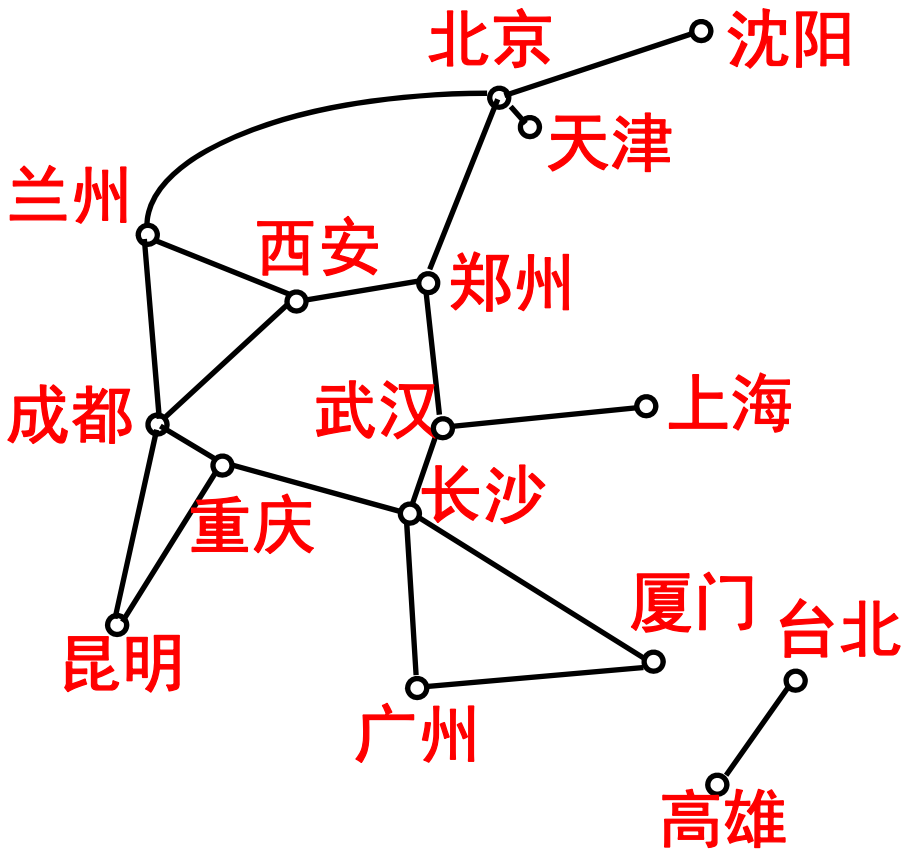
道路与回路

右图是中国铁路交通图的一部分，如果一个旅客要从成都乘火车到北京，那么他一定会经过其它车站；而旅客不可能从成都乘火车到达台北。这就引出了图的道路与连通的概念。



道路与回路

道路与回路是图论中两个重要的基本概念。我们所述定义一般来说既适合有向图，也适合无向图，否则，将加以说明或分开定义。

定义 给定图$G=<V,E>$中结点和边相继交错出现的序列$\Gamma=v_0e_1v_1e_2v_2\cdots e_kv_k$。

1. 若Γ中边e_i的两端点是v_{i-1}和v_i（G是有向图时要求v_{i-1}与v_i分别是e_i的始点和终点），$i=1,2,\cdots,k$，则称Γ为结点v_0到结点v_k的道路。v_0和v_k分别称为此道路的始点和终点，统称为道路的端点。道路中边的数目k称为此道路的长度。当$v_0=v_k$时，此道路称为回路。

道路与回路

2. 若道路中的所有边互不相同，则称此道路为简单道路或一条迹；若回路中的所有边互不相同，则称此回路为简单回路或一条闭迹。

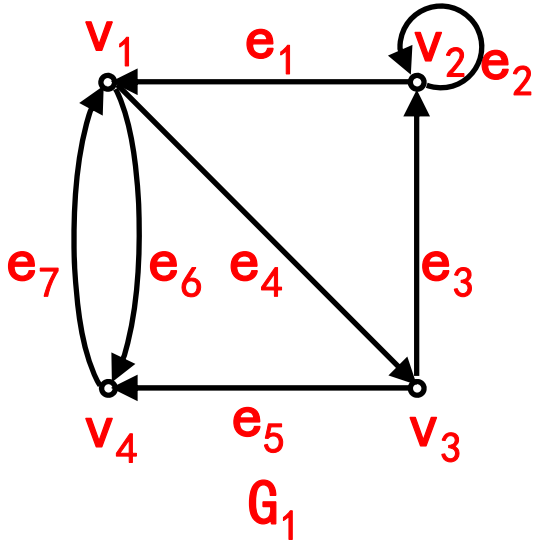
3. 若道路中的所有结点互不相同（从而所有边互不相同），则称此通路为初级道路；若回路中除$v_0=v_k$外的所有结点互不相同（从而所有边互不相同），则称此回路为初级回路。

说 明

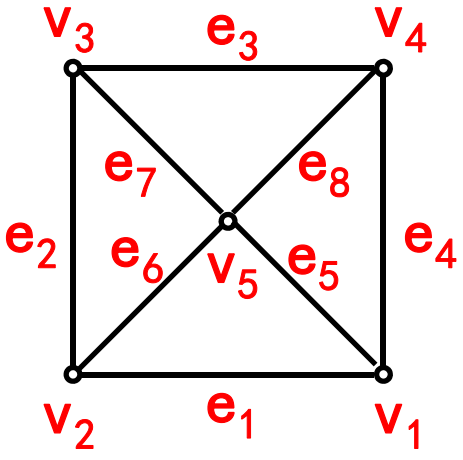
1. 回路是道路的特殊情况。因而，我们说某条道路，它可能是回路。但当我们说一初级道路时，一般是指它不是初级回路的情况。
2. 初级道路（回路）一定是简单道路（回路），但反之不真。因为没有重复的结点肯定没有重复的边，但没有重复的边不能保证一定没有重复的结点。
3. 在不会引起误解的情况下，一条道路$v_0e_1v_1e_2v_2\cdots e_nv_n$也可以用边的序列$e_1e_2\cdots e_n$来表示，这种表示方法对于有向图来说较为方便。在线图中，一条道路$v_0e_1v_1e_2v_2\cdots e_nv_n$也可以用结点的序列$v_0v_1v_2\cdots v_n$来表示。

道路与回路

判断下图G1中的回路$v_3e_5v_4e_7v_1e_4v_3e_3v_2e_1v_1e_4v_3$、$v_3e_3v_2e_2v_2e_1v_1e_4v_3$、$v_3e_3v_2e_1v_1e_4v_3$是否是简单回路、初级回路？图$G_2$中的道路$v_1e_1v_2e_6v_5e_7v_3e_2v_2e_6v_5e_8v_4$、$v_1e_5v_5e_7v_3e_2v_2e_6v_5e_8v_4$、$v_1e_1v_2e_6v_5e_7v_3e_3v_4$是否是简单道路、初级道路？并求其长度。



G_1



G_2

道路与回路

判断一条道（回）路是否是简单道（回）路、基本道（回）路，主要是看它有无重复的边、结点。

在图G_1中，$v_3e_5v_4e_7v_1e_4v_3e_3v_2e_1v_1e_4v_3$中有重复的边$e_4$，因此它不是简单回路，也不是初级回路。它是一条长度为6的回路；

$v_3e_3v_2e_2v_2e_1v_1e_4v_3$虽然没有重复的边，但有重复的结点$v_2$，因此只能是简单回路，长度为4，而不是初级回路；

而$v_3e_3v_2e_1v_1e_4v_3$中既没有重复的边，也没有重复的结点，因此既是初级回路，也是简单回路，长度为3；

道路与回路

在图G_2中，$v_1e_1v_2e_6v_5e_7v_3e_2v_2e_6v_5e_8v_4$中有重复的边$e_6$，因此，它既不是简单道路，也不是初级道路。它是长度为6的道路；

$v_1e_5v_5e_7v_3e_2v_2e_6v_5e_8v_4$虽然没有重复的边，但有重复的结点$v_5$，因此只能简单道路，长度为5，但不是初级道路；

$v_1e_1v_2e_6v_5e_7v_3e_3v_4$中既没有重复的边，也没有重复的结点，因此既是初级道路，也是简单道路，长度为4。

道路与回路

说明　在图G_1中，简单回路$v_3e_3v_2e_2v_2e_1v_1e_4v_3$既可以用边的序列$e_3e_2e_1e_4$来表示，也可以用结点的序列$v_3v_2v_2v_1v_3$来表示；

在图G_2中，简单道路$v_1e_5v_5e_7v_3e_2v_2e_6v_5e_8v_4$既可以用边的序列$e_5e_7e_2e_6e_8$来表示，也可以用结点的序列$v_1v_5v_3v_2v_5v_4$来表示。

无向图的连通性

定义 在图G = <V, E>中，v_i, $v_j \in V$。如果从v_i到v_j存在道路，则称v_i到v_j是可达的，否则称v_i到v_j不可达。规定：任何结点到自己都是可达的。

定义 若无向图G中的任何两个结点都是可达的，则称G是连通图，否则称G是非连通图或分离图。

无向完全图K_n（$n \geqslant 1$）都是连通图，而多于一个结点的零图都是非连通图。

无向图的连通性

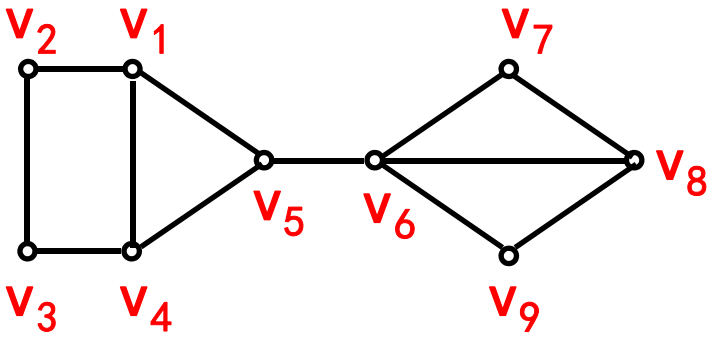
若连通子图H不是G的任何连通子图的真子集，则称H是G的极大连通子图，或称连通分支。用p(G)表示G中的连通分支个数。

显然，无向图G是连通图当且仅当p(G) = 1；每个结点和每条边都在且仅在一个连通分支中。

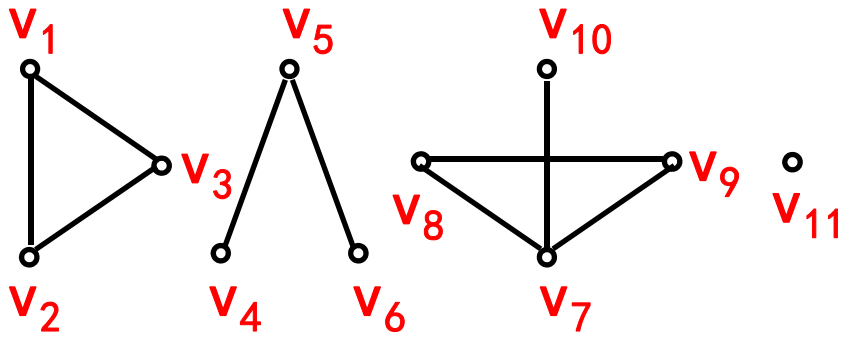
- 定义：若$G' = G - e$比G的连通支数多，则称e是G的割边。
  - 删去$e = (u,v)$，则u, v分属于不同的连通分支。

无向图的连通性

判断下图中图G_1和G_2的连通性，并求其连通分支个数。



G_1



G_2

解 G_1是连通图，G_2是非连通图。容易看出，G_1有1个连通分支；G_2中有4个连通分支，它们的结点集分别是$\{v_1, v_2, v_3\}$，$\{v_4, v_5, v_6\}$，$\{v_7, v_8, v_9, v_{10}\}$，$\{v_{11}\}$。

有向图的连通性

定义7.3.5　设G = <V, E>是一个有向图，

（1）略去G中所有有向边的方向得无向图G’，如果无向图G’是连通图，则称有向图G是连通图或称为弱连通图。否则称G是非连通图；

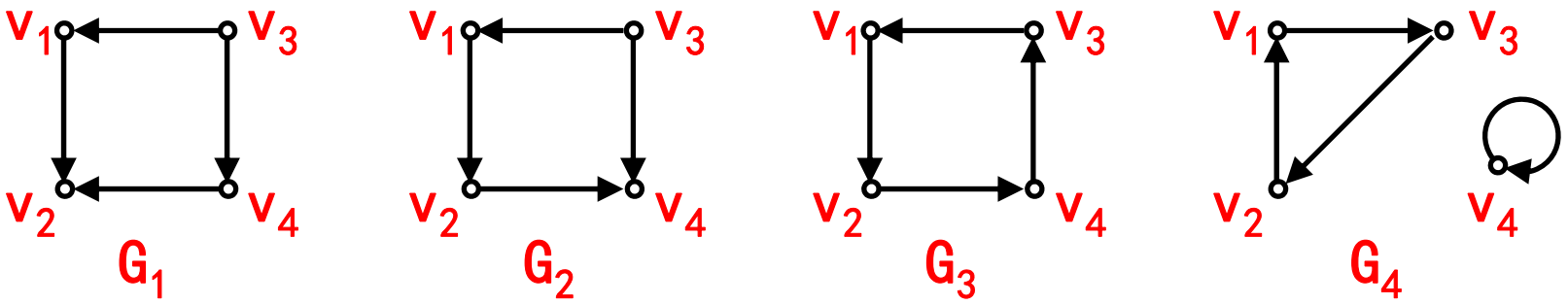
（2）若G中任何一对结点之间至少有一个结点到另一个结点是可达的，则称G是单向连通图；

（3）若G中任何一对结点之间都是相互可达的，则称G是强连通图。

若G是强连通图，则它必是单向连通图；若G是单向连通图，则它必是（弱）连通图。反之，均不成立。

有向图的连通性

判断下图中4个图的连通性。



分析　先看略去图中所有有向边的方向得无向图，容易看出G_1、G_2、G_3是连通的有向图，G_4是非连通的有向图。再看有向连通图中结点间的可达情况，G_1中v_1到v_4不可达，v_4到v_1不可达，所以G_1是弱连通图；G_2中任何一对结点之间至少有一个结点到另一个结点是可达的，所以G_2是单向连通图（当然它也是弱连通图）；G_3中任何一对结点之间都是相互可达的，所以G_3是强连通图（当然它也是单向连通图和弱连通图）。

有向图的连通性

定理　有向图G是强连通图的充分必要条件是G中存在一条经过所有结点的回路。

分析　只需要利用回路中的任二点相互可达和强连通的定义即可证明。

证明　**充分性：**如果G中存在一条经过所有结点的回路C，则G中任意二结点均在回路C上，所以G中任二结点都是相互可达的，因而G是强连通图。

有向图的连通性

必要性：设G是强连通图，那么G中任二结点均是相互可达的。不妨设G中的结点为$v_1, v_2, \cdots, v_n$，因为v_i到v_{i+1}是可达的，$i=1,2,\cdots,n-1$，且v_n到v_1是可达的，所以v_i到v_{i+1}存在通路，$i=1,2,\cdots,n-1$，且v_n到v_1存在通路。让这些连通首尾相接，则得一回路C。显然所有结点均在该回路中出现。

最短路径（补充）

（1） 赋权图

在图G的每条边上标上一个实数w (e)后, 称G为边赋权图。被标上的实数称为边的权值。

若H是赋权图G的一个子图，称 $W(H)=\sum_{e\in E(H)} w(e)$ 为子图H的权值。

权值的意义是广泛的。可以表示距离，可以表示交通运费，可以表示网络流量，在朋友关系图甚至可以表示友谊深度。但都可以抽象为距离。

（2） 赋权图中的最短路径

设G为边赋权图。u与v是G中两点，在连接u与v的所有路中，路中各边权值之和最小的路，称为u与v间的最短路径。

最短路径算法（补充）

Dijkstra算法（正权图的单源最短路径）

采用“贪心”策略，声明一个数组dis来保存源点到各个顶点的最短距离，和一个保存已经找到了最短路径的顶点的集合T；

初始时，原点 s 的路径权重被赋为 0 （dis[s] = 0）。若对于顶点 s 存在能直接到达的边（s,m），则把dis[m]设为w（s, m）,同时把所有其他（s不能直接到达的）顶点的路径长度设为无穷大。初始时，集合T只有顶点s；

然后，从dis数组选择最小值，则该值就是源点s到该值对应的顶点的最短路径，并且把该点加入到T中，此时完成一个顶点；

然后，看看新加入的顶点是否可以到达其他顶点，并且看看通过该顶点到达其他点的路径长度是否比源点直接到达短，如果是，那么就替换这些顶点在dis中的值；

然后，又从dis中找出最小值，重复上述动作，直到T中包含了图的所有顶点。

最短路径算法（补充）

例1 在一河岸有狼，羊和菜。摆渡人要将它们渡过河去，由于船太小，每次只能载一样东西。由于狼羊，羊菜不能单独相处。问摆渡人至少要多少次才能将其渡过河？

分析：人，狼，羊，菜所有组合形式为：

$$C_4^0 + C_4^1 + C_4^2 + C_4^3 + C_4^4 = 2^4 = 16$$

但是以下组合不能允许出现：

狼羊菜，羊菜，狼羊，人，人狼，人菜，共6种。

岸上只能允许出现10种组合：

人狼羊菜，人狼羊，人狼菜，人羊，空，菜，羊，狼，狼菜，人羊菜。

最短路径算法（补充）

每种情况用点表示；

两点连线，当且仅当两种情况可用载人(或加一物)的渡船相互转变；

每条边赋权为1；

于是，问题转化为求由顶点“人狼羊菜”到顶点“空”的一条最短路。

结果为：

(1) 人狼羊菜→狼菜→人狼菜→狼→人狼羊→羊→人羊→空；

(2) 人狼羊菜→狼菜→人狼菜→菜→人羊菜→羊→人羊→空。

最短路径算法（补充）

例2 某两三个量杯容量分别为8升、5升和3升，现8升的量杯装满了水，问怎样才能把水分成2个4升，求最少的操作次数。

解：设x_1,x_2,x_3分别表示8,5,3升量杯中的水量。则

$$x_1 + x_2 + x_3 = 8, x_1 \le 8, x_2 \le 5, x_3 \le 3.$$

容易算出(x_1,x_2,x_3) 的组合形式共24种。

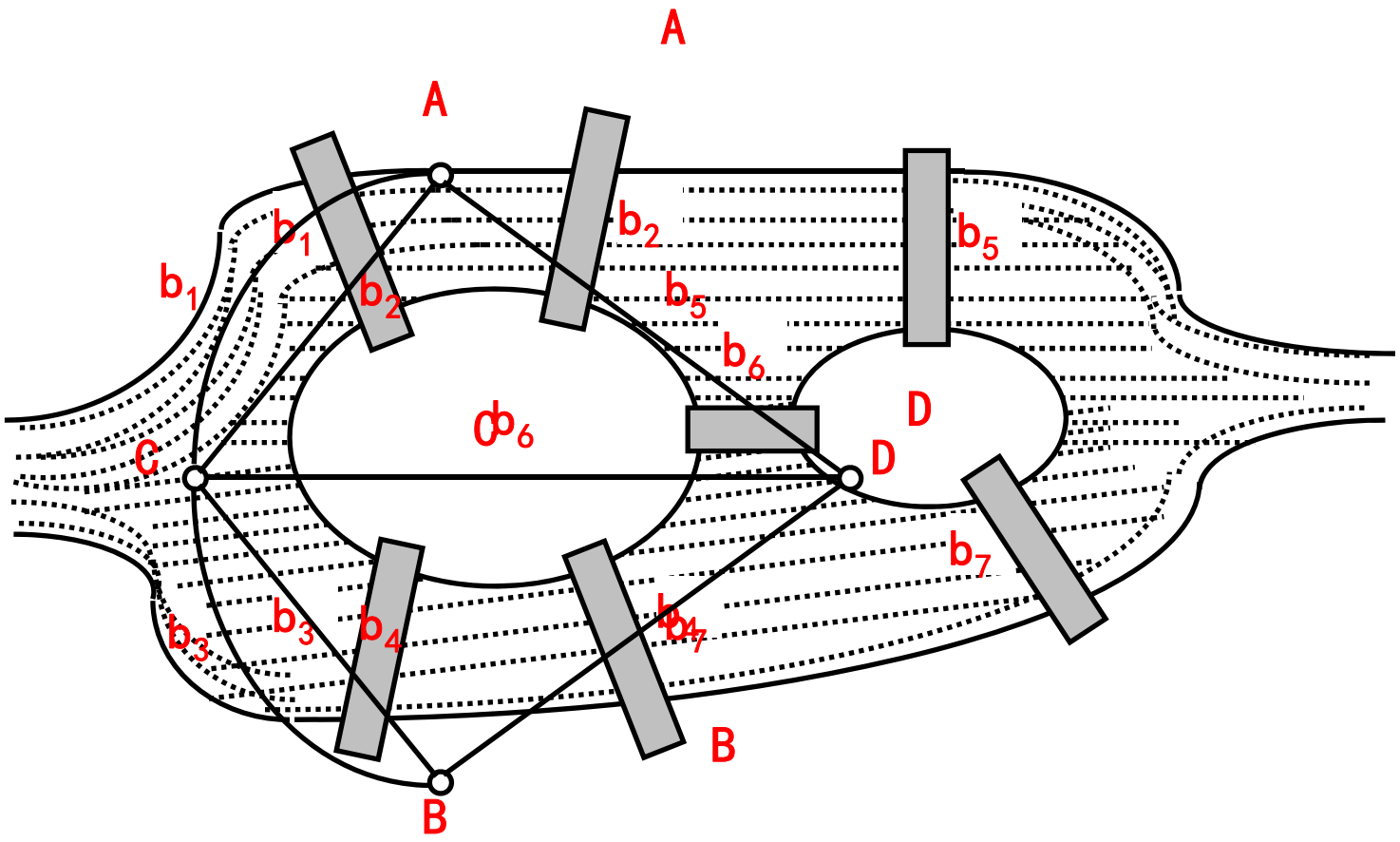
每种组合用一个点表示，两点连线，当且仅当可通过倒水的方式相互变换。

若各边赋权为1，则问题转化为在该图中求(8,0,0)到(4,4,0)的一条最短路。结果如下：

$$(8,0,0) \to (3,5,0) \to (3,2,3) \to (6,2,0) \to (6,0,2) \to (1,5,2) \to (1,4,3) \to (4,4,0).$$

欧拉图

欧拉图的引入与定义



欧拉图定义

定义　设G是无孤立结点的图，若存在一条道路(回路)，经过图中每条边一次且仅一次，则称此道路(回路)为该图的一条欧拉道路(回路)。具有欧拉回路的图称为欧拉图。

规定：平凡图为欧拉图。

以上定义既适合无向图，又适合有向图。

欧拉道路与欧拉回路

欧拉道路是经过图中所有边的道路中长度最短的道路，即为通过图中所有边的简单道路；

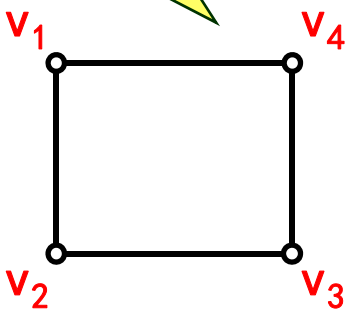
欧拉回路是经过图中所有边的回路中长度最短的回路，即为通过图中所有边的简单回路。

如果仅用边来描述，欧拉道路和欧拉回路就是图中所有边的一种全排列。

欧拉道路与欧拉回路

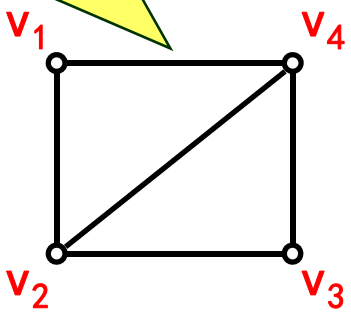
判断下面的6个图？是

欧拉图



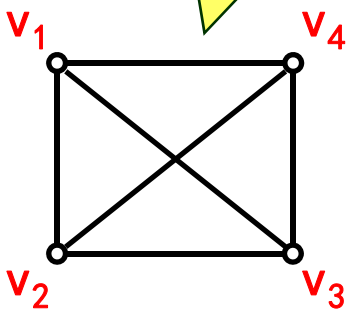
(a)

不是欧拉图，但存在欧拉道路



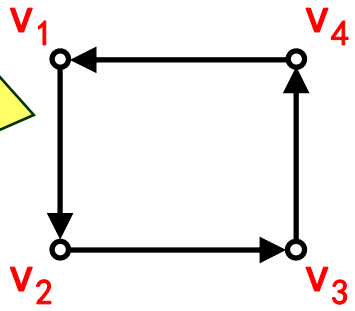
(b)

不存在欧拉道路

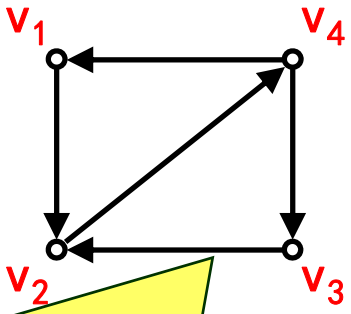


(c)

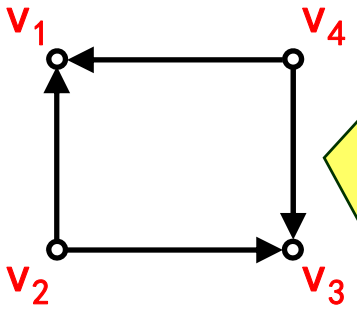
欧拉图



(d)



不是欧拉图，但存在欧拉道路



不存在欧拉道路

欧拉道路与欧拉回路的判定

定理　无向连通图G = <V, E>存在欧拉回路的充要条件是：G中所有结点的度数均为偶数。

推论　如果无向连通图G = <V, E>只有零个或两个奇度数结点，则G存在欧拉道路。

推论　如果有向连通图G中所有结点的入度等于出度，则G存在欧拉回路。

推论　如果有向连通图G除了两个结点以外，其余结点的入度等于出度，而这两个例外的结点，一个结点的入度比出度大1，另一个结点的出度比入度大1，则G存在欧拉道路。

欧拉道路与欧拉回路的判定

对任意给定的无向连通图，只需通过对图中各结点度数的计算，就可知它是否存在欧拉道路及欧拉回路，从而知道它是否为欧拉图；对任意给定的有向连通图，只需通过对图中各结点出度与入度的计算，就可知它是否存在欧拉道路及欧拉回路，从而知道它是否为欧拉图。

利用这项准则，很容易判断出哥尼斯堡七桥问题是无解的，因为它所对应的图中所有4个结点的度数均为奇数。

欧拉回路的求解

对一个无向欧拉图G，如何找到一条欧拉回路？

方法一：构造法（逐步插入回路）

方法二：Fleury算法（能不走割边就不走割边）

1）任取G中一结点v_0，令$P_0=v_0$

2）假设沿$P_i=v_0e_1v_2e_2\ldots e_iv_i$走到结点$v_i$，按下面的方法，从$E(G)-\{e_1e_2\ldots e_i\}$中选取$e_{i+1}$

a) e_{i+1}与v_i相关联

b) 非割边优先（除非没有别的边可选，否则e_{i+1}不是$Gi=G-\{e_1e_2\ldots e_i\}$的割边）　　割边：删除该边之后图变为非连通图

c) 设$e_{i+1}=(v_i, v_{i+1})$，将e_{i+1} v_{i+1}加入P_i得到P_{i+1}。令i=i+1，返回2）

3）当 2）不能再进行时算法停止

欧拉图的应用

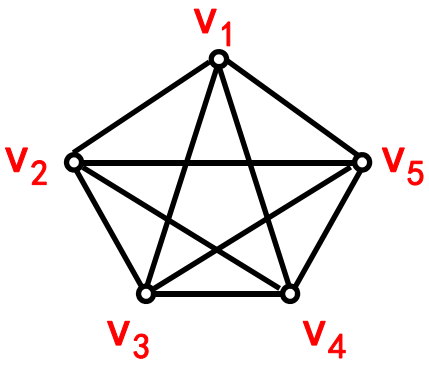
1、一笔画问题

所谓“一笔画问题”就是画一个图形，笔不离纸，每条边只画一次而不许重复，画完该图。

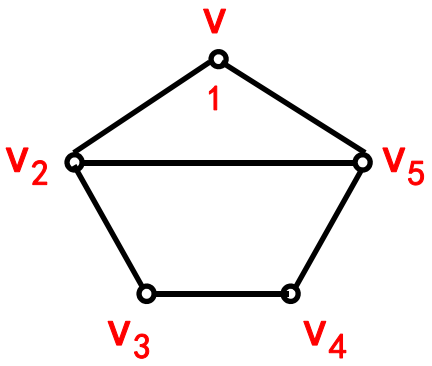
“一笔画问题”本质上就是一个无向图是否存在欧拉道路(回路)的问题。如果该图为欧拉图，则能够一笔画完该图，并且笔又回到出发点；如果该图只存在欧拉道路，则能够一笔画完该图，但笔回不到出发点；如果该图中不存在欧拉道路，则不能一笔画完该图。

欧拉图的应用

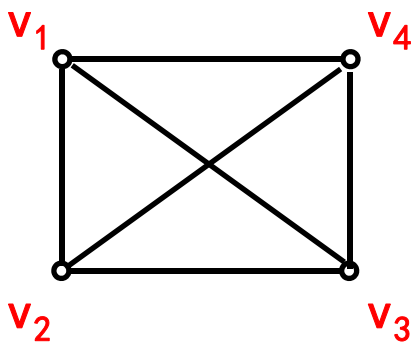
图中的三个图能否一笔画？为什么？



(a)



(b)

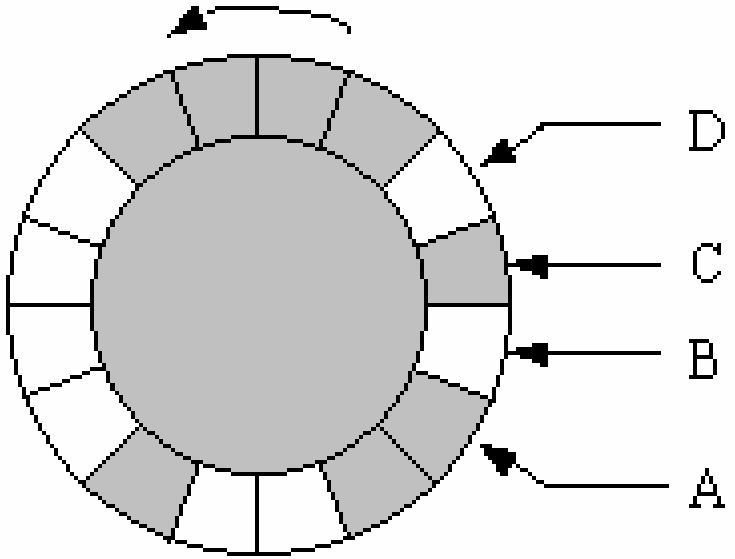


(c)

解　因为图(a)和(b)中分别有0个和2个奇度数结点，所以它们分别是欧拉图和存在欧拉道路，因此能够一笔画，并且在(a)中笔能回到出发点，而(b)中笔不能回到出发点。图c中有4个度数为3的结点，所以不存在欧拉道路，因此不能一笔画。

欧拉图的应用

2、计算机轮鼓设计

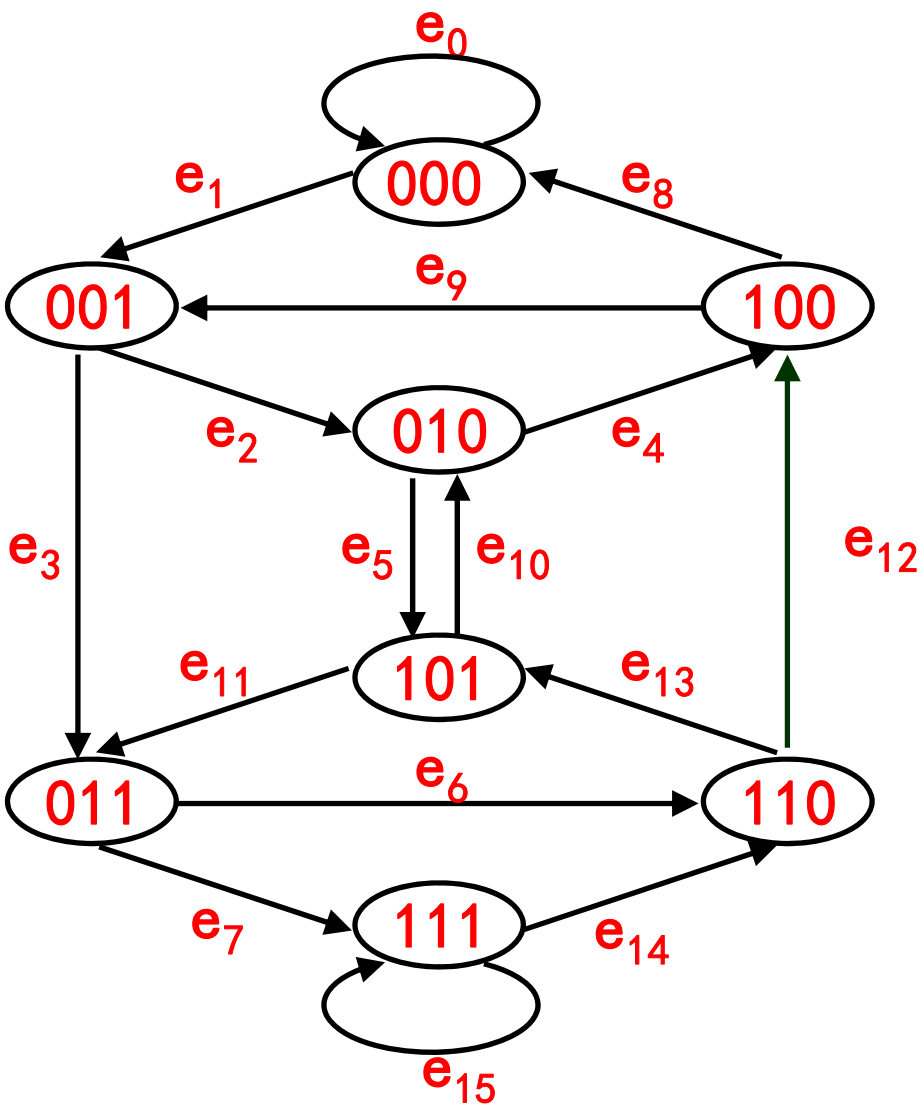


假设一个旋转鼓的表面被等分为16个部分，如图所示，其中每一部分分别由导体或绝缘体构成，图中阴影部分表示导体，空白部分表示绝缘体，导体部分给出信号1，绝缘体部分给出信号0。根据鼓轮转动时所处的位置，四个触头A、B、C、D将获得一定的信息。因此，鼓轮的位置可用二进制信号表示。试问如何选取鼓轮16个部分的材料才能使鼓轮每转过一个部分得到一个不同的二进制信号，即每转一周，能得到0000到1111的16个数。

欧拉图的应用

问题的等价描述：把如果从状态$a_1a_2a_3a_4$逆时针旋转一个扇面，新的输出$a_2a_3a_4a_5$，其中三位数字不变，因此可以用8个结点表示从000到111这8个二进制数。

欧拉图的应用



e_0=0000　e_8=1000

e_1=0001　e_9=1001

e_2=0010　e_{10}=1010

e_3=0011　e_{11}=1011

e_4=0100　e_{12}=1100

e_5=0101　e_{13}=1101

e_6=0110　e_{14}=1110

e_7=0111　e_{15}=1111

欧拉图的应用

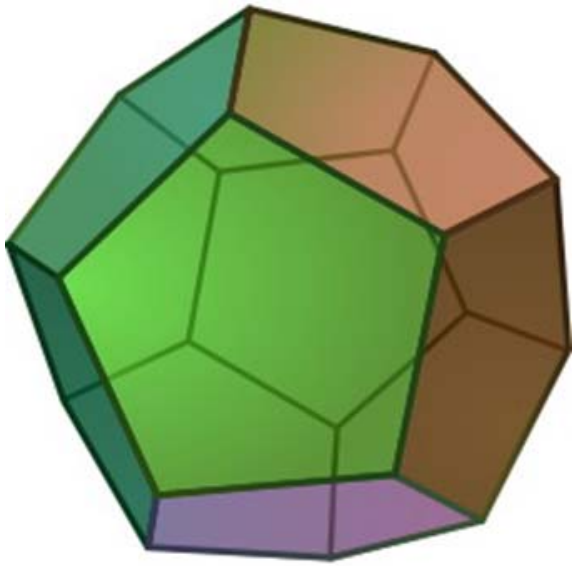
问题转化为在这个有向图中找一条欧拉回路。

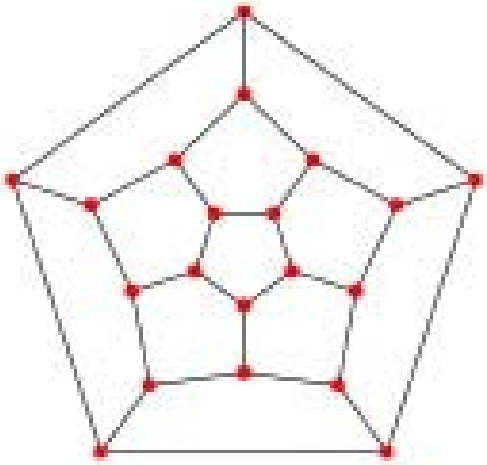
这个有向图中8个结点的出度和入度都是2，因此存在欧拉回路。

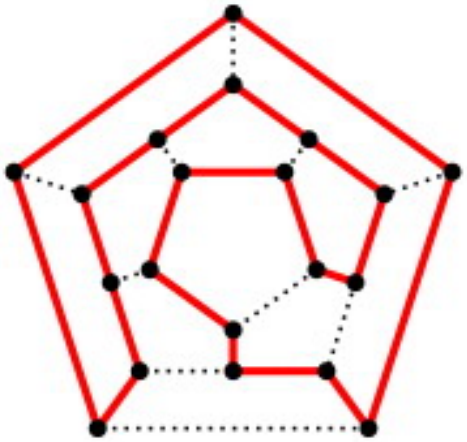
例如 $e_0e_1e_2e_4e_9e_3e_6e_{13}e_{10}e_5e_{11}e_7e_{15}e_{14}e_{12}e_8$ 就是一条欧拉回路。根据邻接边的标号记法，这16个二进制数的可写成对应的二进制序列0000100110101111，把这个序列排成一个圆圈，与所求的鼓轮相对应，就得到鼓轮的设计。

哈密顿图

哈密顿图的引入与定义







哈密顿图定义

定义　经过图中每个结点一次且仅一次的道路(回路)称为哈密顿道路(回路)。存在哈密顿回路的图称为哈密顿图。

规定：平凡图为哈密顿图。

以上定义既适合无向图，又适合有向图。

对哈密顿回路问题：

要求$V(G) = n \geq 3$

只需考虑简单图,因为重边和自环不起作用

对哈密顿回路的判定很困难,没有发现充分必要的条件,只有若干充分条件

哈密顿道路与哈密顿回路

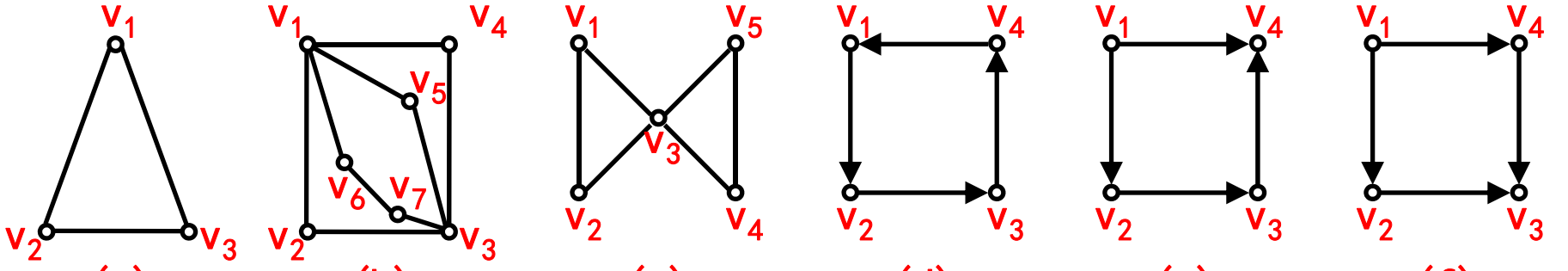
- 哈密顿道路是经过图中所有结点的道路中长度最短的道路，即为通过图中所有结点的基本道路；
- 哈密顿回路是经过图中所有结点的回路中长度最短的回路，即为通过图中所有结点的基本回路。

如果我们仅用结点来描述的话

- 哈密顿道路就是图中所有结点的一种全排列。
- 哈密顿回路就是图中所有结点的一种全排列再加上该排列中第一个结点的一种排列。

哈密顿道路与哈密顿回路

判断下面的6个图中，是否是哈密顿图？是否存在哈密顿道路？



哈密顿图

不存在哈密顿道路

不是哈密顿图，但存在哈密顿道路

哈密顿图

不是哈密顿图，但存在哈密顿道路

不存在哈密顿道路

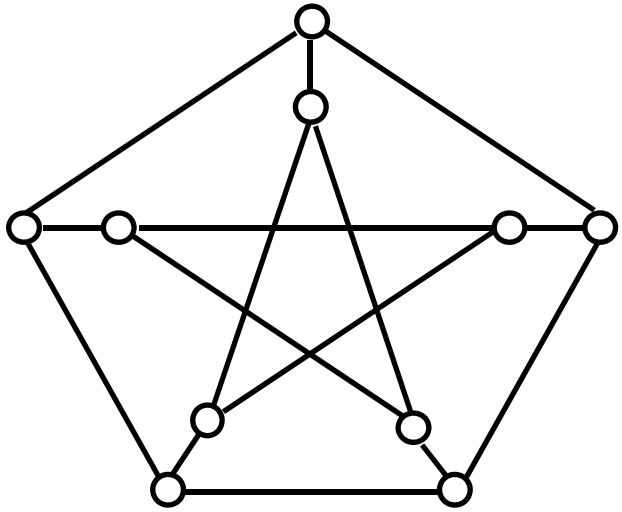
哈密顿图的判定

定理 设图G = <V, E>是哈密顿图，V_1是V的任意非空子集，则

$$p(G-V_1) \leqslant |V_1|$$

其中$p(G-V_1)$是从G中删除V_1后所得到图的连通分支数。

注意：
上述定理给出的是哈密顿图的必要条件，而不是充分条件。

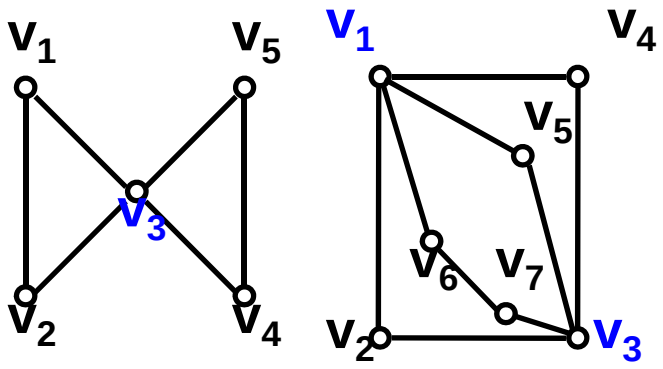


哈密顿图的判定

推论 设图G = <V, E>中存在哈密顿道路，则对V的任意非空子集V_1，都有

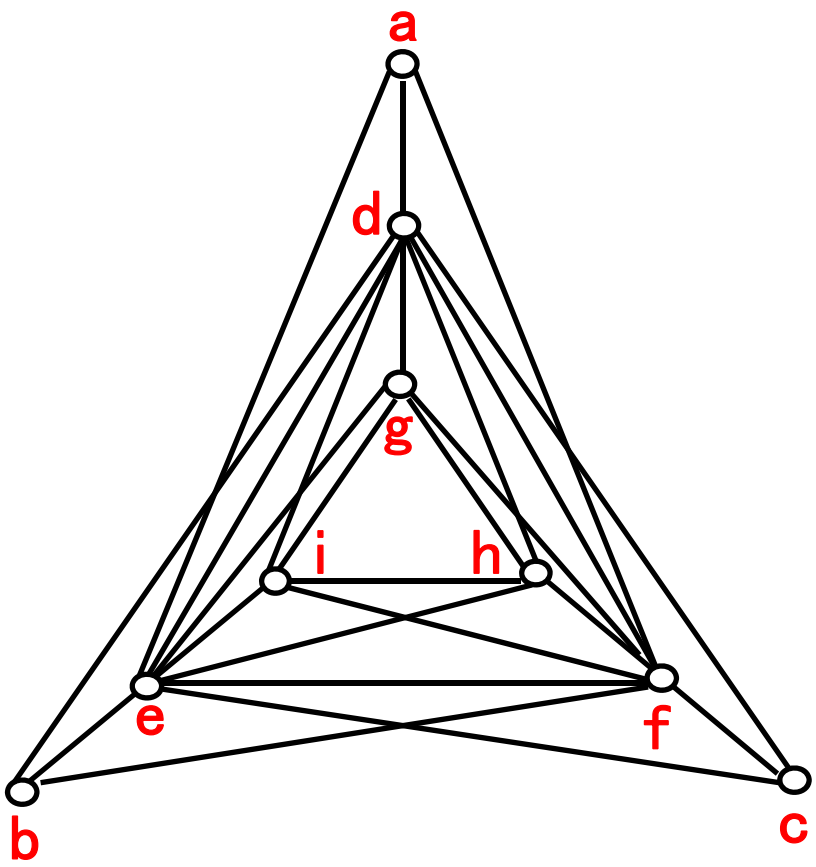
$$p(G-V_1) \leqslant |V_1| + 1$$

定理本身用处不大，但它的逆否命题却非常有用。经常利用逆否命题来判断某些图不是哈密顿图，即：若存在V的某个非空子集V_1，使得$p(G-V_1) > |V_1|$，则G不是哈密顿图。



哈密顿图的判定

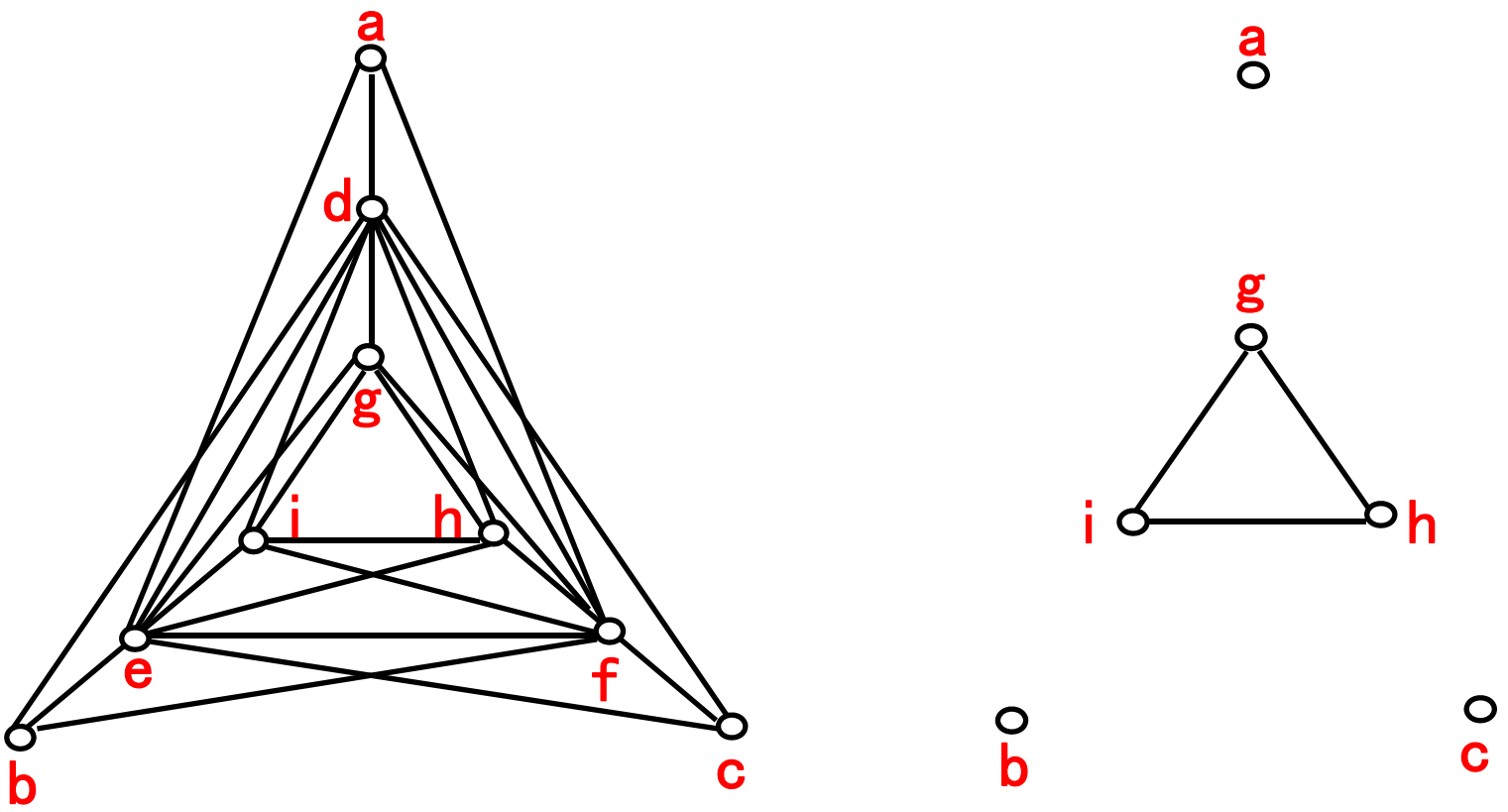
证明图中不存在哈密顿回路。



分析　利用定理的逆否命题，寻找V的某个非空子集V_1，使得$p(G-V_1) > |V_1|$，则G不是哈密顿图。

找到V_1 = {d, e, f}满足要求。

哈密顿图的判定



证明 在图中，删除结点子集{d，e，f}，得新图，它的连通分支为4个，由定理知，该图不是哈密顿图，因而不会存在哈密顿回路。

哈密顿图的判定

定理 设G = <V, E>是具有n（n ≥ 3）个结点的简单图。如果G中任意两个不相邻结点u, v的度数，均有

$$\deg(u)+\deg(v)\geqslant n-1$$

则G中存在哈密顿道路。

需要注意，定理给出的是哈密顿图的充分条件，而不是必要条件。在六边形中，任两个不相邻的结点的度数之和都是4＜6，但六边形是哈密顿图。

哈密顿图的判定

推论 设G = <V, E>是具有n（n ≥ 3）个结点的简单图。如果对任意两个不相邻结点u, v∈V，均有

$$\deg(u)+\deg(v)\geqslant n$$

则G中存在哈密顿回路。

推论 设G = <V, E>是具有n个结点的简单图，n≥3。如果对任意v∈V，均有deg(v)≥ n/2，则G是哈密顿图。

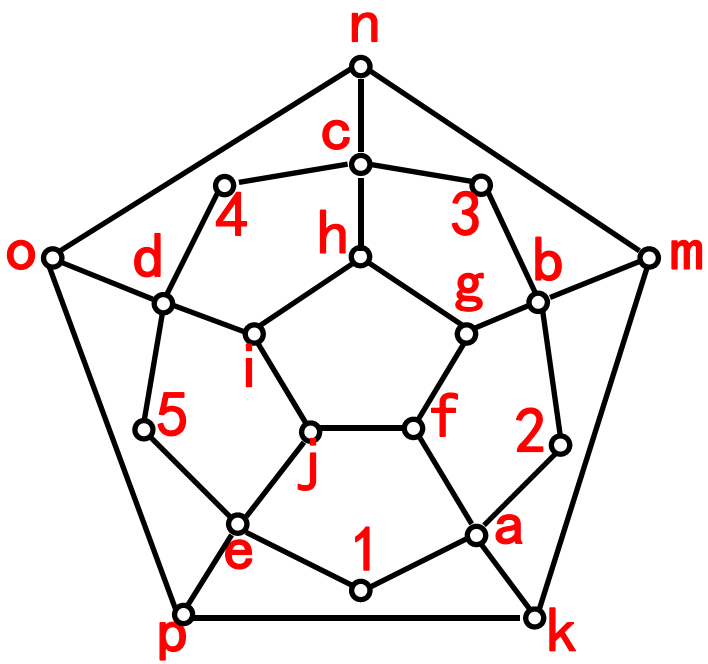
哈密顿图的应用

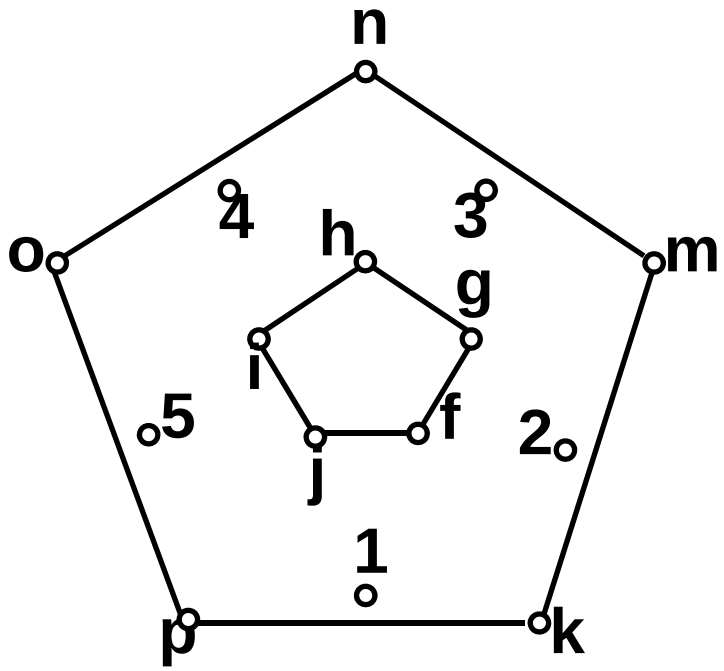
某地有5个风景点，若每个风景点均有2条道路与其他点相通。问游人可否经过每个风景点恰好一次而游完这5处？

解 将5个风景点看成是有5个结点的无向图，两风景点间的道路看成是无向图的边，因为每处均有两条道路与其他结点相通，故每个结点的度数均为2，从而任意两个不相邻的结点的度数之和等于4，正好为总结点数减1。故此图中存在一条哈密顿道路，因此游人可以经过每个风景点恰好一次而游完这5处。

哈密顿图的应用

判断下图是否存在哈密顿回路。





解　删除结点子集{a，b，c，d，e}，得到的图有7个连通分支，由定理知，该图不是哈密顿图，因而不会存在哈密顿回路。

树的概念

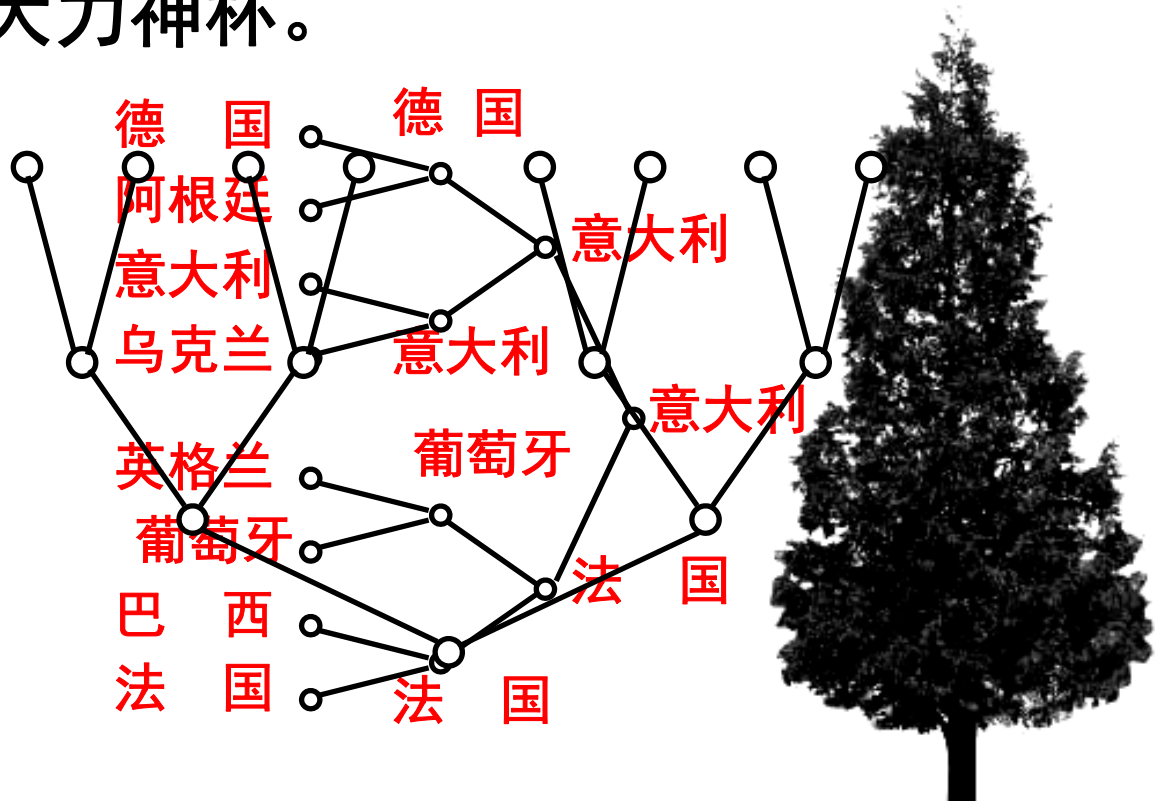
树是图论中的一个非常重要的概念，而在计算机科学中有着非常广泛的应用，例如现代计算机操作系统均采用树形结构来组织文件和文件夹，本章介绍树的基本知识和应用。

在本章中，所谈到的图都假定是简单图；所谈到的回路均指简单回路或基本回路。

树的概念

树的定义与性质

例 2006年德国世界杯8强的比赛结果图，最后胜利的队捧得大力神杯。



树的概念

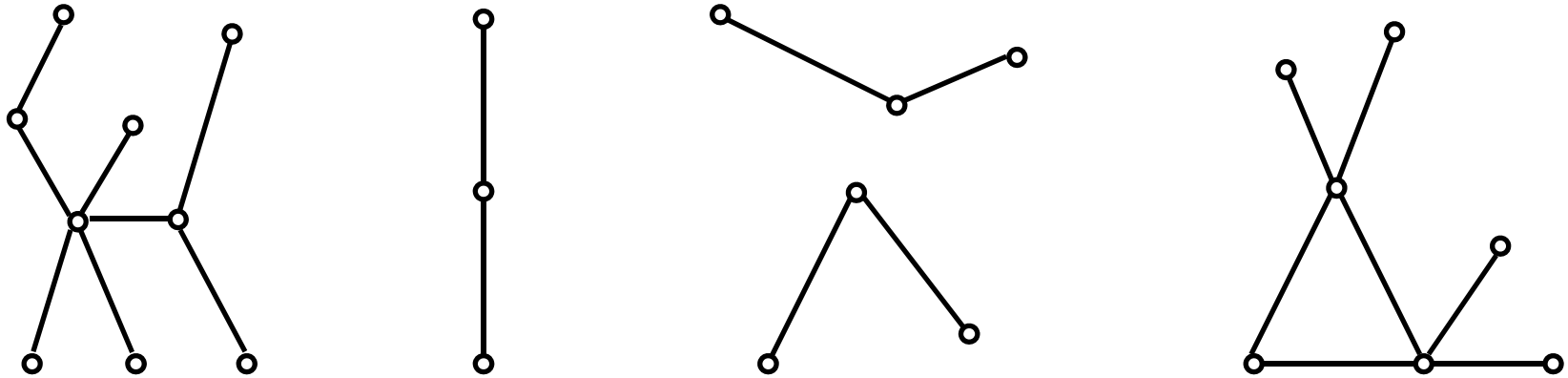
- 连通而不含回路的无向图称为无向树(Undirected Tree)，简称树(Tree)，常用T表示树。
- 树中度数为1的结点称为树叶(Leaf)；度数大于1的结点称为分支点(Branch Point) 。
- 每个连通分支都是树的无向图称为森林(Forest)。
- 平凡图称为平凡树(Trivial Tree)。

➢ 树中没有环和平行边，因此一定是简单图

➢ 在任何非平凡树中，都无度数为0的结点。

树的概念

判断下图中的图哪些是树？为什么？



(a)　(b)　(c)　(d)

分析 判断无向图是否是树，根据定义，首先看它是否连通，然后看它是否有回路。

解 图(a)、(b)都是连通，并且不含回路，因此是树；图(c)不连通，因此不是树，但由于它不含回路，因此是森林；图(d)虽然连通，但存在回路，因此不是树。

割边的概念

- **定义：若$G' = G - e$比G的连通支数多, 则称e是G的割边.**
  - 删去$e = (u,v)$, 则u,v分属于不同的连通支.
- **定理：$e = (u,v)$是割边 *iff* e不属于任何回路.**

  $\Rightarrow$:若$e = (u,v)$属于某回路,则$G' = G - e$中仍有从u到v的道路,故u,v属于同一连通支,与e是割边矛盾.

  $\Leftarrow$:若$e = (u,v)$不是割边,则$G' = G - e$中与的连通支数一样,故u,v属于同一连通支,有从u到v的道路,连同e即构成回路,矛盾.
- **显然，树中的边都是割边.**

树的性质

定理 设T是结点数为n≥2的树，则下列性质是等价的：

① T连通且不含回路；

② T连通且每条边都是割边；

③ T是连通的，且有 n-1条边；

④ T中无回路，但在T中任二结点之间增加一条新边，就得到惟一的一个回路；

⑤ T是连通的，但删除G中任一条边后，便不连通；

⑥ T中每一对结点之间有惟一一条道路。

树的特点

在结点给定的无向图中，

树是边数最多的无回路图

树是边数最少的连通图

由此可知，在无向图G = (n, m)中，

若m＜n-1，则G是不连通的

若m＞n-1，则G必含回路

树的特点

任意非平凡树T = (n, m) 都至少有两片叶。

证明 因树T是连通的，从而T中各结点的度数均大于等于1。设T中有k个度数为1的结点(即k片叶)，其余的结点度数均大于等于2。

$$2m = \sum_{v \in V} \deg(v) \geq k + 2(n - k) = 2n - k$$

由于树中有m = n-1，于是2(n-1)≥2n-k，因此可得k≥2，这说明T中至少有两片叶。

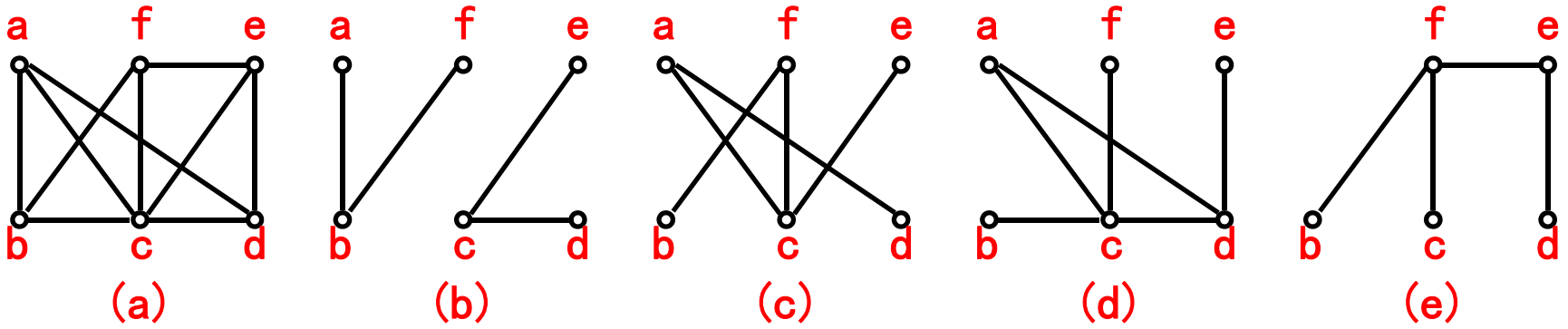
生成树

定义　给定图G = <V, E>，若G的某个**生成子图**是**树**，则称之为G的**生成树**(Spanning Tree)，记为T_G。生成树T_G中的边称为**树枝**(Branch)；G中不在T_G中的边称为**弦**(Chord)；T_G的所有弦的集合称为生成树的**补**(Complement)。

定理　一个图G = <V, E>存在生成树T_G = <V_T, E_T>的充分必要条件是G是连通的。

生成树

判断下图中的图(b)、(c)、(d)、(e)是否是图(a)的生成树。



分析 判断是否是生成树，根据定义，首先看它是否是树，然后再看它是否是生成子图。由于图(b)和(d)不是树，图(e)不是生成子图，因此它们都不是图(a)的生成树，而图(c)既是树，又是生成子图，因此是生成树。

最小生成树

定义　设G = <V, E>是**连通的赋权图**，T是G的一棵生成树，T的每个**树枝所赋权值之和**称为T的**权**(Weight)，记为w(T)。G中具有**最小权的生成树**称为G的**最小生成树**(Minimal Spanning Tree)。

一个无向图的生成树不是惟一的，同样地，一个赋权图的最小生成树也不一定是惟一的。

Kruskal算法

（1）在G中选取最小权边e_1，置i = 1。

（2）当i = n-1时，结束，否则转(3)。

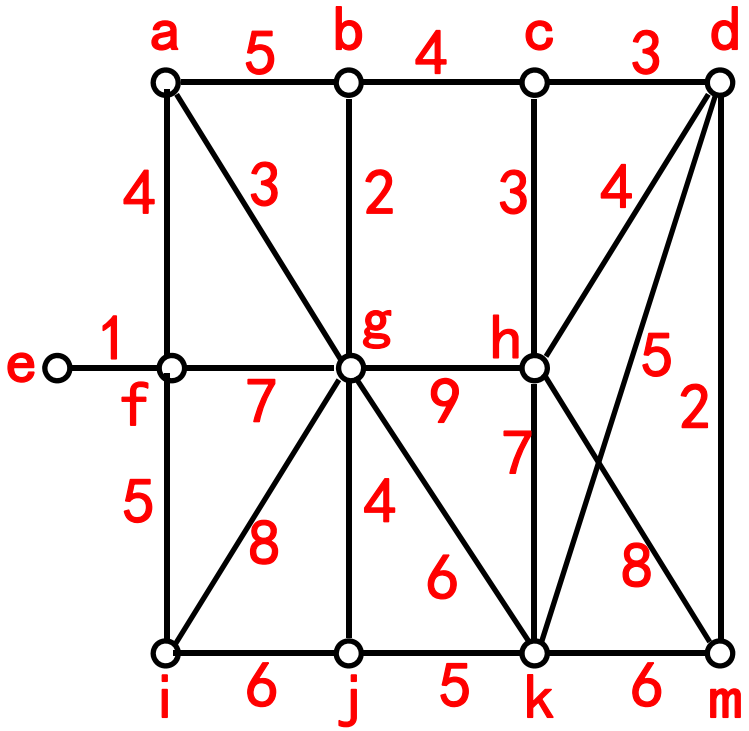
（3）设已选取的边为e_1，e_2，…，e_i，在G中选取不同于e_1，e_2，…，e_i的边e_{i+1}，使{e_1，e_2，…，e_i，e_{i+1}}中无回路且e_{i+1}是满足此条件的最小权边。

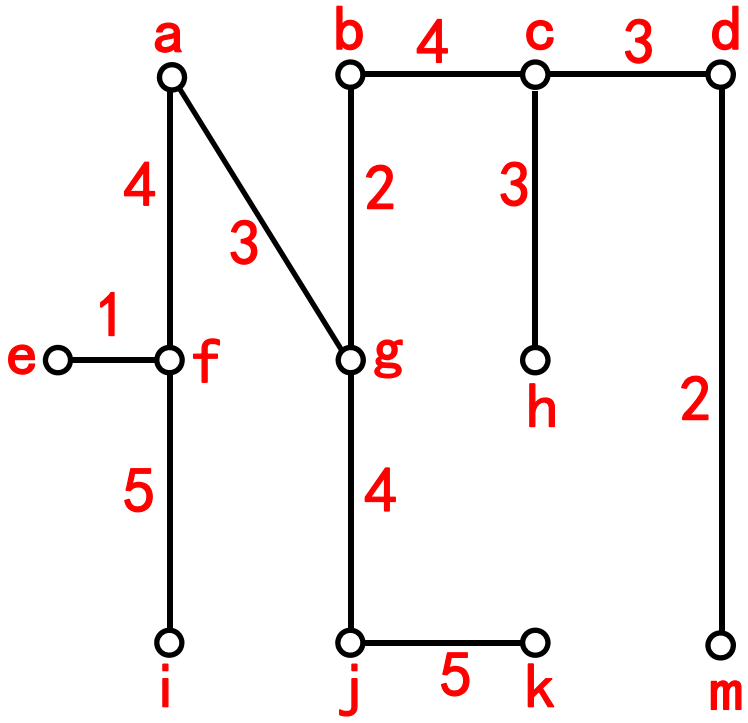
（4）置i = i+1，转(2)。

在Kruskal算法的步骤1和3中，若满足条件的最小权边不止一条，则可从中任选一条，这样就会产生不同的最小生成树。

Kruskal算法

用Kruskal算法求图中赋权图的最小生成树。





解 n=12，按算法要执行n-1=11次，w(T)= 36。

Prim算法

（1）在G中任意选取一个结点v_1，置$V_T = \{v_1\}$，$E_T = \Phi$，$k = 1$；

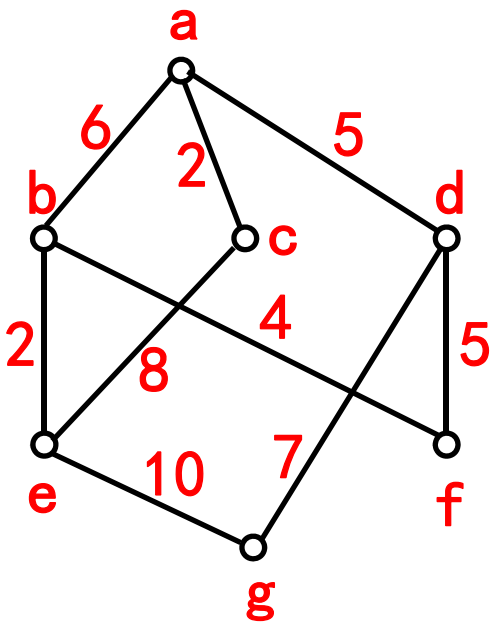
（2）在$V-V_T$中选取与某个$v_i \in V_T$邻接的结点v_j，使得边(v_i, v_j)的权最小，置$V_T = V_T \cup \{v_j\}$，$E_T = E_T \cup \{(v_i, v_j)\}$，$k = k+1$；

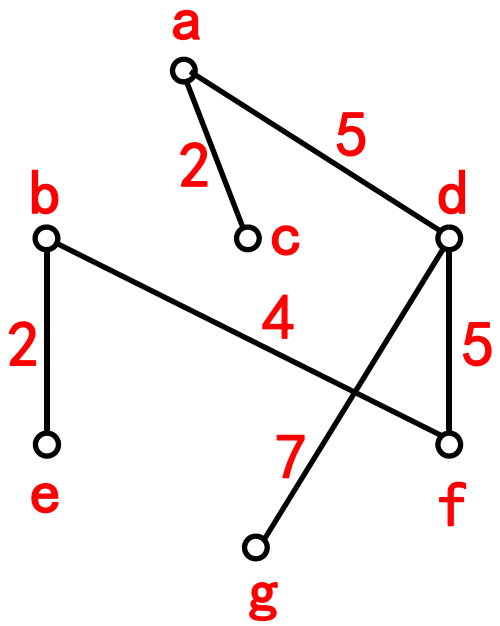
（3）重复步骤2，直到$k = |V|$。

在Prim算法的步骤2中，若满足条件的最小权边不止一条，则可从中任选一条，这样就会产生不同的最小生成树。

Prim算法

用Prim算法求图中赋权图的最小生成树。





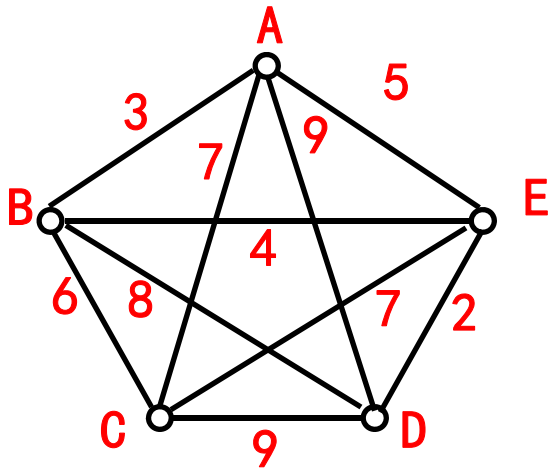
解 n = 7，按算法要执行n-1 = 6次，

w(T)= 25。

由Prim算法可以看出，每一步得到的图一定是树，故不需要验证是否有回路，因此它的计算工作量较Kruskal算法要小。

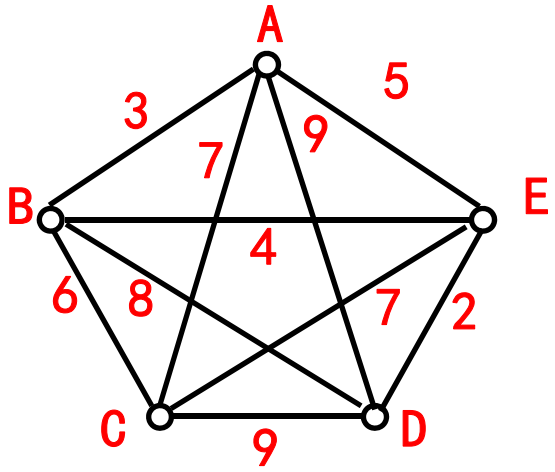
无向树的应用

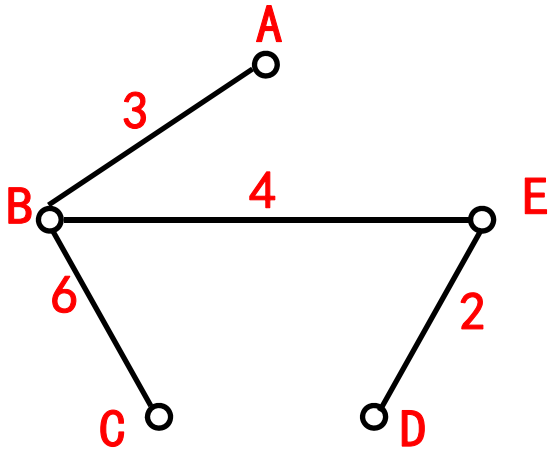
例 假设有5个信息中心A、B、C、D、E，它们之间的距离(以百公里为单位)如图所示。要交换数据，我们可以在任意两个信息中心之间通过光纤连接，但是费用的限制要求铺设尽可能少的光纤线路。重要的是每个信息中心能和其它中心通信，但并不需要在任意两个中心之间都铺设线路，可以通过其它中心转发。



分析 这实际上就是求赋权连通图的最小生成树问题，可用Prim算法或Kruskal算法求解。

无向树的应用





解　求得图的最小生成树如图所示，w(T) = 15百公里。即按图的图铺设，使得铺设的线路最短。

Huffman树

在计算机及通讯事业中，常用二进制编码来表示符号，称之为码字(codeword)。例如，可用00、01、10、11分别表示字母A、B、C、D。如果字母A、B、C、D出现的频率是一样的，传输100个字母用200个二进制位。但实际上字母出现的频率很不一样，如A出现的频率为50%，B出现的频率为25%，C出现的频率为20%，D出现的频率为5%。能否用不等长的二进制序列表示字母A、B、C、D？事实上，可用000表示字母D，用001表示字母C，01表示B，1表示A。这样表示，传输100个字母所用的二进制位为

$$3\times5+3\times20+2\times25+1\times50 = 175。$$

这种表示比用等长的二进制序列表示法好，节省了二进制位。但当我们用1表示A，用00表示B，用001表示C，用000表示D时，如果接收到的信息为001000，则无法辨别它是CD还是BAD。因而，不能用这种二进制序列表示A、B、C、D，要寻找另外的表示法。

前缀码定义

设$a_1a_2\cdots a_{n-1}a_n$为长度为n的符号串，称其子串a_1，a_1a_2，…，$a_1a_2\cdots a_{n-1}$分别为$a_1a_2\cdots a_{n-1}a_n$的长度为1，2，…，n-1的**前缀**(Prefix)。

设A = $\{b_1, b_2, \cdots, b_m\}$是一个符号串集合，若对任意$b_i$，$b_j \in A$，$b_i \neq b_j$，$b_i$不是$b_j$的前缀，$b_j$也不是$b_i$的前缀，则称A为**前缀码**。若符号串$b_i$(i = 1，2…，m)中，只出现0和1两个符号，则称A为**二元前缀码**。

$\{1, 01, 001, 000\}$是前缀码

$\{1, 11, 001, 0011\}$不是前缀码。

前缀码生成

定义　除树叶外，其余结点的正度最多为2的树称为二叉树。如果它们的正度都是2，称为完全二叉树。

给定一棵二叉树T，假设它有t片树叶。

设v是T任意一个分支点，则v至少有一个儿子至多有两个儿子。若v有两个儿子，则在由v引出的两条边上，左边的标上0，右边的标上1；若v只有一个儿子，在v引出的边上可标0也可标1。设v_i为T的任意一片树叶，从树根到v_i的通路上各边的标号组成的符号串放在v_i处，t片树叶处的t个符号串组成的集合为一个二元前缀码。

前缀码生成

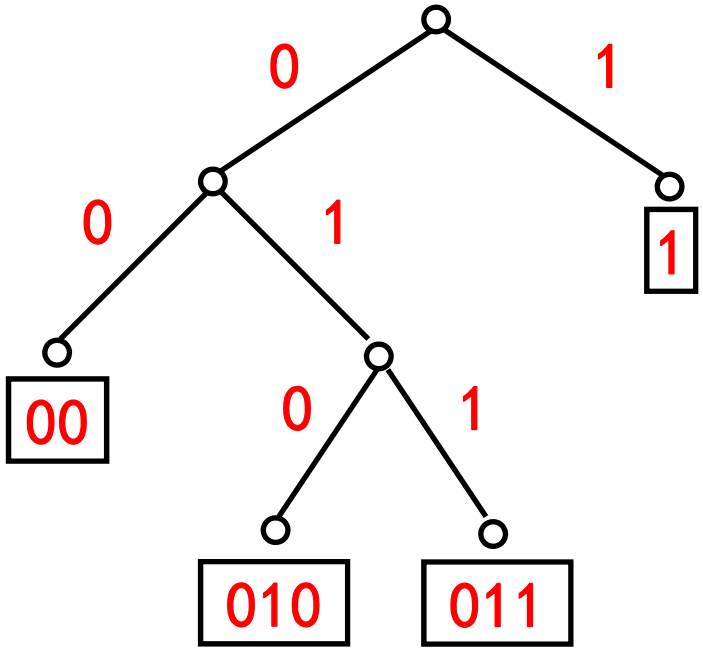
给定一棵二元树T，假设它有t片树叶。

设v是T任意一个分支点，则v至少有一个儿子至多有两个儿子。若v有两个儿子，则在由v引出的两条边上，左边的标上0，右边的标上1；若v只有一个儿子，在v引出的边上可标0也可标1。设v_i为T

v中的符号串的前缀均在v所在的通路上

若T存在带一个儿子的分支点，则由T产生的前缀码不唯一，但T若为完全二元树，则T产生的前缀码就是唯一的了。

前缀码生成



图中所示的二元树产生的前缀码为：{1，00，010，011}。

因此用1表示A，用00表示B，用010表示 C，用011表示D 即可满足要求。

当知道了传输的符号出现的频率时，如何选择前缀码，使传输的二进制位尽可能地少呢？
先产生最优二元树T，然后用T产生二元前缀码。

最优二叉树

定义 设有一棵二叉树，若对其所有的t片叶赋以权值w_1、w_2、…、w_t，则称之为**赋权二叉树**；若权为w_i的叶的**层数**为$L(w_i)$，则称 $W(T)=\sum_{i=1}^{t} w_i \times L(w_i)$ 为该赋权二叉树的权；而在所有赋权w_1、w_2、…、w_t的二叉树中，W(T)最小的二叉树称为**最优二叉树**。

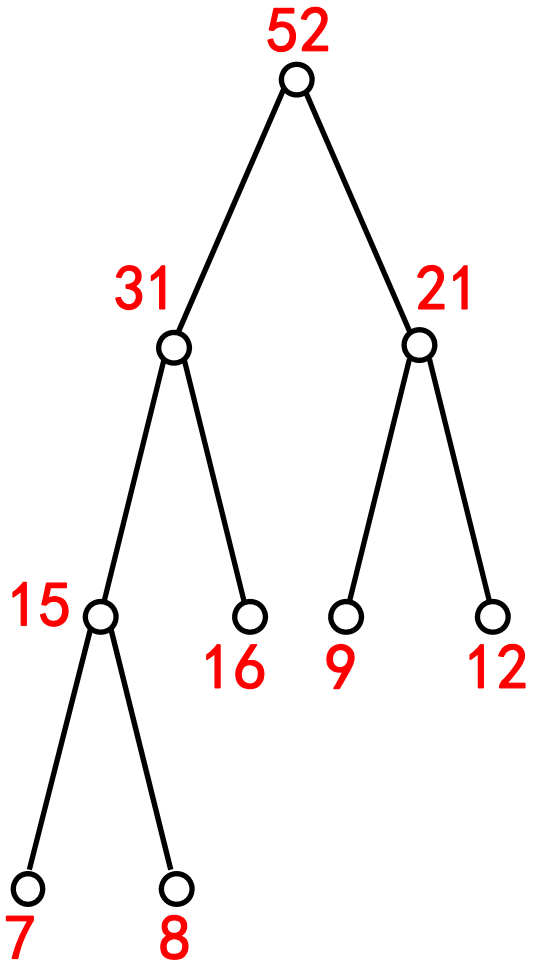
1952年，Huffman给出了求最优二叉树的方法。该方法的关键是：从带权为W_1+W_2、W_3、…、W_t（这里假设$W_1 \leqslant W_2 \leqslant \cdots \leqslant W_t$）的最优树T’中得到带权为$W_1$、$W_2$、…、$W_t$的最优树。

Huffman算法

1. 初始：令S = $\{W_1, W_2, \cdots, W_t\}$；
2. 从S中取出两个最小的权W_i和W_j，画结点v_i，带权W_i，画结点v_j，带权W_j。画v_i和v_j的父亲v，连接v_i和v，v_j和v，令v带权W_i+W_j；
3. 令S = $(S - \{W_i, W_j\}) \cup \{W_i+W_j\}$；
4. 判断S是否只含一个元素？若是，则停止，否则转2。

Huffman算法

求带权7、8、9、12、16的最优树。



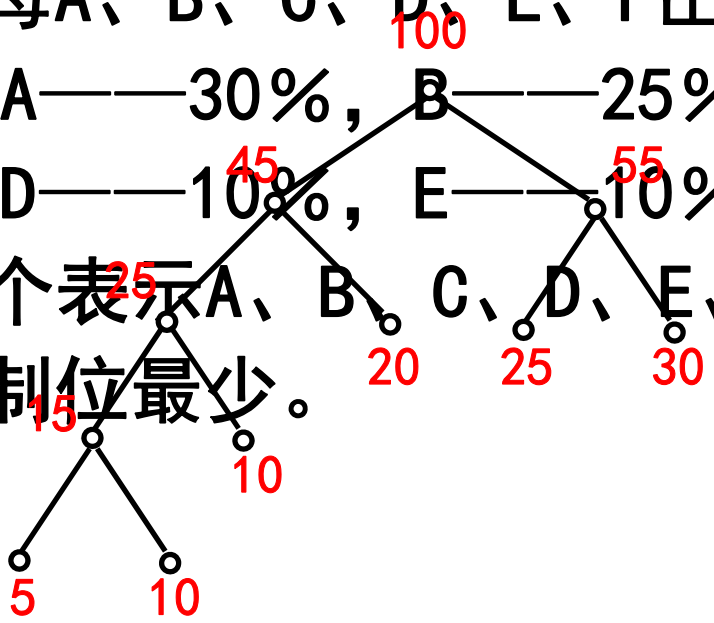
Huffman算法

已知字母A、B、C、D、E、F出现的频率如下：

A——30%，B——25%，C——20%

D——10%，E——10%，F—— 5%

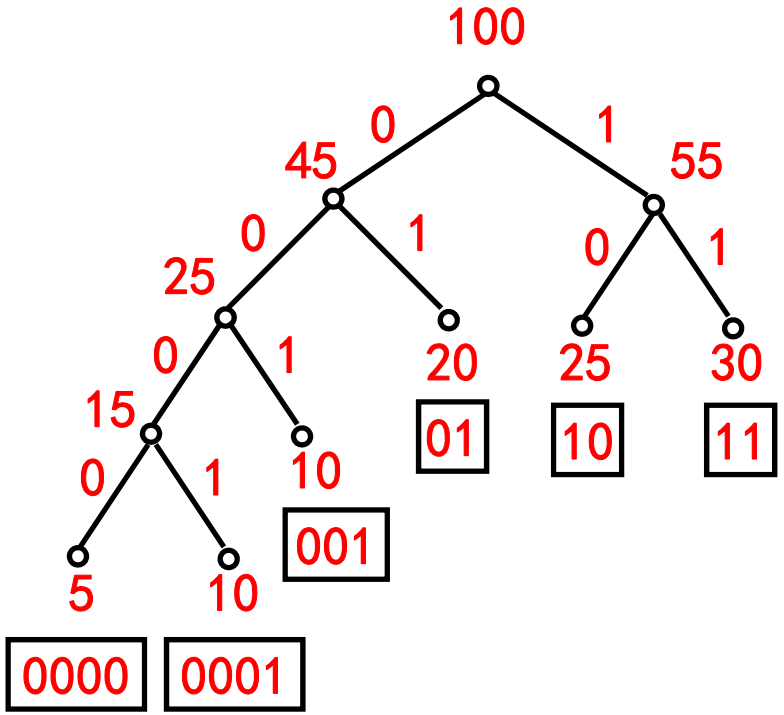
构造一个表示A、B、C、D、E、F前缀码，使得传输的二进制位最少。



解

（1）求带权30, 25, 20, 10, 10, 5的最优二元树T。

（2）在T上求一个前缀码。



Huffman算法

（3）设树叶v_i带权为w%×100 = w，则v_i处的符号串表示出现频率为w%的字母。

{01，10，11，001，0001，0000}

为一前缀码，其中

0000表示F，　　0001表示E，

001表示D，　　01表示C，

10表示B，　　11表示A。

传输100个这样的字母所用的二进制位为：

4×(5+10)+3×10+2×(20+25+30) = 240。