_iv_j\in E_f$, $F(v_iv_j)=b_{ij}$;

当 $f_{ij}=C_{ij}$ 时, $v_jv_i\in E_f$, $F(v_jv_i)=-b_{ij}$;

当 $0<f_{ij}<C_{ij}$ 时, $v_iv_j\in E_f$, $F(v_iv_j)=b_{ij}$, $v_jv_i\in E_f$, $F(v_jv_i)=-b_{ij}$.

然后转向②.

② 求出含有负权的有向赋权图$G_f=(V, E_f, F)$中发点v_s到收点v_t的最短路μ, 若最短路μ存在转向③; 否则f是所求的最小费用最大流, 停止.

③ 增流. $$\delta_{ij}=\begin{cases} C_{ij}-f_{ij}, v_iv_j\in\mu^+, & v_iv_j\text{与}\mu\text{相同,} \\ f_{ij}, \quad v_iv_j\in\mu^-. & v_iv_j\text{与}\mu\text{相反.} \end{cases}$$

令$\delta=\min\{\delta_{ij}\,|\,v_iv_j\in\mu\}$, 重新定义流$f=\{f_{ij}\}$为

$$f_{ij}=\begin{cases} f_{ij}+\delta, v_iv_j\in\mu^+, \\ f_{ij}-\delta, v_iv_j\in\mu^-. \end{cases}$$ 其它情况不变.

如果W_f大于或等于预定的流量值, 则适当减少δ值, 使W_f等于预定的流量值, 那么f是所求的最小费用流, 停止; 否则转向①

下面介绍求解有向赋权图$G=(V, E, F)$中含有负权的最短路的**Ford算法**.

设边的权v_iv_j为w_{ij}, v_1到v_i的路长记为$\pi(i)$. Ford算法的迭代步骤:

① 赋初值 $\pi(1)=0$, $\pi(i)=\infty$, $i=2, 3, \ldots, n$.

② 更新 $\pi(i)$, $i=2, 3, \ldots, n$.

$$\pi(i)=\min\{\pi(i), \min\{\pi(j)+w_{ji} \mid j\neq i\}\}.$$

③ 若所有的$\pi(i)$都无变化, 停止;否则转向②.

在算法的每一步, $\pi(i)$都是从v_1到v_i的最短路长度的上界. 若不存在负长回路, 则从v_1到v_i的最短路长度是$\pi(i)$的下界, 经过$n-1$次迭代后$\pi(i)$将保持不变. 若在第n次迭代后$\pi(i)$仍在变化时, 说明存在负长回路.

7、关键路径问题（拓扑排序）

一项工程任务，大到建造一座大坝，一座体育中心，小至组装一台机床，一架电视机，都要包括许多工序. 这些工序相互约束，只有在某些工序完成之后，一个工序才能开始. 即它们之间存在完成的先后次序关系，一般认为这些关系是预知的，而且也能够预计完成每个工序所需要的时间.

这时工程领导人员迫切希望了解最少需要多少时间才能够完成整个工程项目，影响工程进度的要害工序是哪几个？

PT(Potentialtask graph)图

在PT(Potentialtask graph)图中, 用结点表示工序, 如果工序 i 完成之后工序 j 才能启动, 则图中有一条有向边 (i, j), 其长度 w_i 表示工序 i 所需的时间.

这种图必定不存在有向回路, 否则某些工序将在自身完成之后才能开始, 这显然不符合实际情况.

在PT图中增加两个虚拟结点 v_0 和 v_n, 使所有仅为始点的结点都直接与 v_0 联结, v_0 为新增边的始点, 这些新增边的权都设为0; 使所有仅为终点的结点都直接与 v_n 联结, v_n 为新增边的终点. 这样得到的图 G 仍然不存在有向回路.

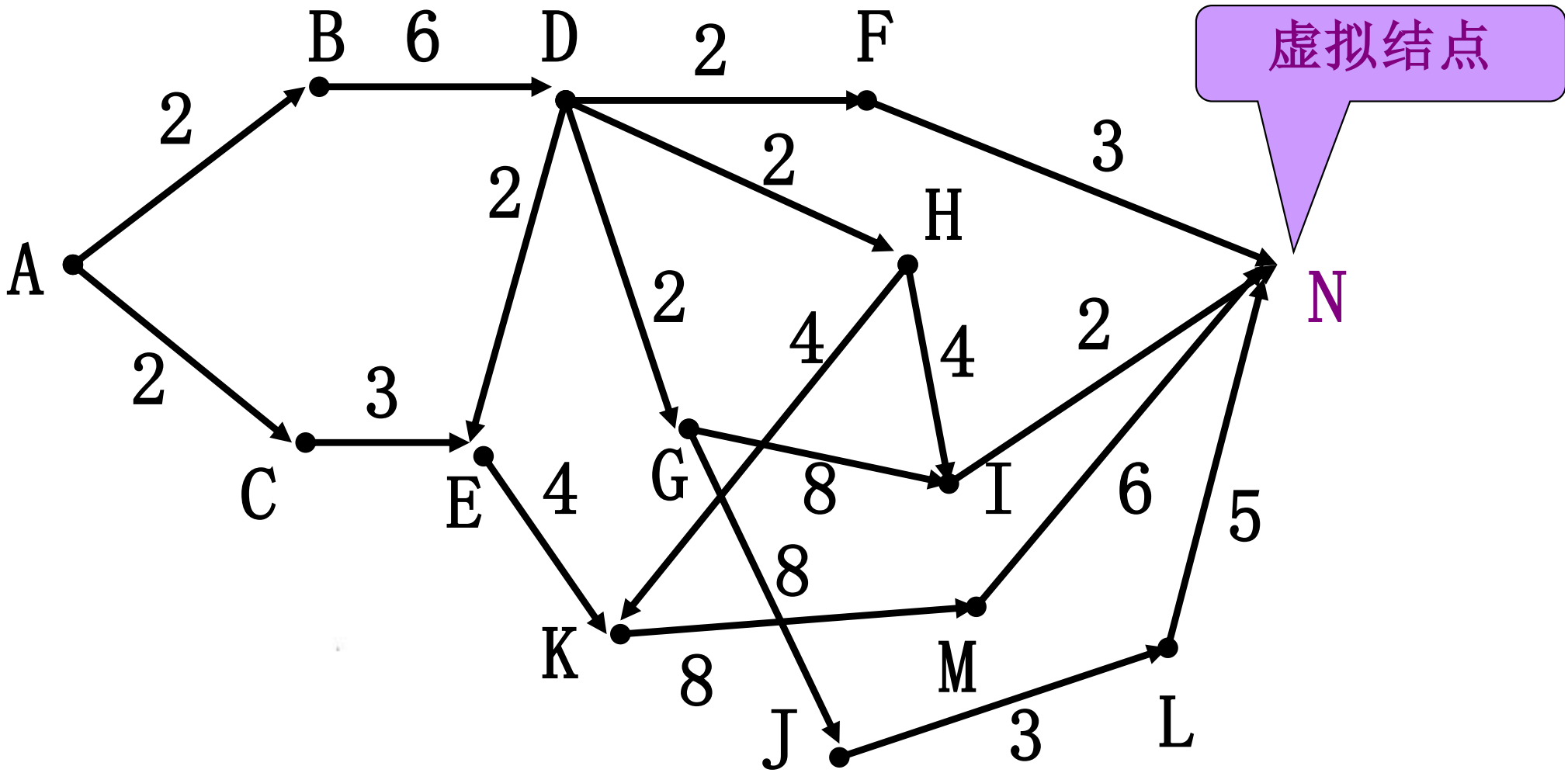
例 一项工程由13道工序组成，所需时间(单位：天)及先行工序如下表所示(P172).

工序序号	A	B	C	D	E	F	G	H	I
所需时间	2	6	3	2	4	3	8	4	2
先行工序	—	A	A	B	C,D	D	D	D	G,H

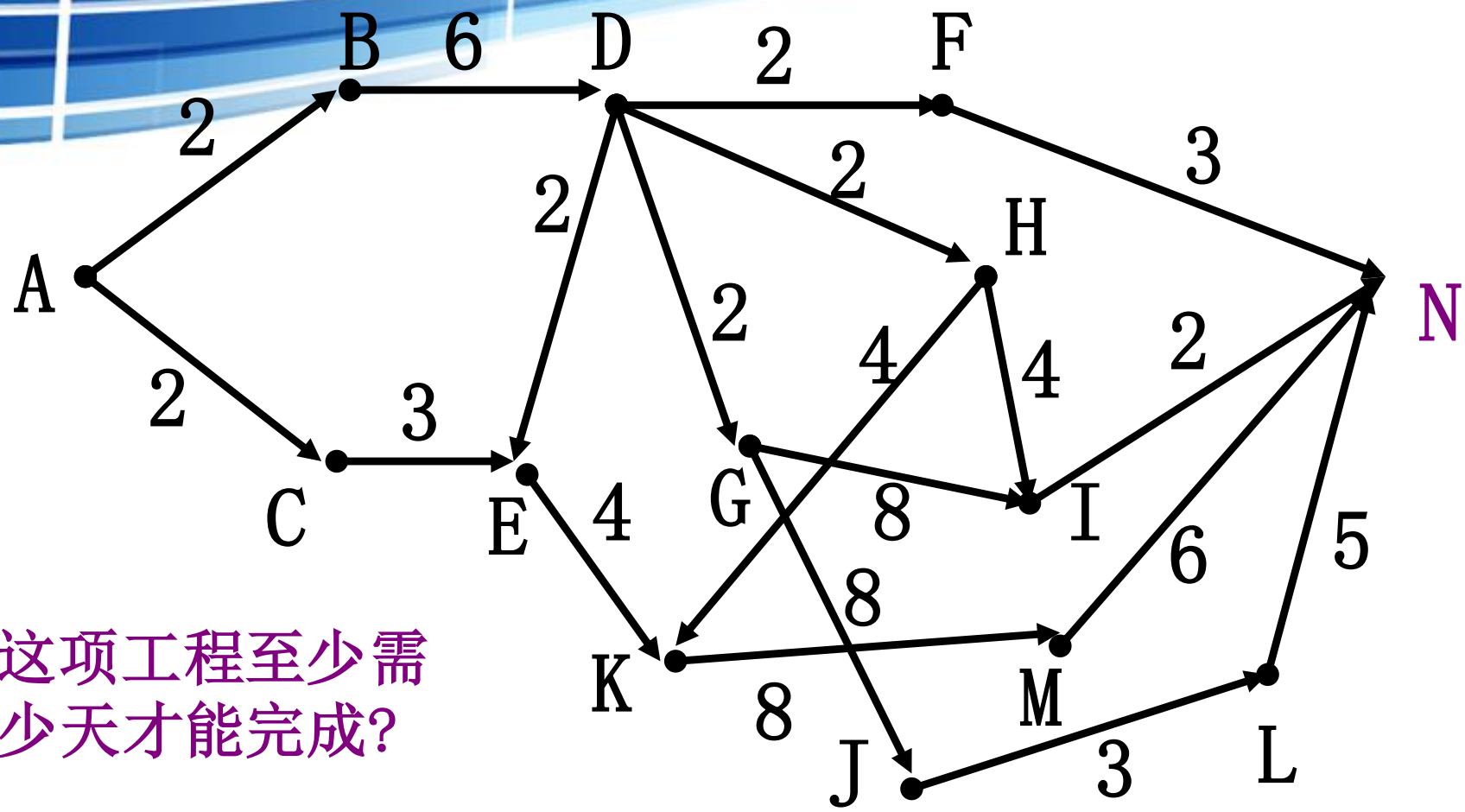
工序序号	J	K	L	M
所需时间	3	8	5	6
先行工序	G	H,E	J	K

试问这项工程至少需要多少天才能完成？那些工程不能延误？那些工程可以延误？最多可延误多少天？

先作出该工程的PT图. 由于除了工序A外，均有先行工序, 因此不必虚设虚拟结点v_0.



在PT图中,容易看出各工序先后完成的顺序及时间.



这项工程至少需要多少天才能完成?

就是要求A到N的最长路,此路径称为关键路径.

那些工程不能延误? 那些工程可以延误? 最多可延误多少天?

关键路径上的那些工程不能延误.

关键路径(最长路径)算法

定理 若有向图G中不存在有向回路, 则可以将G的结点重新编号为$u_1, u_2, \ldots, u_n$, 使得对任意的边$u_iu_j \in E(G)$, 都有$i < j$.

各工序最早启动时间算法步骤:

① 根据定理对结点重新编号为$u_1, u_2, \ldots, u_n$.

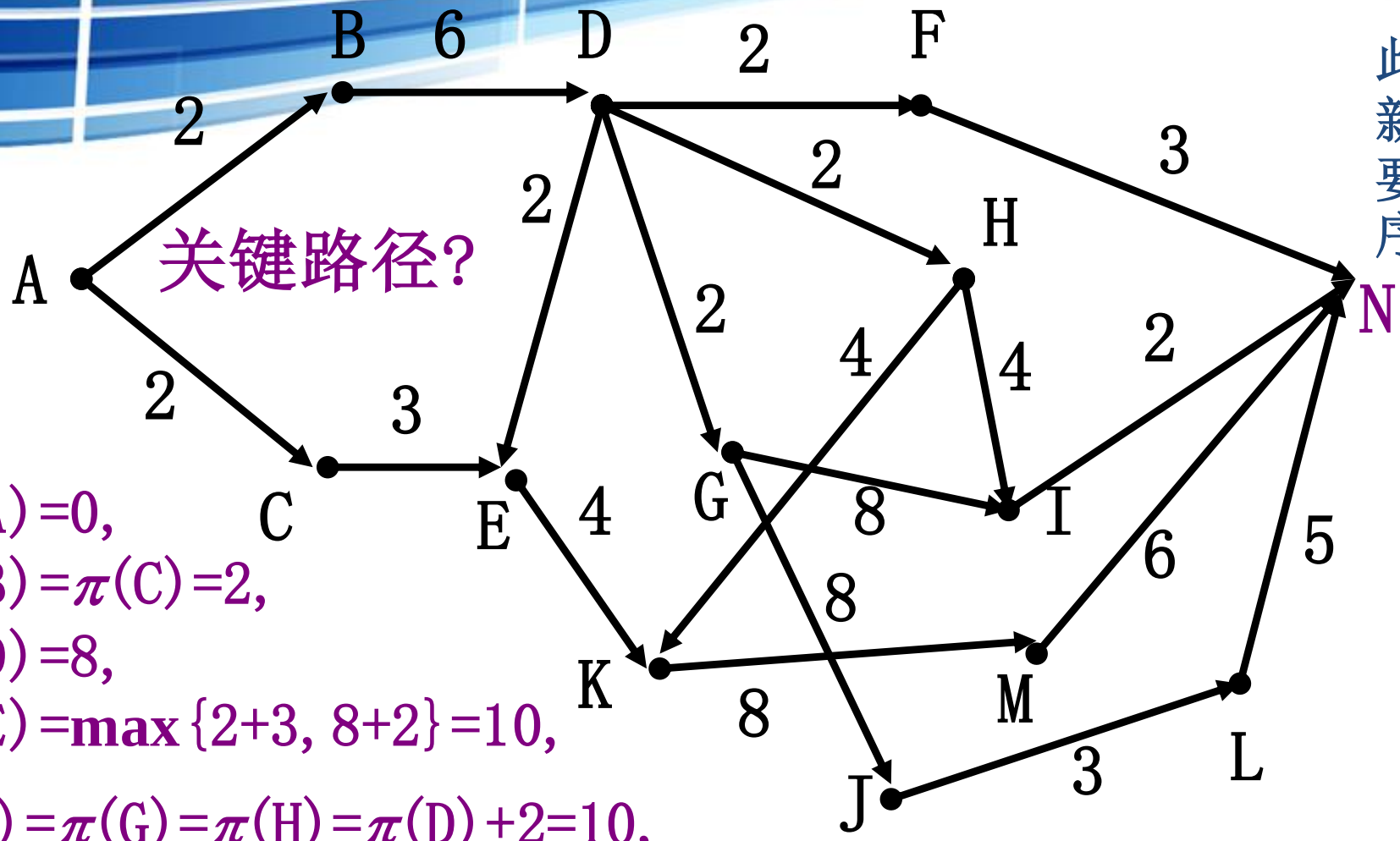
② 赋初值 $\pi(u_1) = 0$.

③ 依次更新 $\pi(u_j)$, $j = 2, 3, \ldots, n$.

$$\pi(u_j) = \max\{\pi(u_i) + \omega(u_i, u_j) \mid u_iu_j \in E(G)\}.$$

④ 结束.

其中$\pi(u_j)$表示工序u_j最早启动时间, 而$\pi(u_n)$即$\pi(v_n)$是整个工程完工所需的最短时间.



此例不必重新编号，只要按字母顺序即可.

$\pi(A)=0,$

$\pi(B)=\pi(C)=2,$

$\pi(D)=8,$

$\pi(E)=\max\{2+3,\ 8+2\}=10,$

$\pi(F)=\pi(G)=\pi(H)=\pi(D)+2=10,$

$\pi(I)=\max\{\pi(G)+8,\ \pi(H)+4\}=18,$

$\pi(K)=\max\{\pi(E)+4,\ \pi(H)+4\}=14,$

$\pi(J)=\pi(G)+8=18,$

$\pi(L)=\pi(J)+3=21,$

$\pi(M)=\pi(K)+8=22,$

$\pi(N)=\max\{\pi(F)+3,\ \pi(I)+2,\ \pi(L)+5,\ \pi(M)+6\}=28.$

通过以上计算表明：

这项工程至少需要28天才能完成.

关键路径(最长路径)：

A→B→D→E→K→M→N

A→B→D→H→K→M→N

工序A, B, D, E, H, K, M不能延误, 否则将影响工程的完成.

但是对于不在关键路径上的工序，是否允许延误？如果允许，最多能够延误多长时间呢？

各工序允许延误时间$t(u_j)$等于各工序最晚启动时间$\tau(u_j)$减去各工序最早启动时间$\pi(u_j)$.

即 $t(u_j)=\tau(u_j)-\pi(u_j)$.

最晚启动时间算法步骤(已知结点重新编号)：

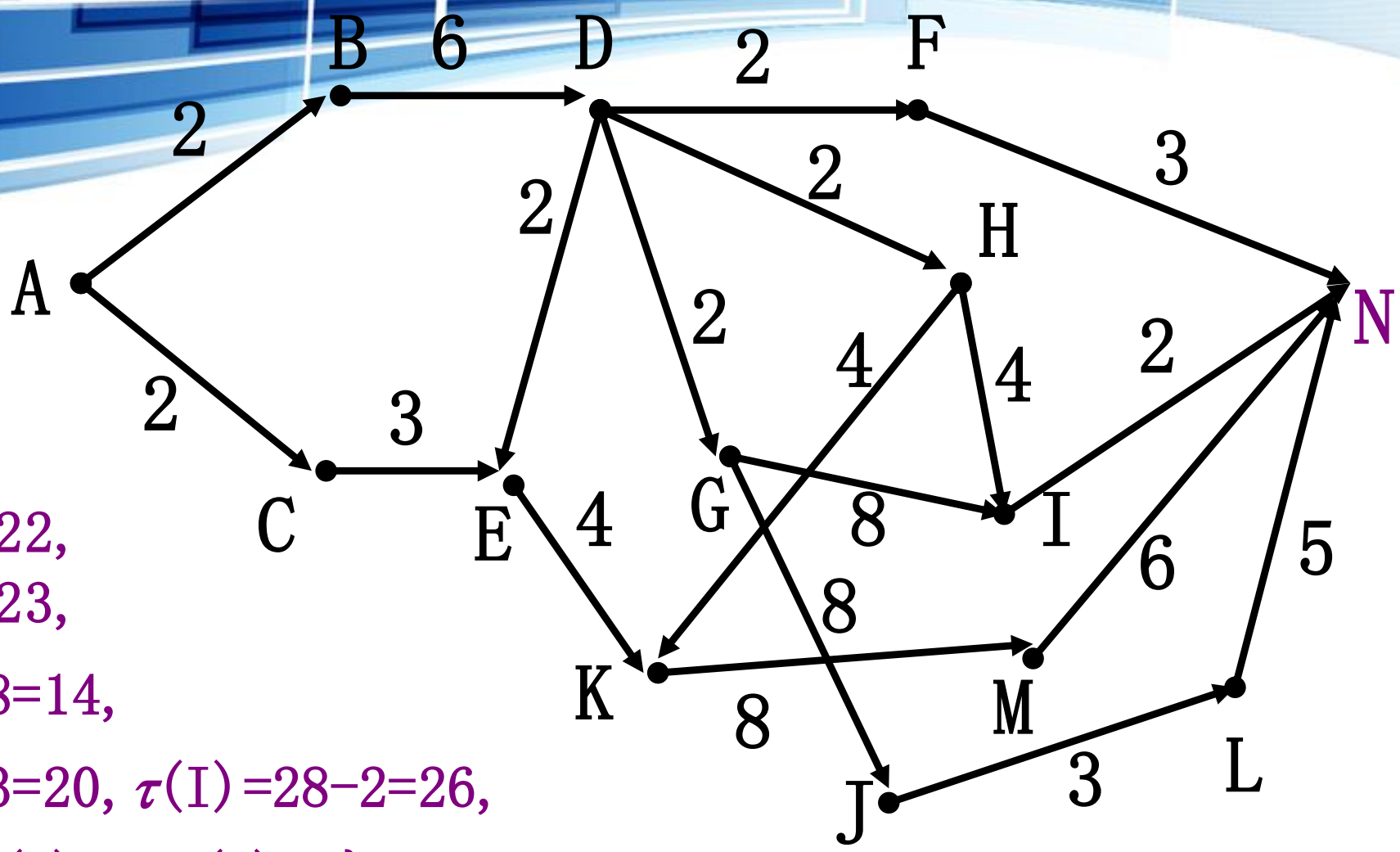
① 赋初值 $\tau(u_n)=\pi(u_n)$.

② 更新 $\tau(u_j)$, $j=n-1, n-2, \ldots, 1$.

$$\tau(u_j)=\min\{\tau(u_i)-\omega(u_i,u_j)\mid u_iu_j\in E(G)\}.$$

③ 结束.

顺便提一句, 根据工序u_j允许延误时间$t(u_j)$是否为0, 可判断该工序是否在关键路径上.



$\tau(N)=28,$

$\tau(M)=28-6=22,$

$\tau(L)=28-5=23,$

$\tau(K)=\tau(M)-8=14,$

$\tau(J)=\tau(L)-3=20,\ \tau(I)=28-2=26,$

$\tau(H)=\min\{\tau(K)-4,\ \tau(I)-4\}=10,\quad \tau(F)=28-3=25,$

$\tau(G)=\min\{\tau(J)-8,\ \tau(I)-8\}=12,\quad \tau(E)=\tau(K)-4=10,$

$\tau(D)=\min\{\tau(E)-2,\ \tau(F)-2,\ \tau(G)-2,\ \tau(H)-2\}=8,$

$\tau(C)=\tau(E)-3=7,\ \tau(B)=\tau(D)-6=2,\ \tau(A)=0.$

各工序允许延误时间如下：

$t(A)=t(B)=t(D)=t(E)=t(H)=t(K)=t(M)=0$,
$t(C)=5$, $t(F)=15$, $t(G)=2$, $t(I)=8$, $t(J)=2$,
$t(L)=2$.

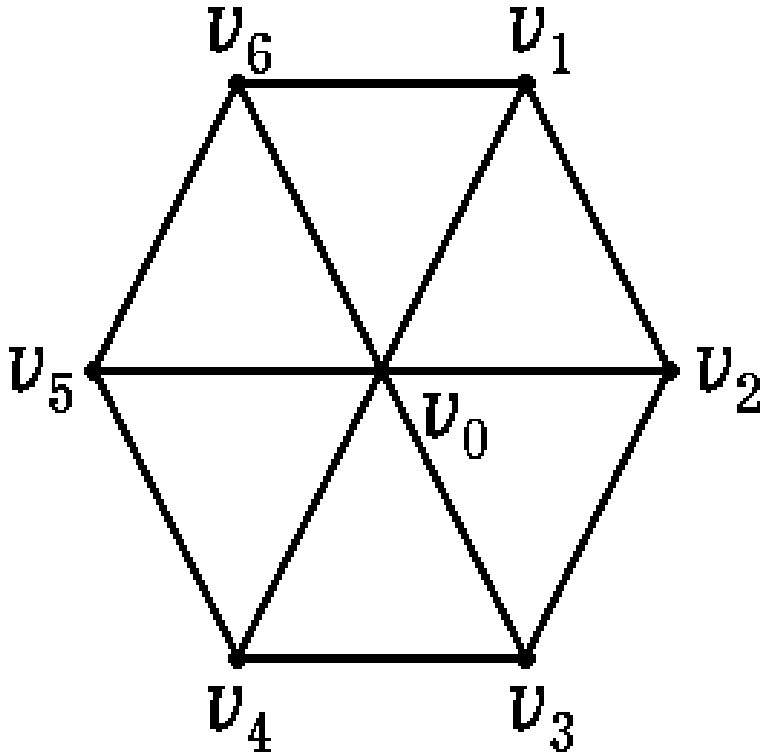
8、系统监控模型

定义1 设图$G=(V,E)$, $K\subset V$如果图G的每条边都至少有一个顶点在K中,则称K是G的一个**点覆盖**.

若G的一个点覆盖中任意去掉一个点后不再是点覆盖,则称此点覆盖是G的一个**极小点覆盖**.

顶点数最少的点覆盖,称为G的**最小点覆盖**.

例如,右图中,



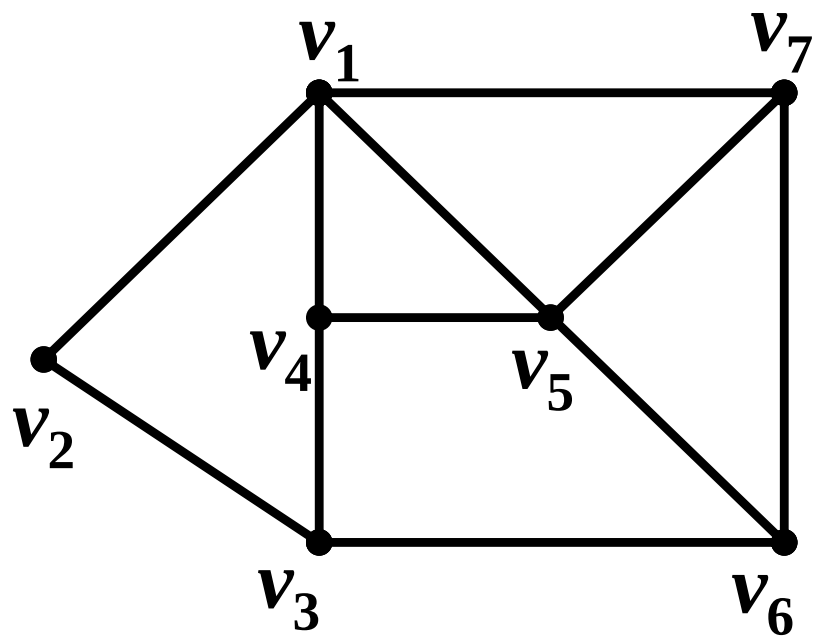
$\{v_0, v_2, v_3, v_5, v_6\}$等都是极小点覆盖.

$\{v_0, v_1, v_3, v_5\}$, $\{v_0, v_2, v_4, v_6\}$ 都是最小点覆盖.

系统监控问题之一

假设$v_1, v_2, \cdots, v_7$是7个哨所, 监视着11条路段(如下图所示), 为节省人力, 问至少需要在几个哨所派人站岗, 就可以监视全部路段?

这就是要求最小点覆盖问题.



$\{v_1, v_3, v_5, v_6\}$和$\{v_1, v_3, v_5, v_7\}$都是最小点覆盖, 所以至少需要在4个哨所派人站岗来监视全部路段.

到目前为止, 还没有找到求最小点覆盖的有效算法, 即多项式时间算法(算法步数不超过n^c, n为G的顶点数, c为常数). 有一些启发式近似算法.

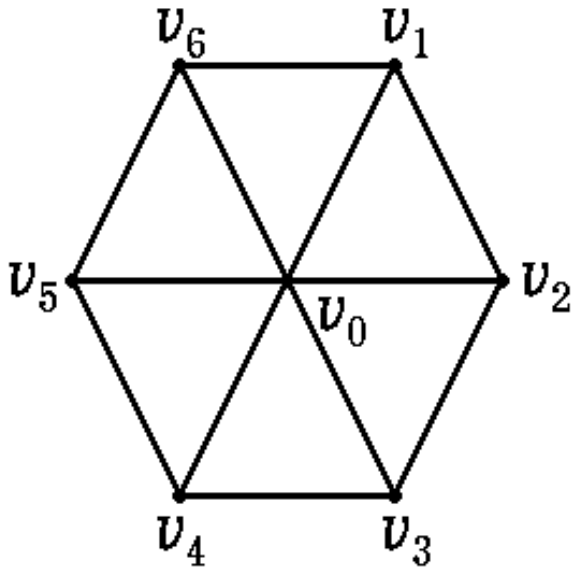
最大独立点集

定义2 设图$G=(V,E)$, $I\subset V$如果I中任意两个顶点在G中都不相邻, 则称I是G的一个**独立点集**.

若G的一个独立点集中, 任意添加一个点后不再是独立点集, 则称此独立点集是G的一个**极大独立点集**.

顶点数最多的独立点集, 称为G的**最大独立点集**.

例如, 右图中,



$\{v_1, v_4\}$等都是极大独立点集.

$\{v_1, v_3, v_5\}$, $\{v_2, v_2, v_6\}$ 是最大独立点集.

最小控制集

定义3 设图$G=(V,E)$, $D\subset V$如果$\forall v\in V$, 要么$v\in D$, 要么v与D的某个点相邻, 则称D是G的一个**控制集**.

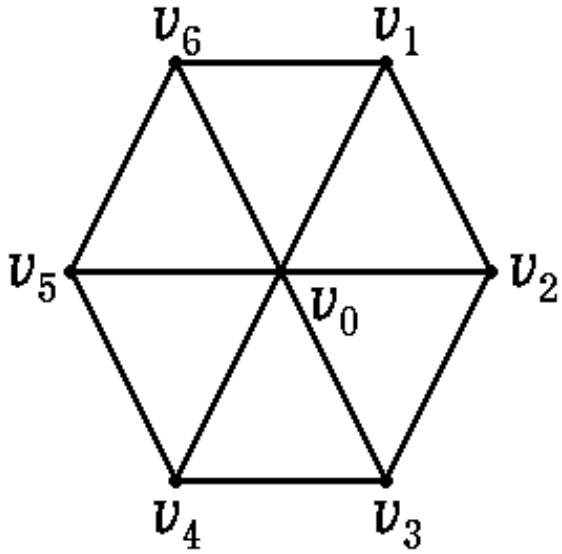
若G的一个控制集中任意去掉一个点后不再是控制集, 则称此控制集是G的一个**极小控制集**.

顶点数最少的控制集, 称为G的**最小控制集**.

例如, 右图中,

$\{v_1, v_3, v_5\}$是极小控制集,

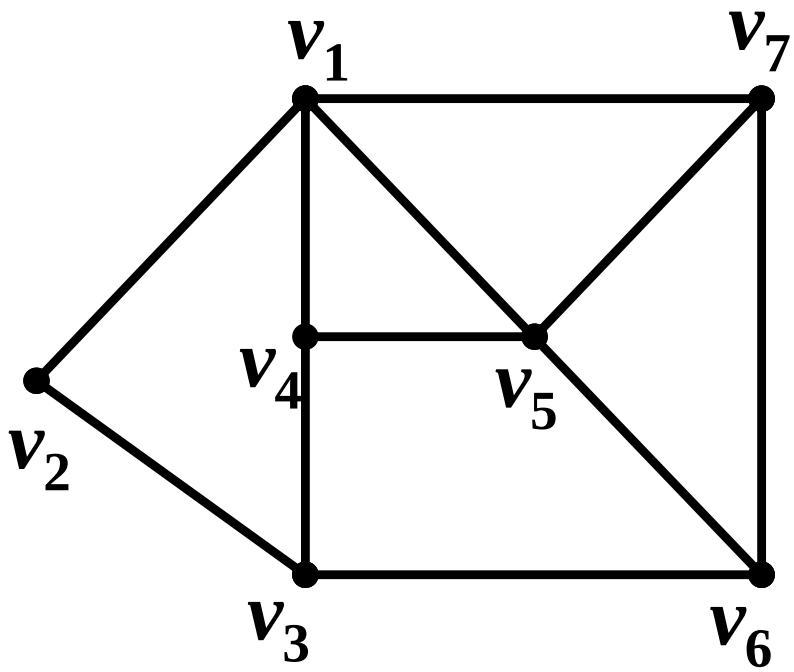
$\{v_0\}$是最小控制集.



系统监控问题之二

假设下图代表一指挥系统, 顶点$v_1, v_2, \cdots, v_7$表示被指挥的单位, 边表示可以直接下达命令的通信线路. 欲在某些单位建立指挥站, 以便可以通过指挥站直接给各单位下达命令, 问至少需要建立几个指挥站?

这就是要求最小控制集问题.



$\{v_1, v_3\}$, $\{v_3, v_5\}$等都是最小控制集, 所以至少需要在2个单位建立指挥站.

到目前为止, 还没有找到求最小控制集的有效算法. .

最小点覆盖、最大独立点集和最小控制集的关系

定理1 设无向图$G=(V, E)$中无孤立点(不与任何边关联的点), 若D是G中极大独立点集, 则D是G中极小控制集.

定理2 设无向图$G=(V, E)$中无孤立点, $K \subset V$, 则K是G的点覆盖当且仅当$K^c=V \backslash K$是G的独立点集.

推论 设无向图$G=(V, E)$中无孤立点, $K \subset V$, 则K是G的最小(极小)点覆盖当且仅当$K^c=V \backslash K$是G的最大(极大)独立点集.

9、着色模型

已知图$G=(V, E)$, 对图G的所有顶点进行着色时, 要求相邻的两顶点的颜色不一样, 问至少需要几种颜色?

这就是所谓的顶点着色问题.

若对图G的所有边进行着色时, 要求相邻的两条边的颜色不一样, 问至少需要几种颜色?

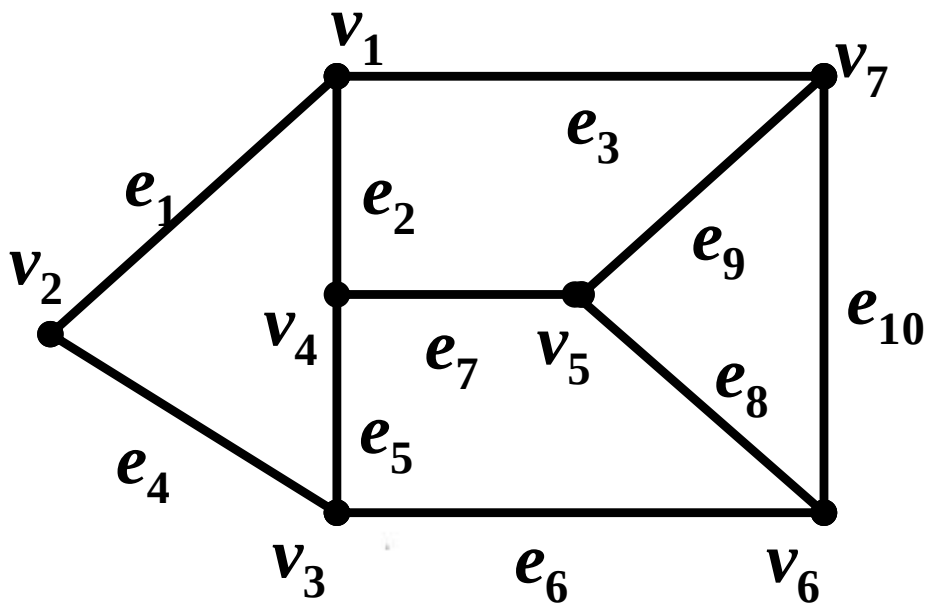
这就是所谓的边着色问题.

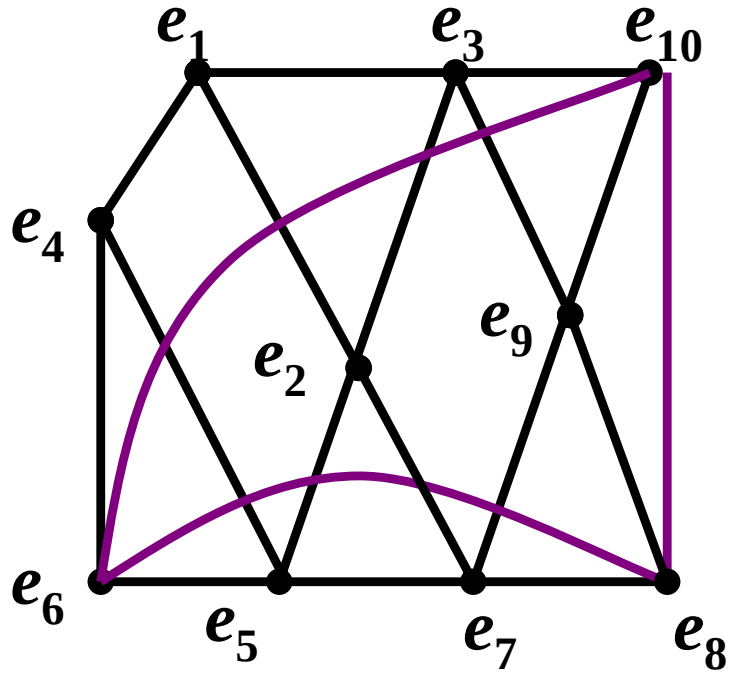
若已知图G为平面图，将平面图嵌入平面，对区域进行着色，并要求任意相邻的两个区域颜色不一样，问至少有需要几种颜色？

这就是所谓的区域着色问题.

对偶图

将原图中的点化为边, 边化为点即可. 这样得到的图称为原图的**对偶图**. 下面图中, 右图是左图的对偶图.



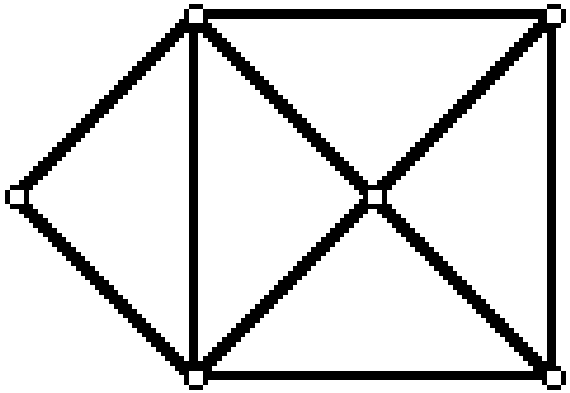


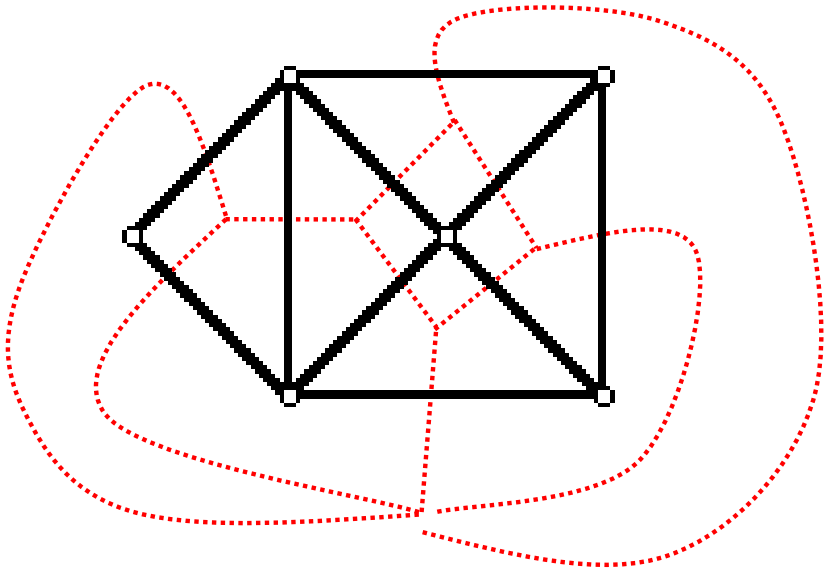
不过还原比较困难, 右图中黄色的边对应左图中的顶点v_6.

因此我们以后只讨论图的**点着色**问题, 简称**着色**(相邻的两顶点的颜色不一样).

对偶图

平面图的对偶图，将原图中的区域化为点，相邻两个区域中的点用边相连即可．这样得到的图称为原图的**对偶图**.





利用对偶图，可以把地图着色问题转化为对其对偶图的结点的着色问题。如果没有两个邻接的结点的颜色相同，就成为给地图正常着色。

物资储存问题

一家公司制造n种化学制品$A_1, A_2, \cdots, A_n$，其中有些化学制品若放在一起则可能产生危险，如引发爆炸或产生毒气等，称这样的化学制品是不相容的. 为安全起见，在储存这些化学制品时，不相容的不能放在同一储存室内. 问至少需要多少个储存室才能存放这种化学制品？

今作图G，用顶点$v_1, v_2, \cdots, v_n$分别表示n种化学制品，顶点v_i与v_j相邻，当且仅当化学制品A_i与A_j不相容.

于是储存问题就化为对图G的顶点着色问题，对图G的顶点最少着色数目便是最少需要的储存室数.

着色方法

对图$G=(V, E)$的顶点进行着色所需最少的颜色数目用$\chi(G)$表示, 称为图G的**色数**.

定理 若图$G=(V, E)$, $d=\max\{d(v)\mid v\in V\}$, 则$\chi(G)\leq d+1$.

这个定理给出了色数的上界. 着色算法目前还没有找到有效算法, 下面给出一种近似算法--最大度数优先的Welsh - Powell算法.

这个算法给出了一个较好的着色方法, 但不是最有效的方法, 即所用的颜色数不一定是最少的.

最大度数优先的Welsh - Powell算法

设$G=(V, E)$, $V=\{v_1, v_2, \dots, v_n\}$, 且不妨假设

$$d(v_1)\geqslant d(v_2)\geqslant \dots \geqslant d(v_n).$$

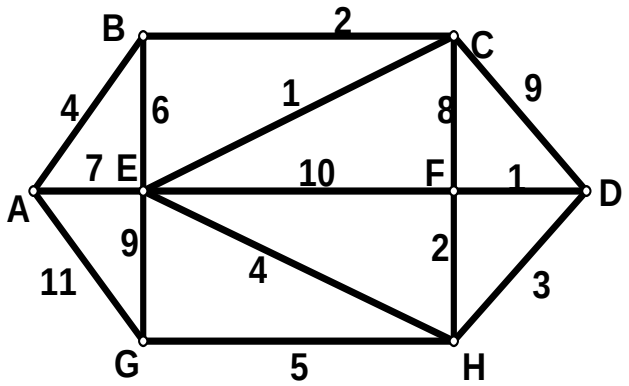
$c_1, c_2, \dots, c_n$为n种不同的颜色.

① 令有序集$C_i=\{c_1, c_2, \dots, c_i\}$, $i=1, 2, \dots, n$. $j=1$. 转向②.

② 给v_j着C_j的第一个颜色C_{j1}. 若$j=n$时, 停; 否则, 转向③.

③ $\forall k>j$, 若v_k和v_j相邻, 则令$C_k=C_k\setminus\{C_{j1}\}$. $j=j+1$, 转向②.

作　业



1. 给定图G，构造图G的邻接矩阵；
2. 分别用Dijkstra算法和Floyd算法求A到D的最短路径；
3. 思考：最短路径是否唯一呢？如果不唯一，该如何修改算法，求出所有最短路径呢？
4. 用Prim算法求图G的最小生成树；
5. *建立自己的图论程序库。
6. 将上述问题的程序及计算结果写成实验报告发到我的邮箱：feiwl@126.com ；邮件主题注明“姓名-2020图论作业”