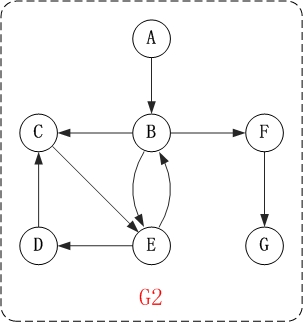
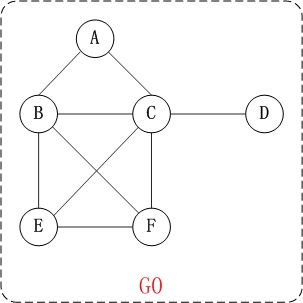
# 图的基本概念

### 1.图的定义

定义：图（graph）是由一些点（vertex）和这些点之间的连线（edge）所组成的；其中，点通常被称为“顶点（vertex）”，而点与点之间的连线则被称为“边或弧（edge）“。通常记为G=（V，E）

### 2.图的种类

根据边是否有方向，将图分为：无向图和有向图



### 3邻接点和度

#### 3.1邻接点

一条边上的两个顶点叫邻接点

在有向图中，除了邻接点之外，还有“入边“和”出边“的概念

顶点的入边，是指以该顶点为终点的边

顶点的出边，是指以该顶点为起点的边

#### 3.2度

在无向图中，某个顶点的度是邻接到该顶点的边（或弧）的数目

在有向图中，度还有“入度“和”出度“之分

某个顶点的入度，是指以该顶点为终点的边的数目

某个顶点的出度，是指以该顶点为起点的边的数目

顶点的度=入度+出度

### 4.路径和回路

路径：如果顶点V1到顶点V2之间存在一个顶点序列，则表示V1到V2是一条路径

路径长度：路径中边的数量

简单路径：若一条路径上顶点不重复出现，则是简单路径

回路：若路径的第一个顶点和最后一个顶点相同，则是回路

简单回路：路径的第一个顶点和最后一个顶点相同，其他各顶点都不重复的回路则是简单回路

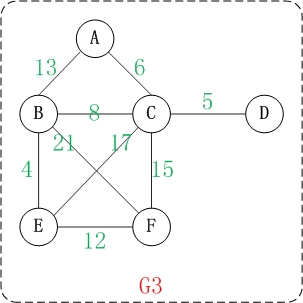
### 5.连通图和连通分量

连通图：对于无向图而言，任意两个顶点之间都存在一条无向路径，则称该无向图为连通图；

对于有向图而言，若图中任意两个顶点之间都存在一条有向路径，则称该有向图为强连通图

连通分量：非连通图中的各个连通子图称为改图的连通分量

### 6.权



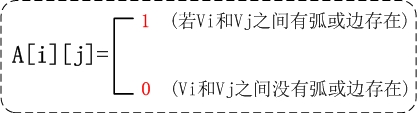
# 图的存储结构

图的存储结构，常用的是邻接矩阵和邻接表

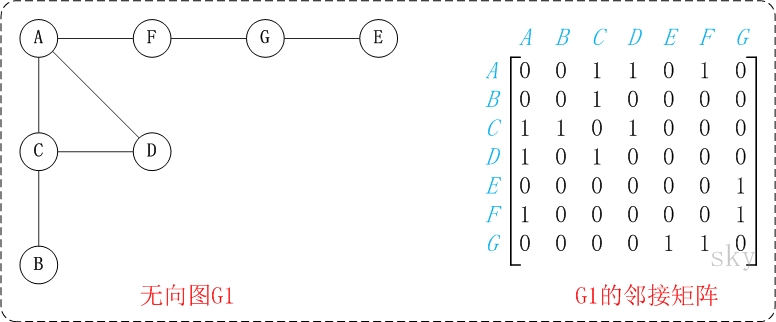
### 1.邻接矩阵

邻接矩阵是指用矩阵来表示图。它是采用矩阵来描述图中顶点之间的关系（即弧或边的权）

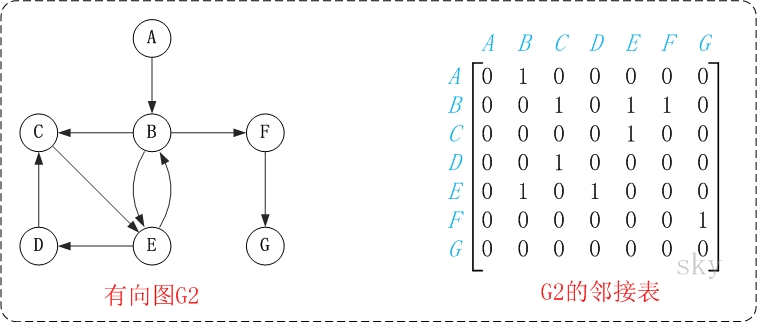
假设图中顶点数为n，则邻接矩阵定义为：



无向图的示意图：



有向图的示意图：



通常用两个数组来实现邻接矩阵：

一个一维数组来保存顶点信息，一个二维数组来保存边的信息

缺点：比较消耗内存

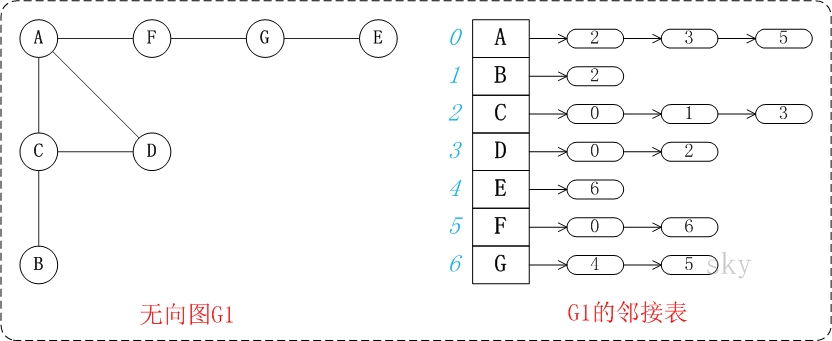
基本定义

1. **public** **class** MatrixUDG {
3. **private** **char**[] mVexs;       // 顶点集合
4. **private** **int**[][] mMatrix;    // 邻接矩阵
6. ...
7. }

### 邻接表

邻接表是图的一种链式存储表示方法。它是改进后的“邻接矩阵“。缺点是不方便判断两个顶点之间是否有边，但是更省空间

无向图的示意图



基本定义

1. **public** **class** ListUDG {
2. // 邻接表中表对应的链表的顶点
3. **private** **class** ENode {
4. **int** ivex;       // 该边所指向的顶点的位置
5. ENode nextEdge; // 指向下一条弧的指针
6. }
8. // 邻接表中表的顶点
9. **private** **class** VNode {
10. **char** data;          // 顶点信息
11. ENode firstEdge;    // 指向第一条依附该顶点的弧
12. };
14. **private** VNode[] mVexs;  // 顶点数组
16. ...
17. }

有向图的示意图

