第7章 图

本章知识点

- 图的类型定义（集合与图论）
- 图的存储表示（集合与图论讲过矩阵的存储）
- 图的深度优先搜索遍历
- 图的广度优先搜索遍历
- 无向网的最小生成树（集合与图论）
- 最短路径（集合与图论讲过 单源最短路径）
- 拓扑排序
- 关键路径

本章难点

- ■最小生成树的算法（集合与图论）；
- ■拓扑排序的算法;
- ■关键路径算法;
- ■求最短路径的Dijkstra算法和Floyed算法。

（集合与图论）

第七章 图

7.1 图的定义和术语（集合与图论）

7.2 图的存储结构

7.3 图的遍历

7.4生成树和最小生成树

7.5 有向无环图的应用

7.6 最短路径

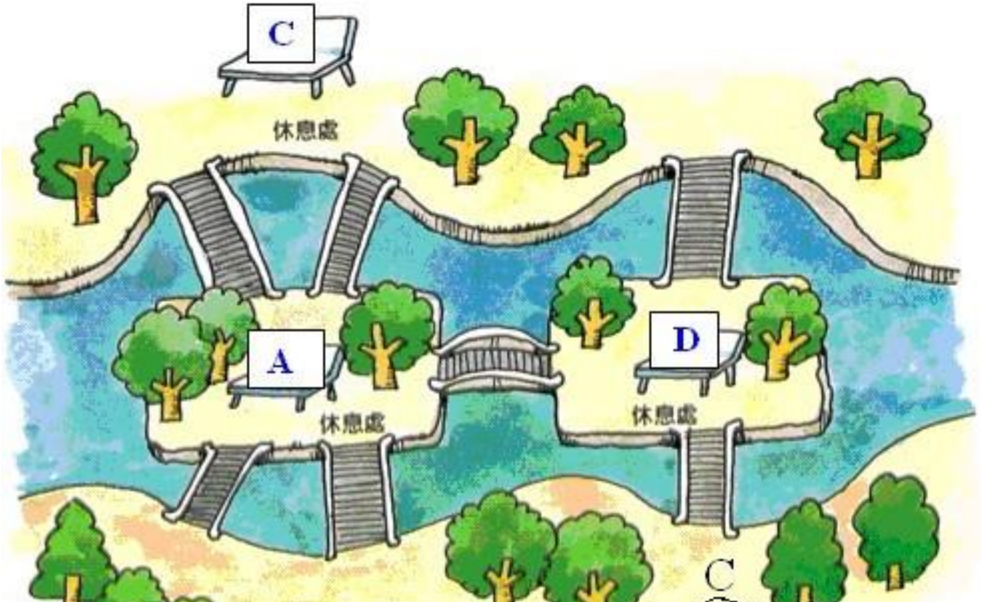
第七章 图

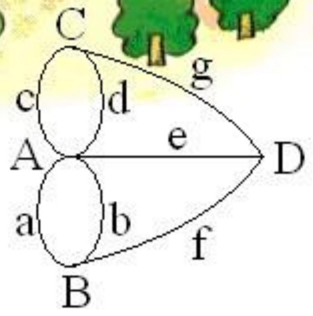
先提几个问题：

【问题1】由于大道路网的维护成本高，需选择停止维护一些道路，但要保证所有村庄之间都有路到达，即使路线并不如以前短，但要使得总的维护费用最少。

第十章 图

【问题2】





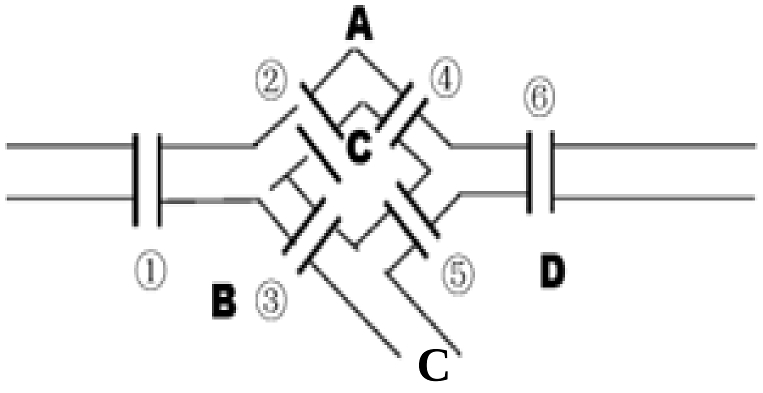
哥尼斯堡是东普鲁士的首都，今俄罗斯加里宁格勒市，普莱格尔河横贯其中。
十八世纪在这条河上建有七座桥，将河中间的两个岛和河岸联结起来。人们闲暇时经常在这上边散步，有人提出：能不能每座桥都只走一遍，最后又回到原来的位置？

——哥尼斯堡城七桥问题

1736年，大数学家欧拉首先把这个问题简化，他把两座小岛和河的两岸分别看作四个点，而把七座桥看作这四个点之间的连线，A、B、C，D表示陆地，形成了著名的——欧拉图。

第七章 图

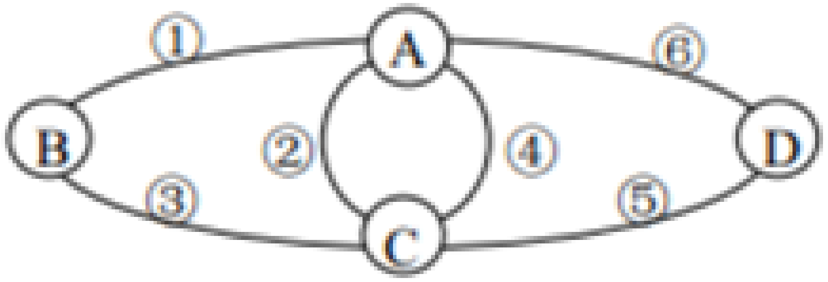
【问题3】简化的哥尼斯堡城问题



设在4地（A，B，C，D）之间架设有6座桥，要求从某一地出发，经过每座桥恰巧一次，最后仍回到原地。

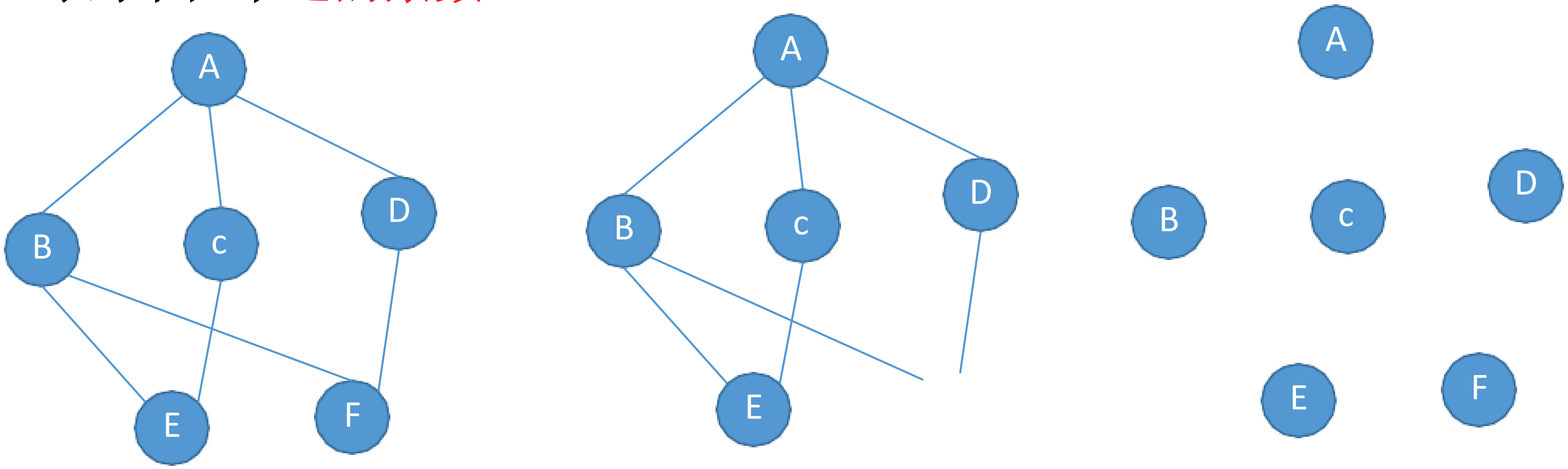
（1）此问题有解的条件是什么？

（2）描述与求解此问题有关的数据结构并编写一个算法，找出满足要求的一条回路。

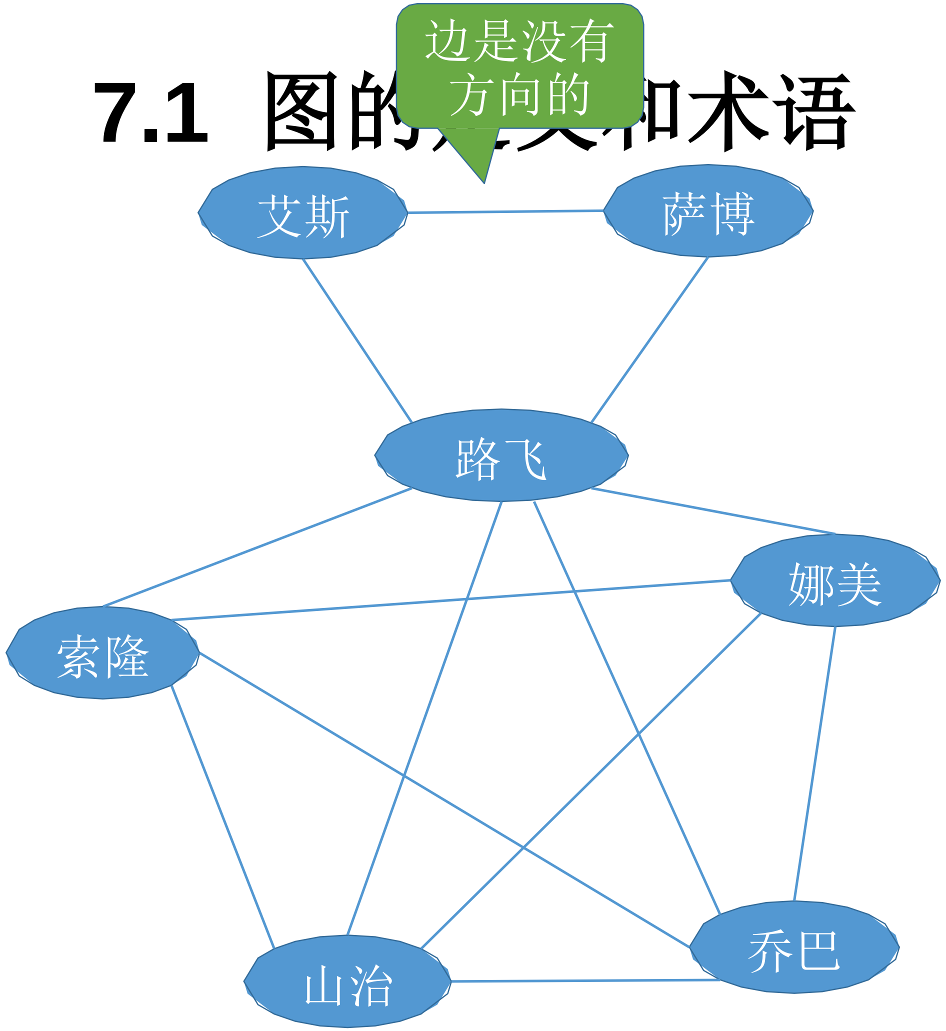


7.1 图的定义和术语

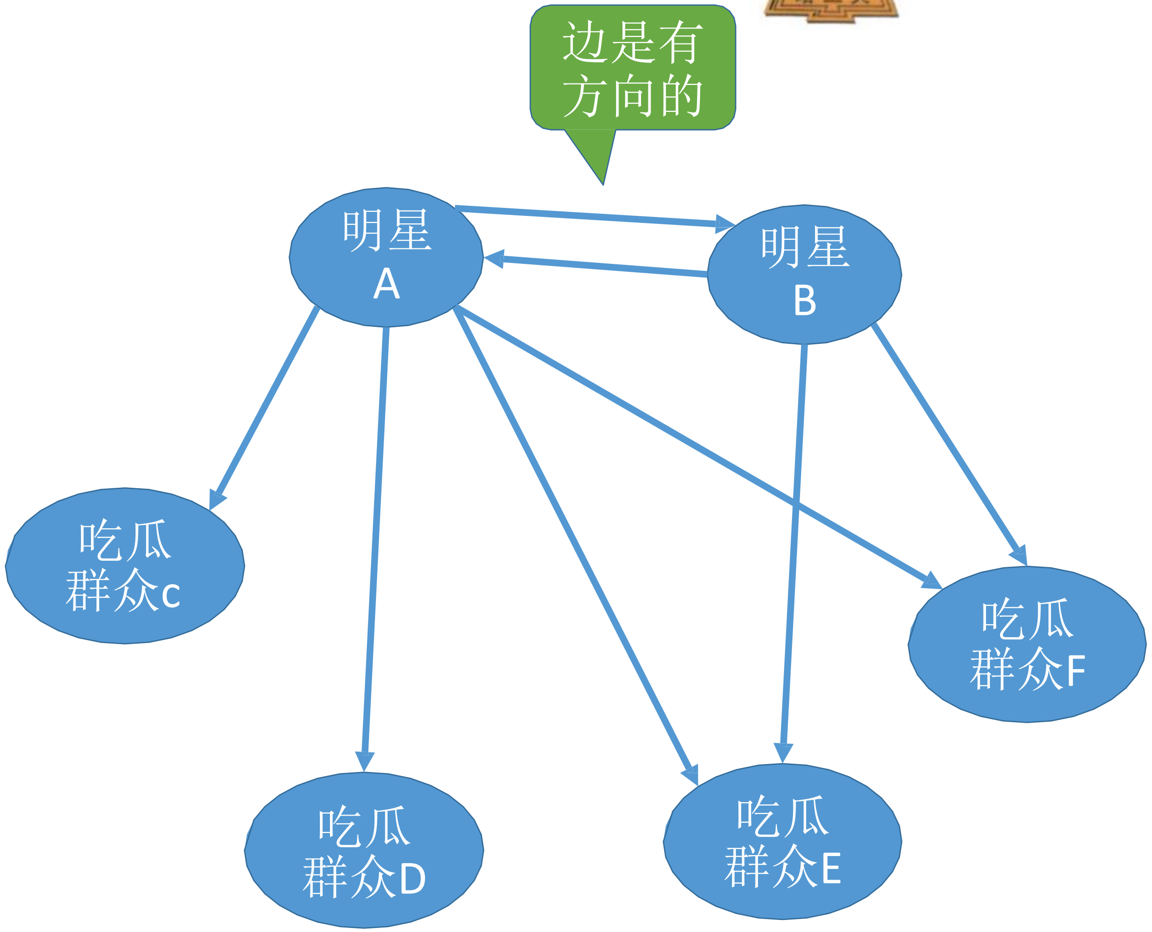
【定义】图G由顶点集V和边集E组成，记为$G = (V, E)$，其中$V(G)$表示图G中顶点的有限非空集；$E(G)$表示图G中顶点之间的关系（边）集合。若$V = \{v_l, v_2, \ldots, v_n\}$，则用 $|V|$ 表示图G中顶点的个数，也称图G的阶，$E = \{(u, v) \ u \in V, v \in V\}$，用$|E|$表示图$G$中边的条数。



7.1 图的定义和术语



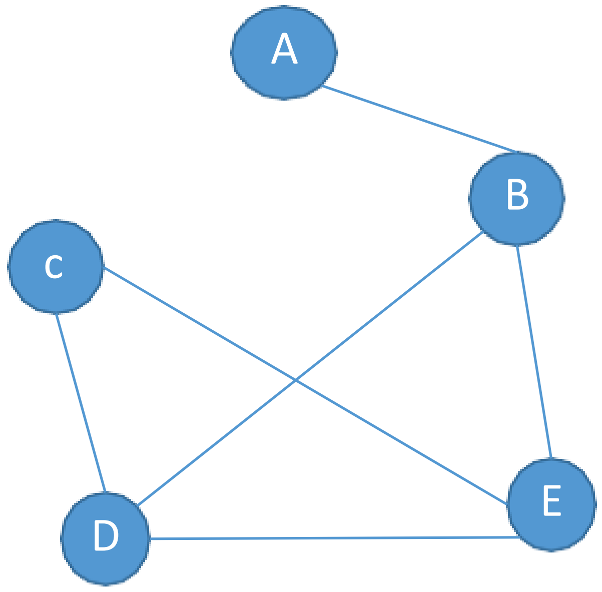
微信好友关系



微博粉丝关系

7.1 图的定义和术语

- 无向图

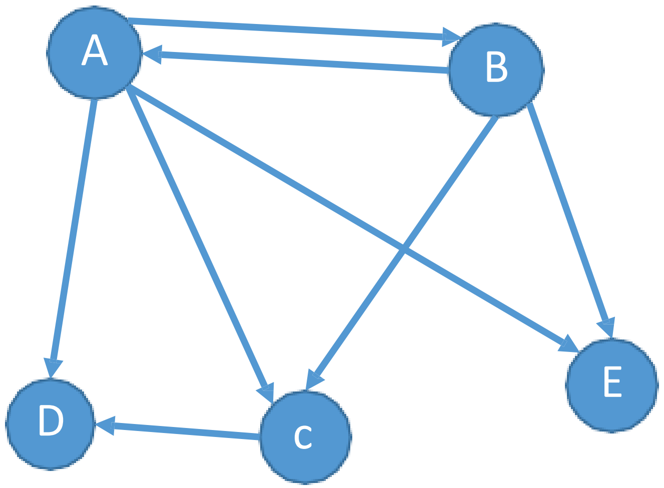


若E是无向边（简称边）的有限集合时，则图G为无向图。边是顶点的无序对，记为(v, w)或(w, v)，因为$(v, w) = (w, v)$，其中v、w是顶点。

可以说顶点w和顶点v互为邻接点。边(v, w) 依附于顶点w和v，或者说边(v, w)和顶点v、w相关联。

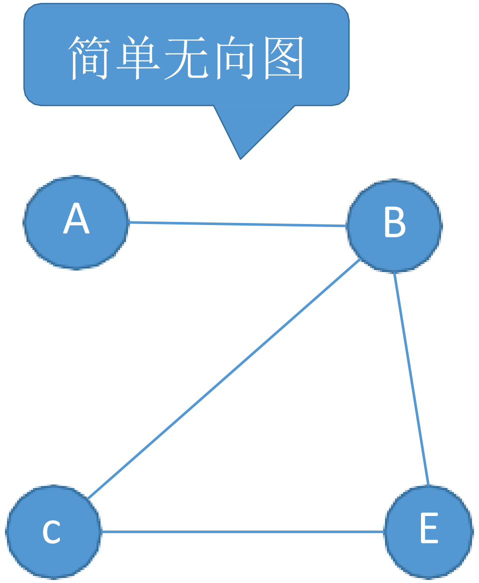
7.1 图的定义和术语

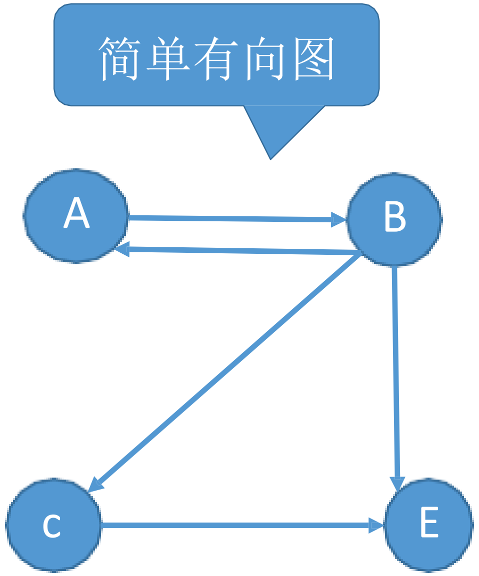
- **有向图**

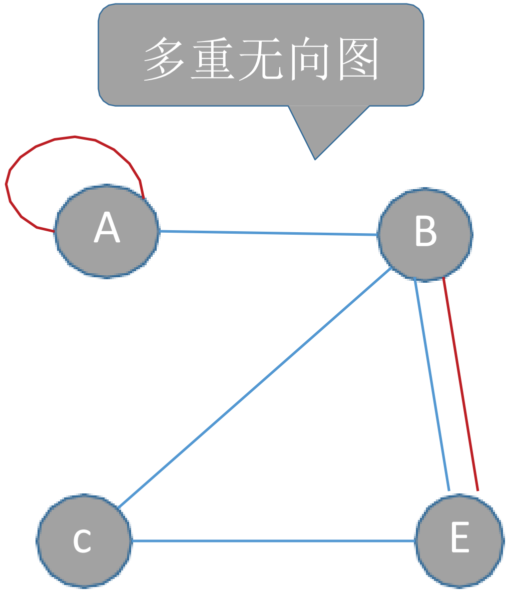


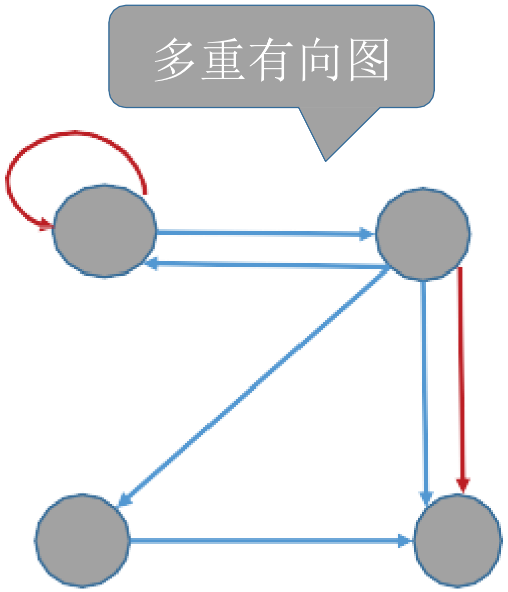
若E是有向边（也称弧）的有限集合时，则图G为有向图。 弧是顶点的有序对，记为$<v, w>$，其中v、w是顶点，v称为弧尾，w称为弧头，$<v, w>$称为从顶点v到顶点w的弧，也称v邻接到w，或w邻接自v。$<v, w> \neq <w, v>$

7.1 图的定义和术语







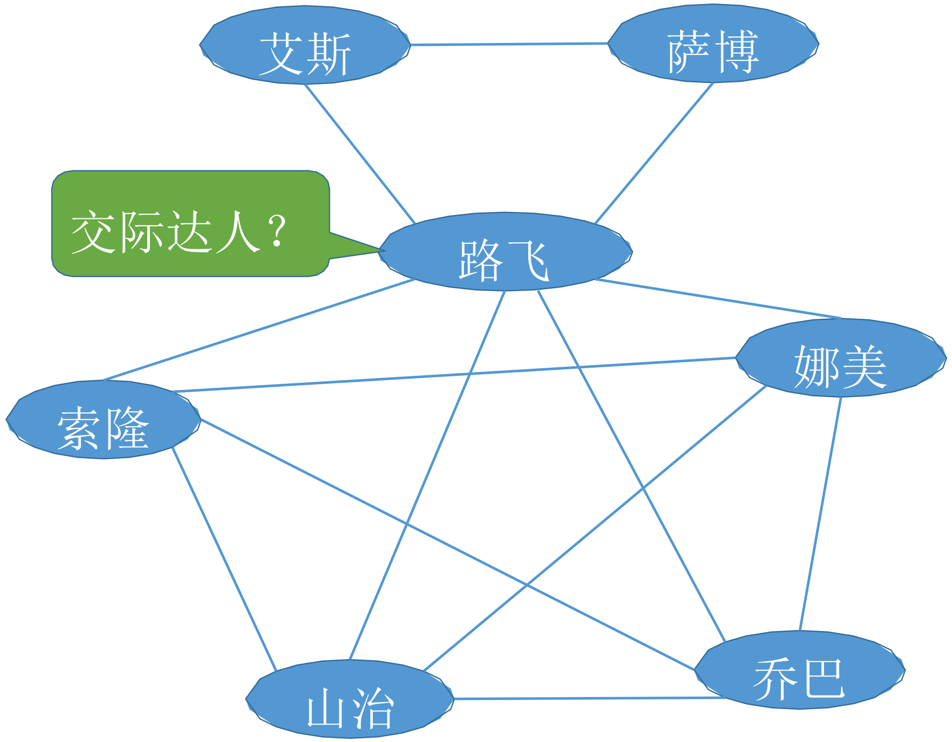


简单图——①不存在重复边；②不存在顶点到自身的边

数据结构课程只探讨“简单图”

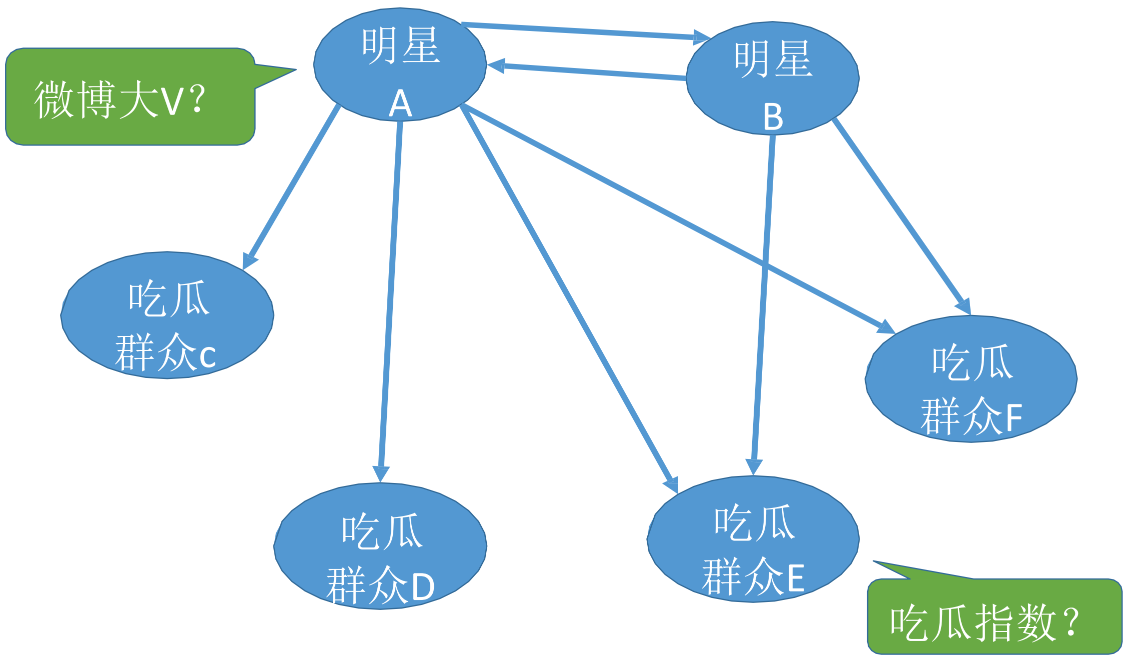
多重图——图*G*中某两个结点之间的边数多千一条，又允许顶点通过同一条边和自己关联，则*G*为多重图

7.1 图的定义和术语



微信好友关系

和顶点v关联的边的数目定义为v的**度**。



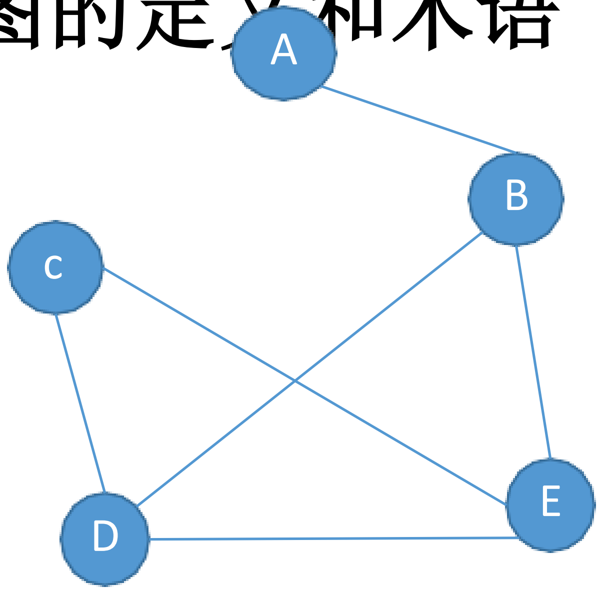
微博粉丝关系

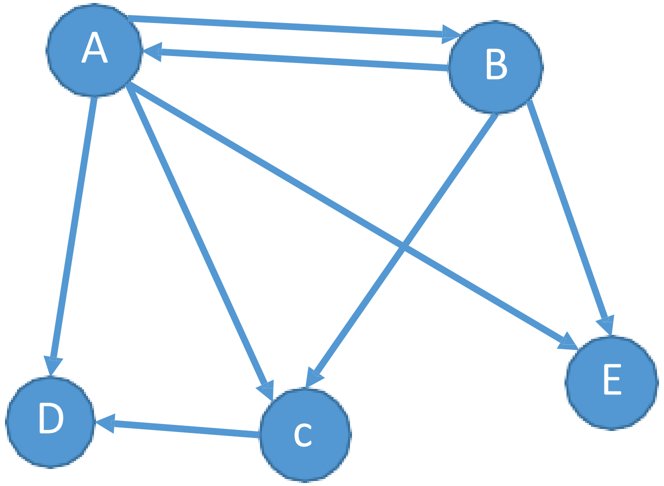
以顶点v为弧尾的弧的数目定义为顶点的**出度(OD)**；

以顶点v为弧头的弧的数目定义为顶点的**入度(ID)**。

顶点v的**度(TD) = 出度(OD) + 入度(ID)**

7.1 图的定义和术语





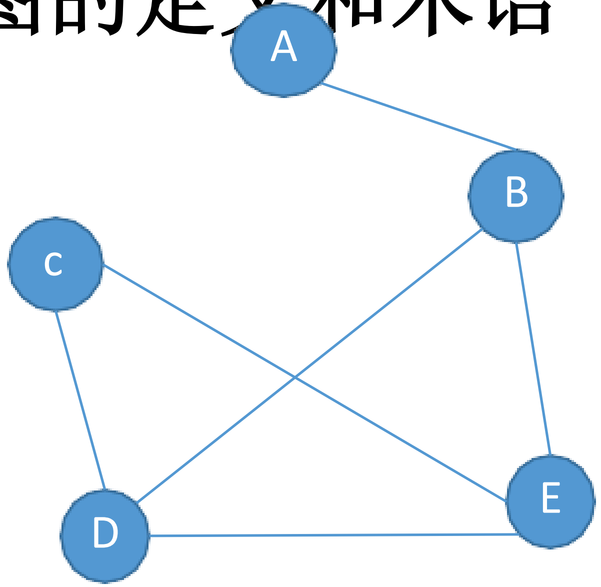
·路径——顶点v_p到顶点v_q之间的一条路径是指顶点序列，$v_p, v_{i_1}, v_{i_2}, \cdots, v_{i_m}, v_q$

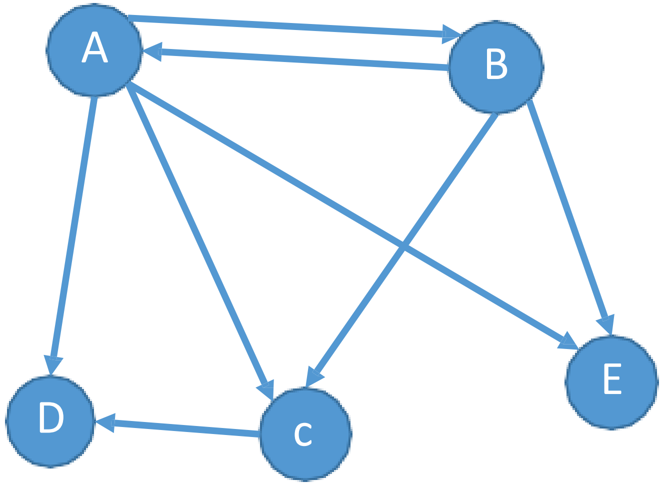
·回路——第一个顶点和最后一个顶点相同的路径称为回路或环

·简单路径——在路径序列中，顶点不重复出现的路径称为简单路径。

·简单回路——除第一个顶点和最后一个顶点外，其余顶点不重复出现的回路称为简单回路。

7.1 图的定义和术语



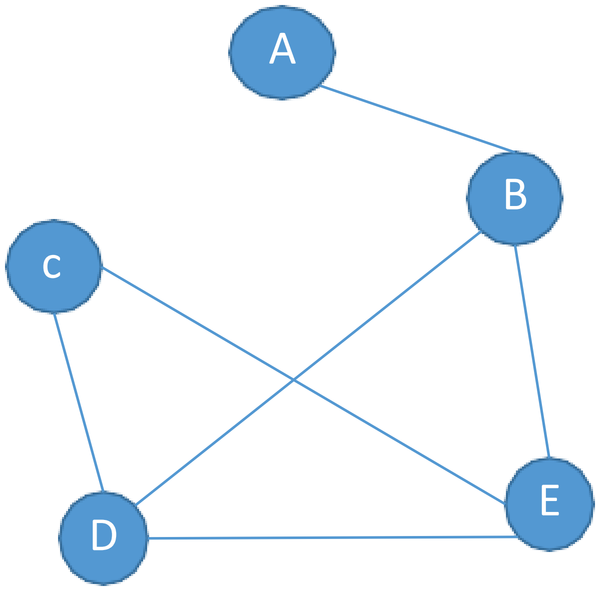


·路径长度——路径上边的数目

·点到点的距离——从顶点*u*出发到顶点*v*的**最短路径若存在**，则此路径的长度称为从*u*到*v*的距离。 若从*u*到*v*根本不存在路径，则记该距离为无穷。

7.1 图的定义和术语

- 无向图



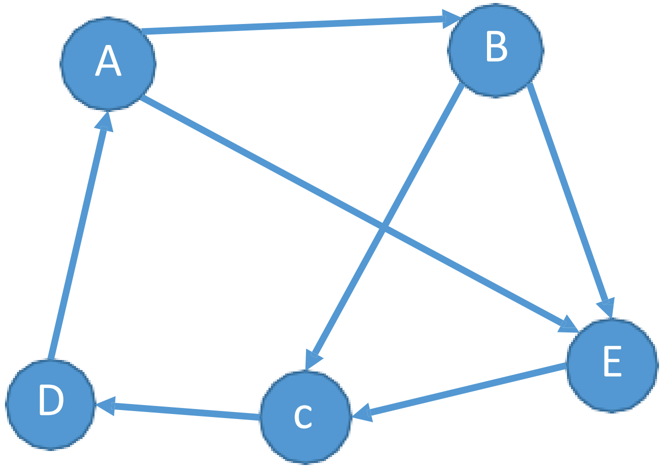
无向图中，若从顶点*v*到顶点*w*有路径存在，则称*v*和*w*是**连通**的

若图*G*中任意两个顶点都是连通的，则称图*G*为连通图，否则称为非连通图。

对于n个顶点的无向图G，
若G是连通图，则最少有？条边 $n-1$
若G是非连通图，则最多可能有？条边
C^2_{n-1}

7.1 图的定义和术语

- **有向图**



有向图中，若从顶点*v*到顶点*w*和从顶点*w*到顶点*v*之间都有路径，则称这两个顶点是**强连通**的

若图中任何一对顶点都是强连通的，则称此图为强连通图。

对于n个顶点的有向图G，若G是强连通图，则最少有？条边（形成回路）

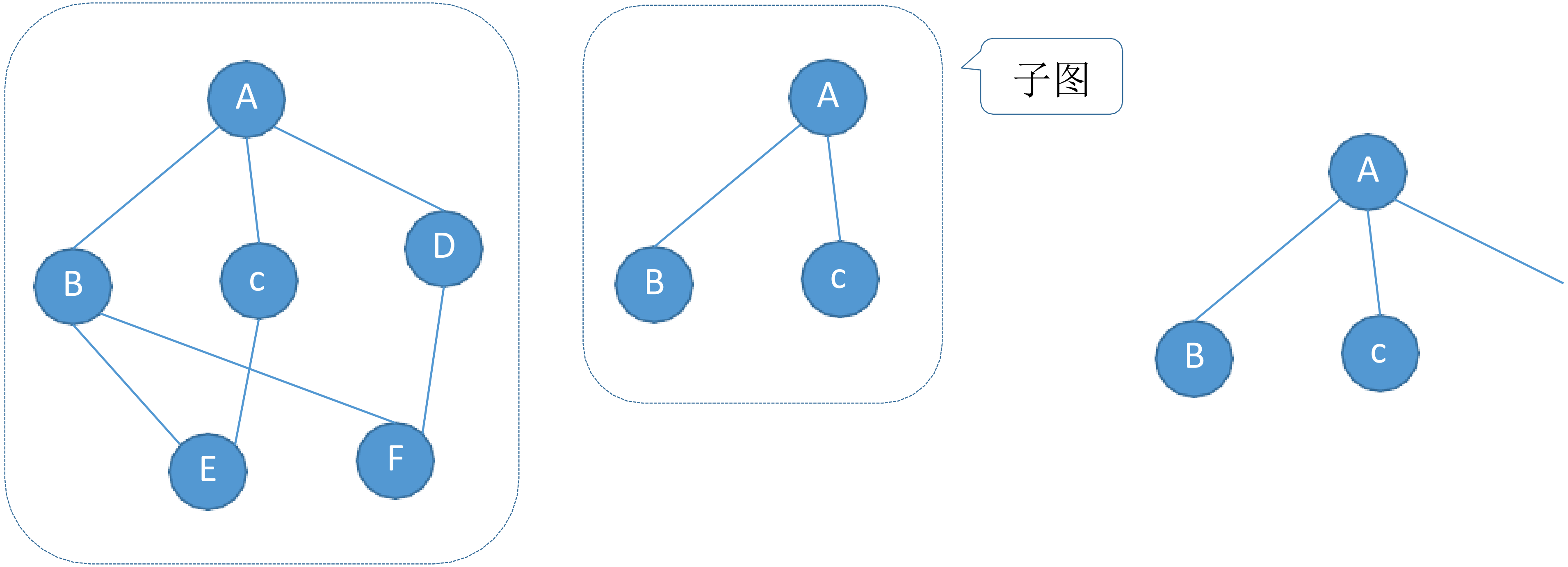
n

7.1 图的定义和术语

- 图
  - 无向图-边、**度**、**连通图**
  - 有向图-弧、弧头、弧尾、**出度**、**入度**、强连通图
  - 路径-简单路径、简单回路、**路径长度**、**点到点的距离**

7.1 图的定义和术语

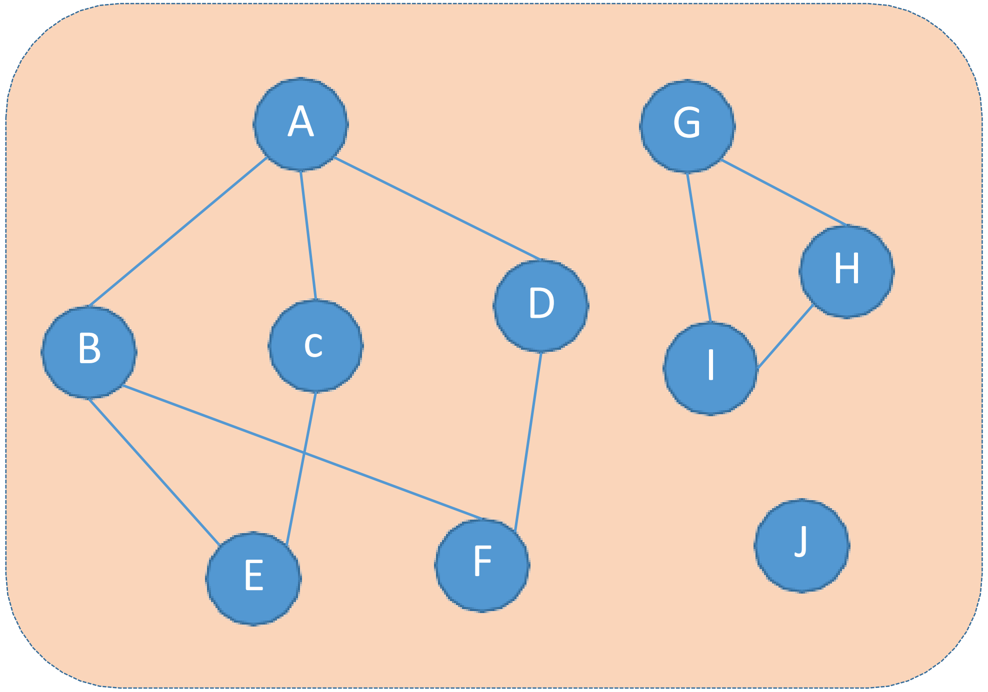
设有两个图$G = \{V, f)$和$G' = \{V', f')$，若V'是V的子集，且f'是f的子集，则称G'是G的**子图**。



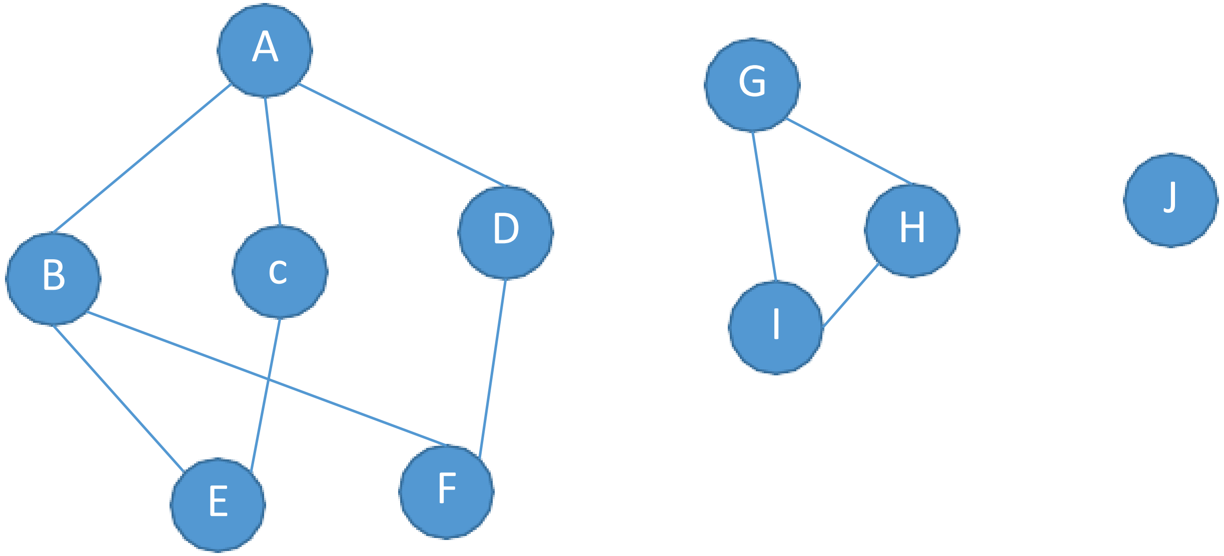
7.1 术语

子图必须连通，且包含尽可能多的顶点和边

无向图中的极大连通子图称为**连通分量**。



无向图G



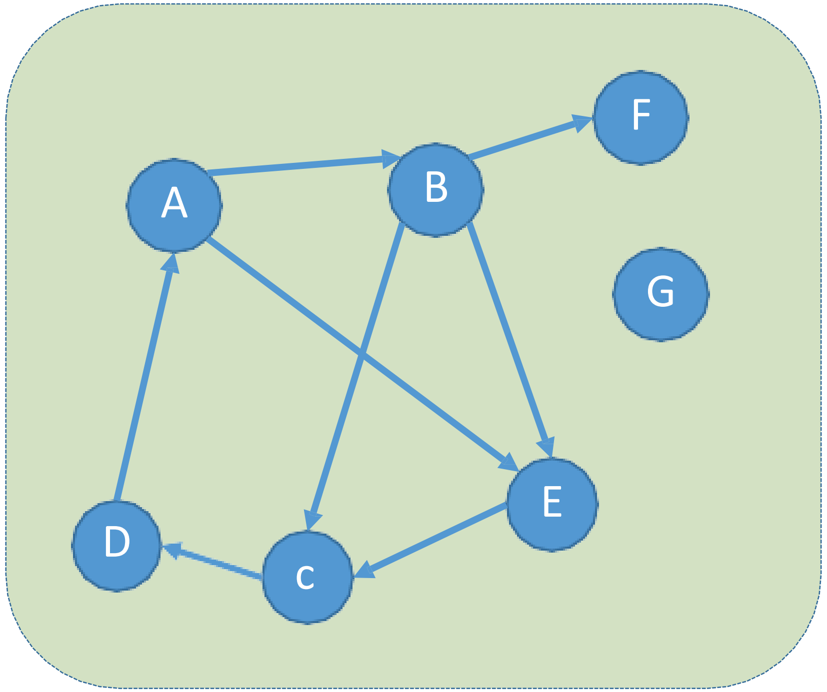
G的三个连通分量

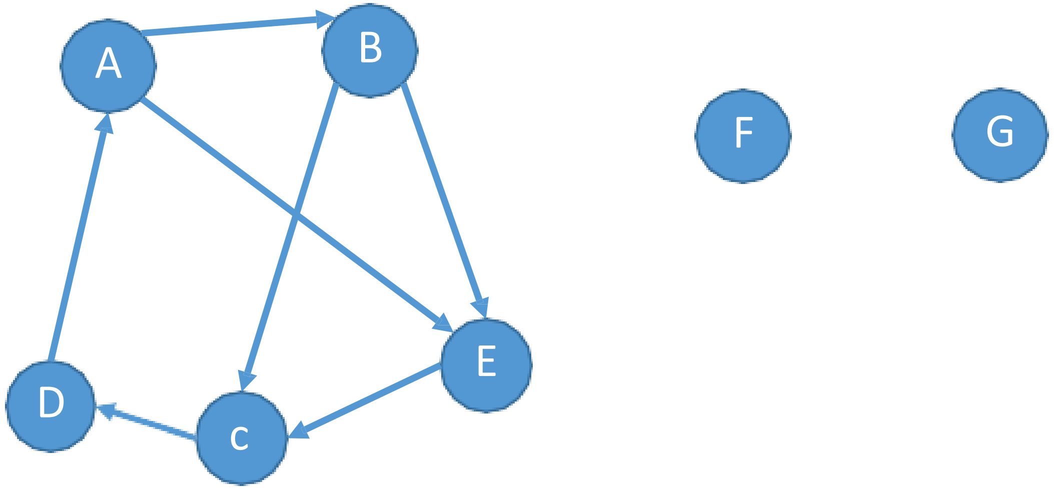
子图必须强连通，且包含尽可能多的顶点和边

7.1 语

有向图中的极大强连通子图称为有向图的**强连通分量**

有向图G

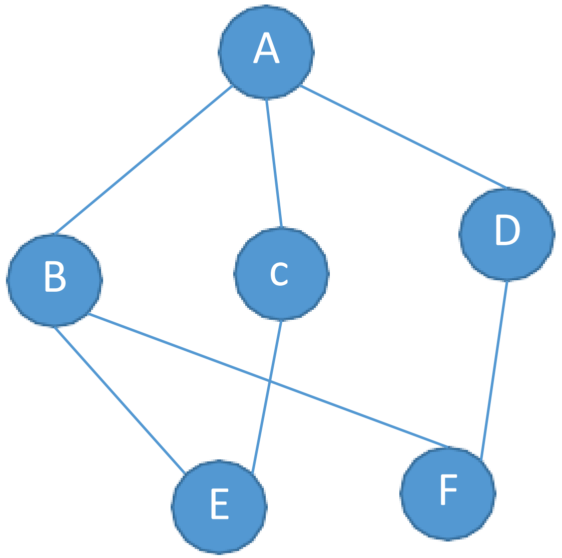




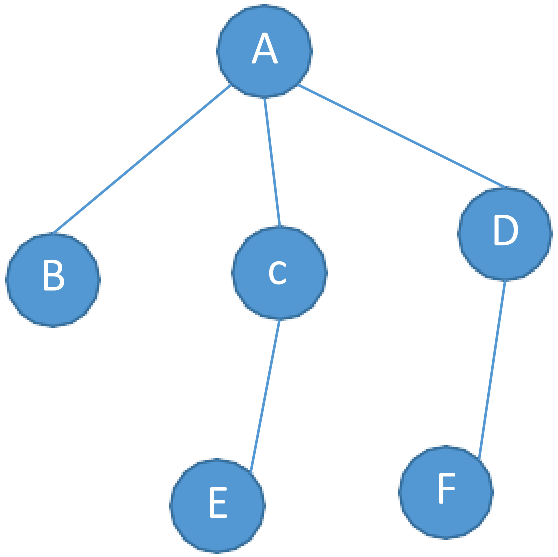
G的三个强连通分量

7.1 图的定义和术语

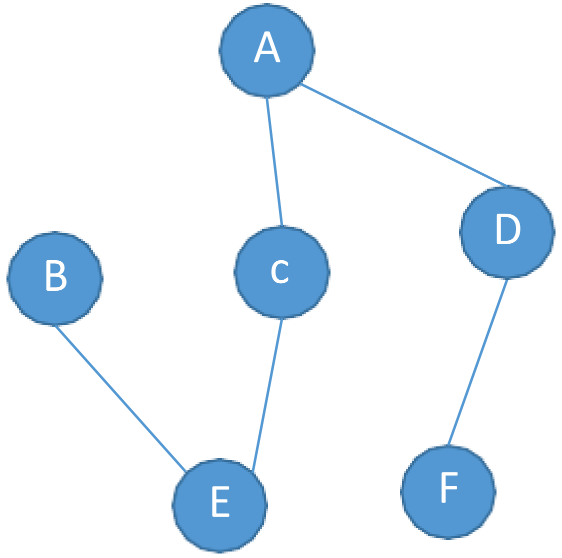
连通图的生成树是包含图中全部顶点的一个极小连通子图。若图中顶点数为n，则它的生成树含有n-1 条边。



G



G的生成树1

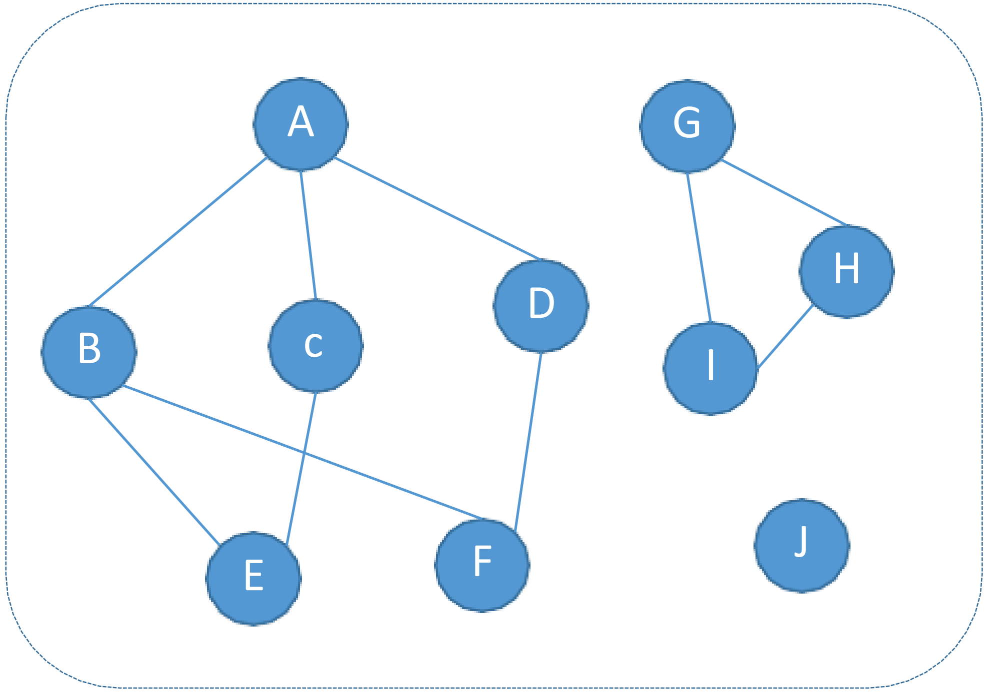


G的生成树2

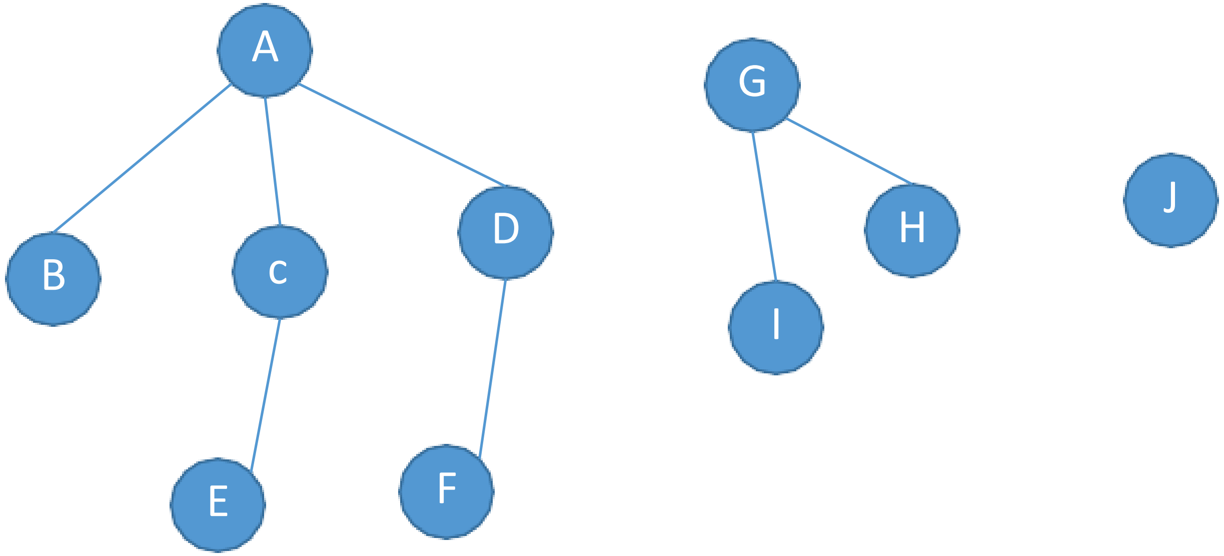
……

7.1 图的定义和术语

在非连通图中，连通分量的生成树构成了非连通图的生成森林。



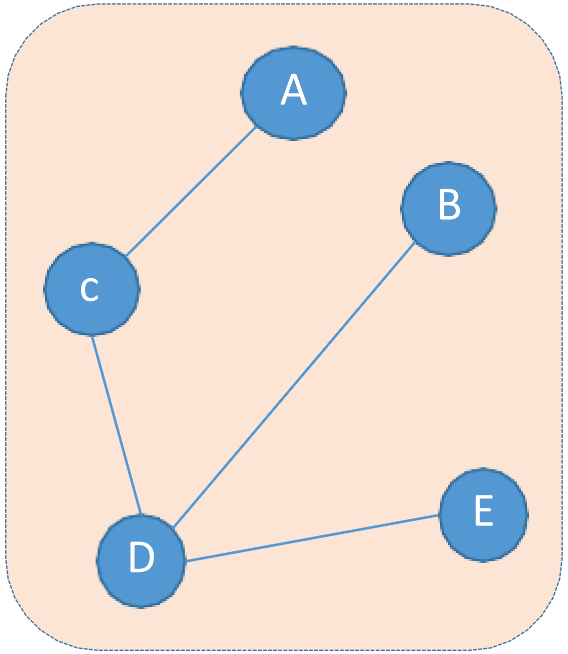
G

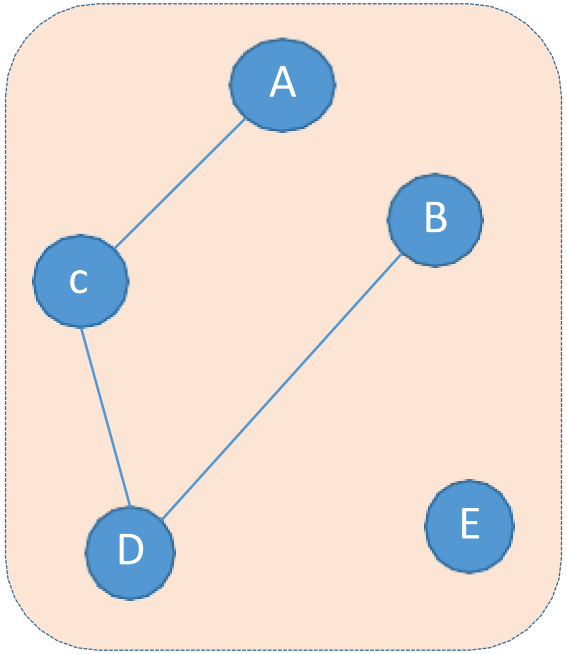


G生成森林

7.1 图的定义和术语

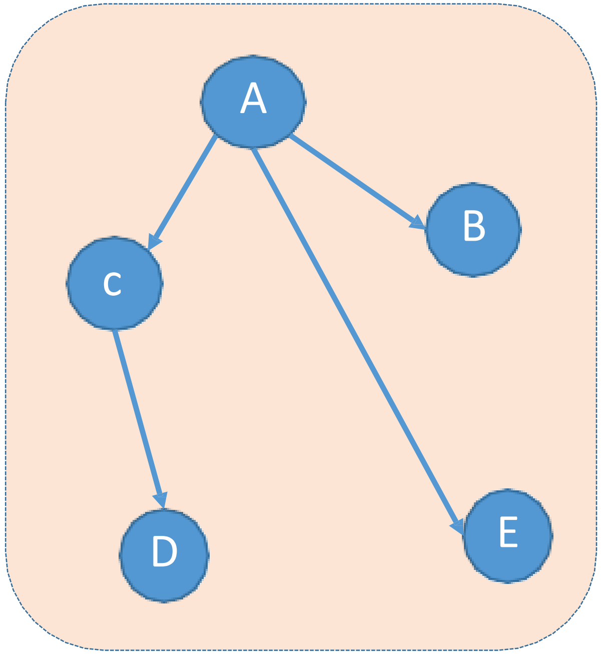
树





森林

有向树



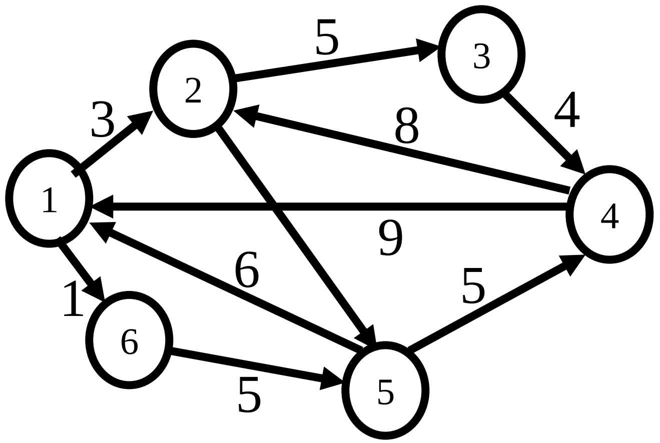
树——不存在回路，且连通的无向图

有向树——一个顶点的入度为0、其余顶点的 入度均为1的有向图，称为有向树。

7.1 图的定义和术语

- **边的权、带权图**

弧或边的权——在一个图中，每条弧或边都可以标上具有某种含义的数值，该数值称为该弧或边的权值。



弧或边带权的图分别称作有向网或无向网。

带权路径长度——当图是带权图时，一条路径上所有边的权值之和，称为该路径的带权路径长度

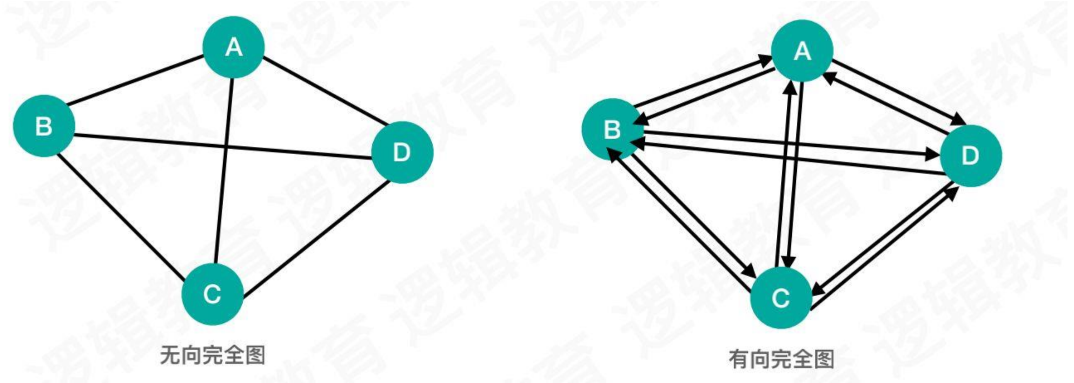
7.1 图的定义和术语

- 完全图、稀疏图、稠密图

假设图中有n个顶点，e条边，则

含有e=n(n-1)/2条边的无向图称作完全图；

含有e=n(n-1)条弧的有向图称作有向完全图；



若边或弧的个数很少（如e<nlogn），则称作稀疏图，反之称作稠密图。

7.1 图的定义和术语

- 图
  - 无向图-边、度、**连通图**、**连通分量**、**生成树**、生成森林
  - 有向图-弧、弧头、弧尾、**出度**、**入度**、强连通图、强连通分量、有向树
  - 路径-简单路径、简单回路、**路径长度**、**点到点的距离**、**带权路径长度**
  - 完全图、稀疏图、稠密图

7.1 图的定义和术语

图的抽象数据类型

【ADT】 Graph G = (V , R)

数据对象V：V是具有相同特性的数据元素的集合，称为顶点集。

数据关系R：

R = { VR }

VR = { <v , w >|v, w ∈V,且P(v,w),<v, w>表示从v到w的弧，

谓词P(v, w)定义了弧 <v, w>的意义或信息 }

7.1 图的定义和术语

- **基本操作：**设图G=(V,E)，图上定义的基本操作如下：

CreatGraph(&G)： 建立图

DestroyGraph(&G)：销毁图

NewNode (G)：建立一个新顶点， V=V∪{v}

DelNone (G, v)：删除顶点v以及与之相关联的所有边

SetSucc (G, v1, v2):增加一条边， E = E∪(v1,v2) ,V=V

DelSucc (G, v1, v2)：删除边（v1,v2）,V不变

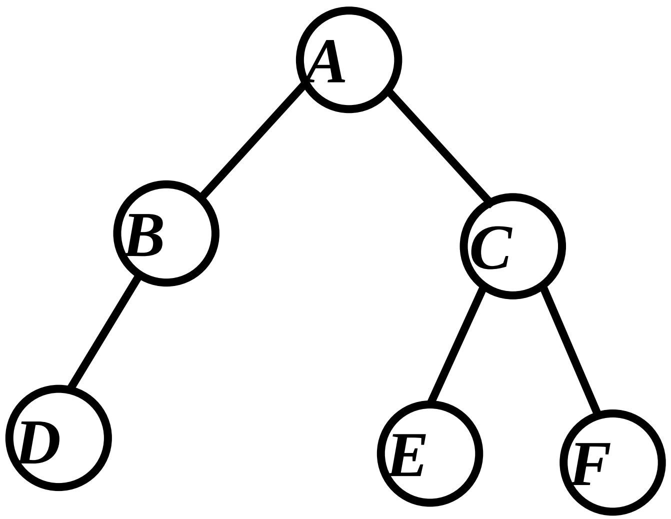
IsEdge (G, v1, v2)：判断（v1,v2）∈E

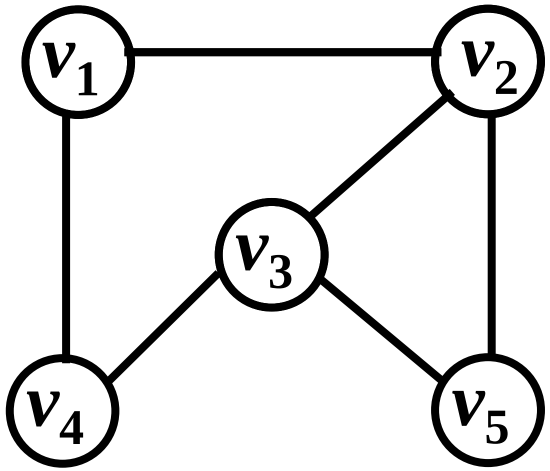
FirstAdjVex(G , v): 顶点v 的第一个邻接顶点

NextAdjVex(G, v, w)：顶点v 的某个邻接点w的下一个邻接顶点。

7.1 图的定义和术语

不同逻辑结构之间的比较

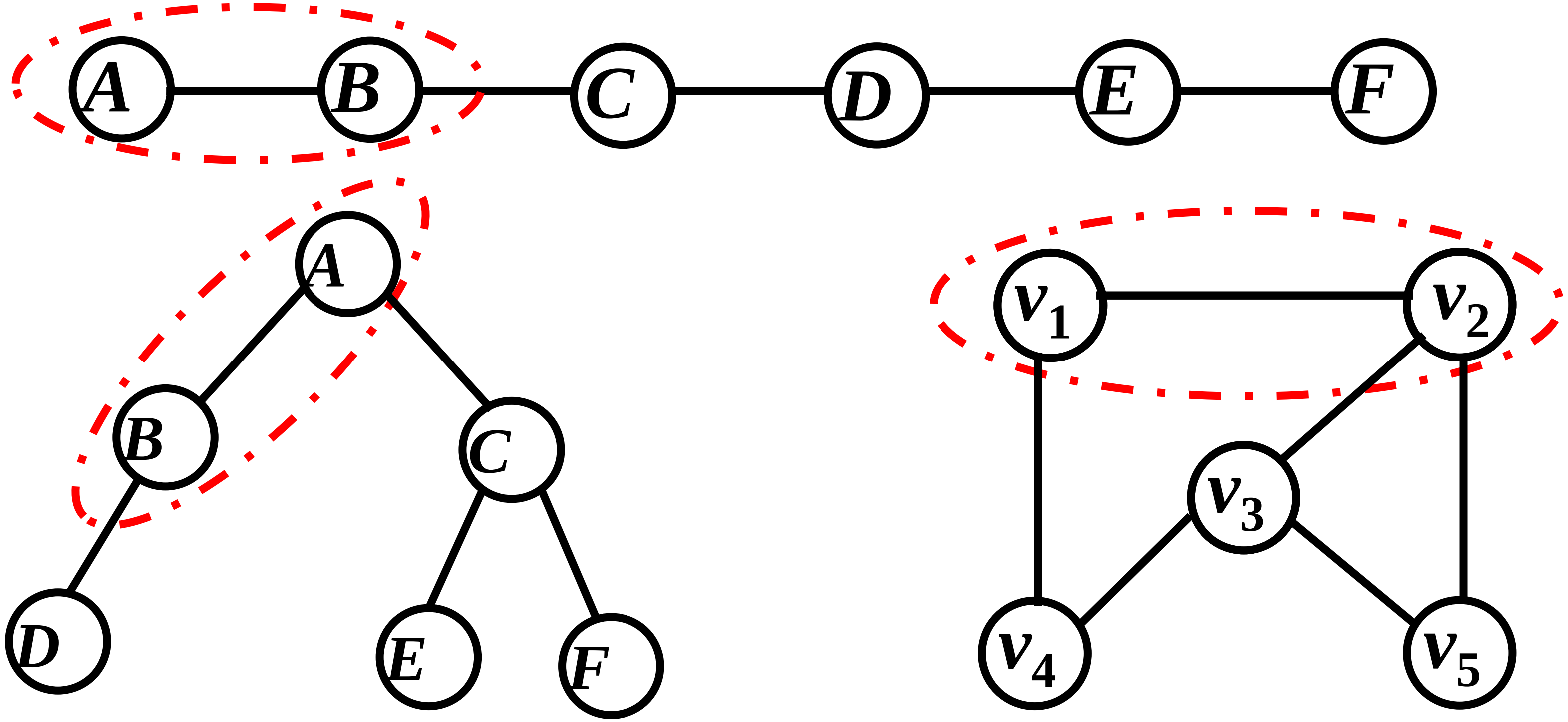




- 在线性结构中，数据元素之间仅具有线性关系(1:1);
- 在树型结构中，结点之间具有层次关系(1:m);
- 在图型结构中，任意两个顶点之间都可能有关系(m:n)。

7.1 图的定义和术语

不同逻辑结构之间的比较



- 在线性结构中，元素之间的关系为前驱和后继；
- 在树型结构中，结点之间的关系为双亲和孩子；
- 在图型结构中，顶点之间的关系为邻接。

2023/4/24