第9节 图

图(Graph)是一种复杂的非线性结构。在人工智能、工程、数学、物理、化学、生物和计算机科学等领域中,图结构有着广泛的应用。

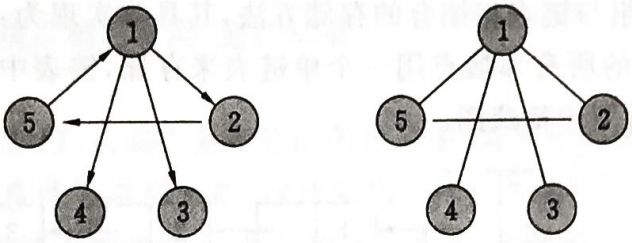
一、图的定义

点用边连起来就叫作图,严格意义上讲,图是一种数据结构,定义为:graph=(V,E)。V是一个非空有限集合,代表顶点(结点),E代表边的集合。

二、图的相关概念

1.有向图:图的边有方向,只能按箭头方向从一点到另一点。(a)就是一个有向图。

2.无向图:图的边没有方向,可以双向。(b)就是一个无向图。



(a) (b)

3.结点的度:无向图中与结点相连的边的数目,称为结点的度。

4.结点的入度:在有向图中,以这个结点为终点的有向边的数目。

5.结点的出度:在有向图中,以这个结点为起点的有向边的数目。

6.权值:边的"费用",可以形象地理解为边的长度。

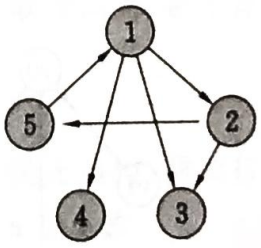
7.连通:如果图中结点U,V之间存在一条从U通过若干条边、点到达V的通路,则称U、V是连通的。

8.回路:起点和终点相同的路径,称为回路,或"环"。

9.完全图:一个n阶的完全无向图含有n*(n-1)/2条边;一个n阶的完全有向图含有n*(n-1)条边;

10.稠密图:一个边数接近完全图的图。

11.稀疏图:一个边数远远少于完全图的图。



12.强连通分量:有向图中任意两点都连通的最大子图。右图中,1-2-5构成一个强连通分量。特殊地,单个点也算一个强连通分量,所以右图有三个强连通分量:1-2-5,4,3。

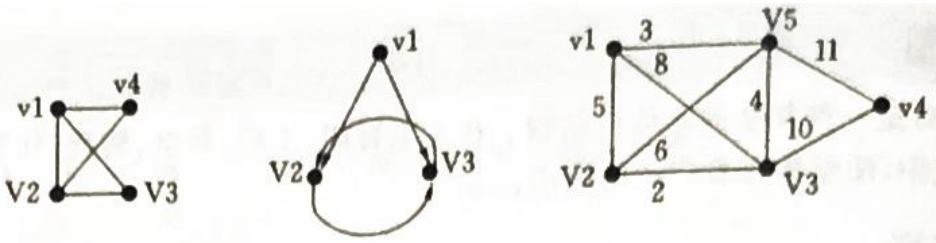
三、图的存储结构

1.二维数组邻接矩阵存储

图的邻接矩阵存储方式是用一个二维数组(称邻接矩阵)存储图中的边或弧的信息。它适用于稠密图。

定义int G[101][101],其中G[i][j]表示从点i到点j的边的权值,定义如下:

$$G[i][j]=\begin{cases}1\text{或权值} & \text{当 } v_i \text{ 与 } v_j \text{ 之间有边或弧时,取值为1或权值}\\ 0\text{或}\infty & \text{当 } v_i \text{ 与 } v_j \text{ 之间无边或弧时,取值为0或}\infty\text{(无穷大)}\end{cases}$$

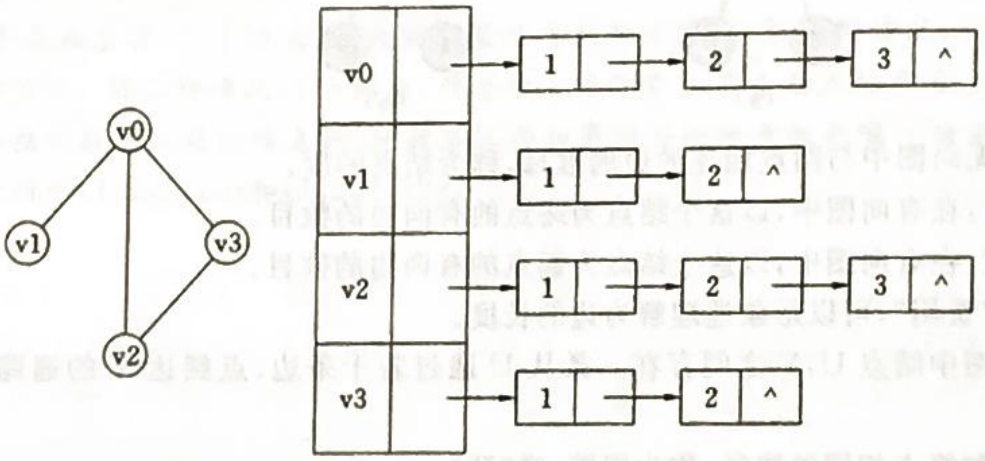


上图中的 3 个图对应的邻接矩阵分别如下：

$$G(A)=\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&0\\1&1&0&0\end{bmatrix}\quad G(B)=\begin{bmatrix}0&1&1\\0&0&1\\0&1&0\end{bmatrix}\quad G(C)=\begin{bmatrix}\infty&5&8&\infty&3\\5&\infty&2&\infty&6\\8&2&\infty&10&4\\\infty&\infty&10&\infty&11\\3&6&4&11&\infty\end{bmatrix}$$

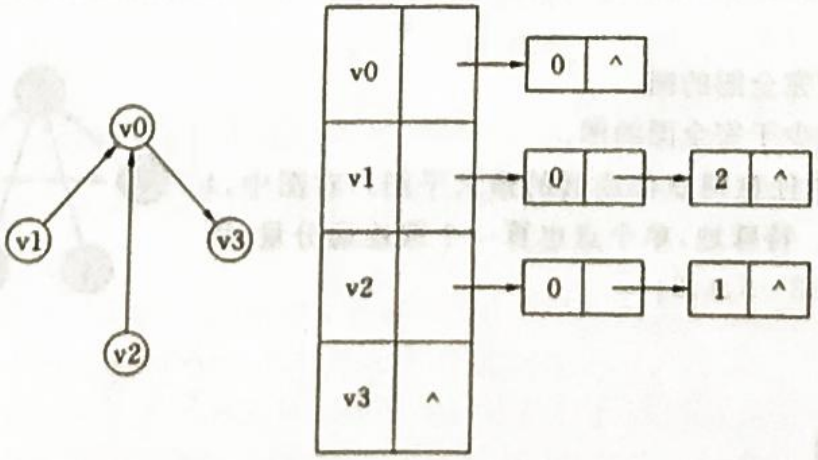
2.邻接表存储结构

邻接表是一种将数组与链表相结合的存储方法，其具体实现为：将图中顶点用一个一维数组存储，每个顶点 Vi 的所有邻接点用一个单链表来存储，链表中存放与当前结点相邻的结点在数组中的下标，适用于稀疏图。



无向图第i个链表的边结点数目正好是第i个顶点的度。

对于具有n个结点、e条边的无向图，邻接表中数组有n个顶点结点、链表有2e个边结点。



有向图第i个链表的边结点数目正好是第i个顶点的出度。

对于具有n个结点、e条边的有向图，邻接表中数组有n个顶点结点、链表有e个边结点。

四、深度优先与广度优先遍历

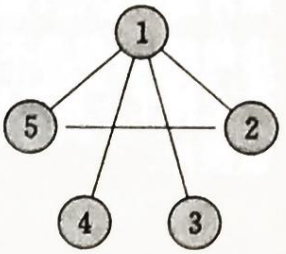
从图中某一顶点出发系统地访问图中所有顶点，使每个顶点恰好被访问一次，这种运算操作被称为图的遍历。为了避免重复访问某个顶点，可以设一个标志数组 visited [i]，未访

问时值为 false，访问一次后改为 true。图的遍历分为深度优先遍历和广度优先遍历两种方法，两者的时间效率都是 O(n * n)。

1.深度优先遍历

深度优先遍历与深度优化搜索相似，从一个点 A 出发，将这个点标为已访问 visited [i] =true，然后再访问所有与之相连且未被访问过的点。当 A 的所有邻接点都被访问过后，退回到 A 的上一个点(假设是 B)，再从 B 的另一个未被访问的邻接点出发，继续遍历。

例如对右边的这个无向图深度优先遍历，假定先从 1 出发。程序按如下顺序遍历：



a.1→2→5，然后退回到 2，退回到 1。

b.从 1 开始再访问未被访问过的点 3，3 没有未访问的邻接点，退回 1。

c.再从 1 开始访问未被访问过的点 4，再退回 1。

d.起点 1 的所有邻接点都已访问，遍历结束。

2.广度优先遍历

广度优先遍历并不常用，从编程复杂度的角度考虑，通常采用的是深度优先遍历。

广度优先遍历和广度优先搜索相似，因此使用广度优先遍历一张图并不需要掌握新的知识，在原有的广度优先搜索的基础上，做一些小的修改，就变成广度优先遍历算法。

五、一笔画问题

如果一个图存在一笔画，则一笔画的路径叫作欧拉路，如果最后又回到起点，那这个路径叫作欧拉回路。

定义奇点是指跟这个点相连的边数目有奇数个的点。对于能够一笔画的图，有以下两个定理。

定理 1：存在欧拉路的条件，图是连通的，有且只有 2 个奇点。

定理 2：存在欧拉回路的条件，图是连通的，有 0 个奇点。

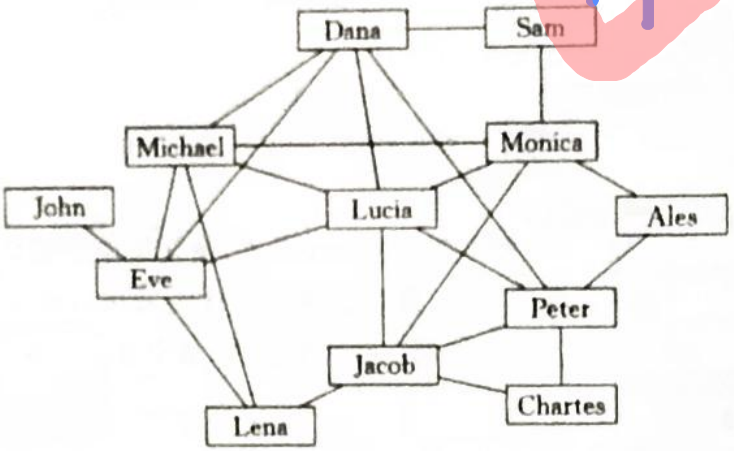
两个定理的正确性是显而易见的，既然每条边都要经过一次，那么对于欧拉路，除了起点和终点外，每个点如果进入了一次，一定要出去一次，显然是偶点。对于欧拉回路，每个点进入和出去次数一定都是相等的，显然没有奇点。

求欧拉路的算法很简单，使用深度优先遍历即可。

根据一笔画的两个定理，如果寻找欧拉回路，对任意一个点执行深度优先遍历；找欧拉路，则对一个奇点执行深度优先搜索，时间复杂度为 O(m+n)，m 为边数，n 是点数。

课 堂 练 目

1.【NOIP2016】Lucia 和她的朋友以及朋友的朋友都在某社交网站上注册了账号。下图是他们之间的关系圈，两个人之间有边相连代表这两个人是朋友，没有边相连代表不是朋友，这个社交网站的规则是：如果某人 A 向他（她）的朋友 B 分享了某张照片，那么 B 就可以对该照片进行评论：如果 B 评论了该照片，那么他（她）的所有朋友都可以看见这个评论以及被评论的照片，但是不能对该照片进行评论（除非 A 也向他（她）分享了该照片）。现在 Lucia 已经上传了一张照片，但是她不想让 Jacob 看见这张照片，那么她可以向以下朋友（ ）分享该照片。



A. Dana，Michael，Eve　　B. Dana，Eve，Monica

C. Michael，Eve，Jacob　　D. Micheal，Peter，Monica

2.【NOIP2014】有向图中每个顶点的度等于该顶点的（）

A.入度　B.出度　C.入度与出度之和　D.入度与出度之差

3.【NOIP2014】在无向图中，所有顶点的度数之和是边数的（ ）倍。

A.0.5　B.1　C.2　D.4

4.【NOIP2016】设简单无向图 G 有 16 条边且每个顶点的度数都是 2，则图 G 有（）个顶点.

A.10　B.12　C.8　D.16

5.【NOIP2010】关于拓扑排序，下面说法正确的是（）

A.所有连通的有向图都可以实现拓扑排序

B.对一个图而言，拓扑排序的结果是唯一的

C.拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面

D.拓扑排序结果序列中的第一个结点一定是入度为 0 的点

6.【NOIP2008】设 T 是一棵有 n 个顶点的树，下列说法不正确的是（）。

A.T 有 n 条边　B.T 是连通的　C.T 是无环的　D.T 有 n−1 条边

7.【NOIP2017】设 G 是有 n 个结点、m 条边（n≤m）的连通图，必须删去 G 的（）条边，才能使得 G 变成一棵树。

A.m-n+1　B.m-n　C.m+n+1 D.n-m+1

8.【NOIP2015】6 个顶点的连通图的最小生成树，其边数为（）

A.6　B.5　C.7　D.4

9.【NOIP2014】设 G 是有 6 个结点的完全图，要得到一棵生成树，需要从 G 中删去（）条边。

A.6　B.9　C.10　D.15

10.【NOIP2009】已知 n 个顶点的有向图，若该图是强连通的（从所有顶点都存在路径到达其他顶点），则该图中最少有（）条有向边。

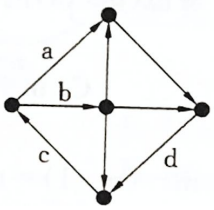
成环

A.n　B.n + 1　C.n-1　D.n*(n-1)

11.【NOIP2011】无向完全图是图中每对顶点之间都恰有一条边的简单图。已知无向完全图 G 有 7 个顶点，则它共有（）条边。

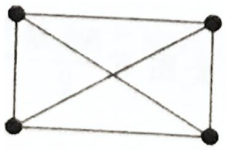
A.7　B.21　C.42　D.49

12.【NOIP2011】对一个有向图而言，如果每个结点都存在到达其他任何结点的路径，那么就称它是强连通的。例如，下图就是一个强连通图。事实上，在删掉边（）后，它依然是强连通的。



A.a　B.b　C. c　D.d

13.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。图是一个有 4 个顶点、6 条边的连通图。若要使它不再是连通图，至少要删去其中的（）条边。



A.1　B.2　C.3　D.4

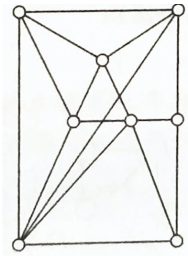
14.【NOIP2013】在一个无向图中，如果任意两点之间都存在路径相连，则称其为连通图。右图是一个有 5 个顶点、8 条边的连通图。若要使它不再是连通图，至少要删去其中的（）条边。

A.2　B.3　C.4　D.5

15.【NOIP2013】二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，12 个顶点的二分图至多有（）条边。

A.18　B.24　C.36　D.66

16.【NOIP2015】对图 G 中各个结点分别指定一种颜色，使相邻结点颜色不同，则称为图 G 的一个正常着色。正常着色图 G 所必需的最少颜色数，称为 G 的色数。那么下图的色数是()。



A.3　B.4　C.5　D.6

17.【NOIP2016】G 是一个非连通简单无向图，共有 28 条边，则该图至少有（）个顶点。

A.10　B.9　C.8　D.7

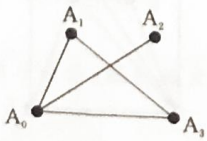
18.【NOIP2017】由四个不同的点构成的简单无向连通图的个数是（）。

A.32　B.35　C.38　D.41

19.【NOIP2018】由四个没有区别的点构成的简单无向连通图的个数是（）。

A.6　B.7　C.8　D.9

20.【NOIP2013】以 A_0 作为起点，对下面的无向图进行深度优先遍历时，遍历顺序不可能是(　　)。



A. A_0, A_1, A_2, A_3　　B. A_0, A_1, A_3, A_2

C. A_0, A_2, A_1, A_3　　D. A_0, A_3, A_1, A_2

21.【NOIP2015】具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历和广度优先遍历运算的时间复杂度均为(　　)。

A. $O(n^2)$　　B. $O(e^2)$　　C. $O(ne)$　　D. $O(n+e)$

22.【NOIP2013】对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果不使用堆或其他优先队列进行优化，则其时间复杂度为(　　)。

A. $O(mn+n^3)$　　B. $O(n^2)$　　C. $O((m+n)\log n)$　　D. $O((m+n^2)\log n)$

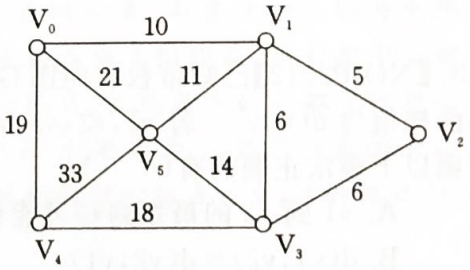
23.【NOIP2009】右图给出了一个加权无向图，从顶点 V_0 开始用 prim 算法求最小生成树，则依次加入最小生成树的顶点集合的顶点序列为(　　)。

A. V0, V1, V2, V3, V5, V4

B. V0, V1, V5, V4, V3, V3

C. V1, V2, V3, V0, V5, V4

D. V1, V2, V3, V0, V4, V5



不定项选择题

1.【NOIP2014】以下哪些结构可以用来存储图（）

A.邻接矩阵 B.栈 C. 邻接表 D.二叉树

2.【NOIP2009】若 3 个顶点的无权图 G 的邻接矩阵用数组存储为 {{0,1,1},{1,0,1},{0,1,0}}，假定在具体存储中顶点依次为 v1，v2，v3，关于该图，下面的说法正确的是（）

A.该图是有向图　B.该图是强连通的

C.该图所有顶点的入度之和减所有顶点的出度之和等于 1

D.从 v1 开始的深度优先遍历所经过的顶点序列与广度优先的顶点序列是相同

3.【NOIP2012】已知带权有向图 G 上的所有权值均为正整数，记顶点 u 到顶点 v 的最短路径的权值为 d（u，v）。若 vl，v2，v3，v4，v5 是图 G 上的顶点，且它们之间两两都存在路径可达,则以下说法正确的有（）

A.v1 到 v2 的最短路径可能包含一个环

B.d(vl,v2)=d(v2,v1)

C.d(vl,v3)≤d(vl,v2) + d(v2,v3)

D.如果 vl→v2→v3→v4→v5 是 v1 到 v5 的一条最短路径，那么 v2→v3→v4 是 v2 到 v4 的一条最短路径

4.【NOIP2018】下列关于最短路算法的说法，正确的是（）

A.当图中不存在负权回路但是存在负权边时，Dikstra 算法不一定能求出源点到所有点的最短路

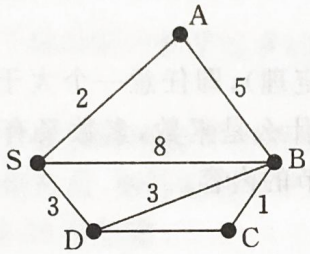
B.当图中不存在负权边时，调用多次 Dijkstra 算法能求出每对顶点间最短路径、

C.图中存在负权回路时，调用一次 Dijkstra 算法也一定能求出源点到所有点的最短路

D.当图中不存在负权边时，调用一次 Dijkstra 算法不能用于每对顶点间最短路计算

6.【NOIP2011】对右图使用 Dijkstra 算法计算 S 点到其余各点的最短路径长度时，到 B 点的距离 d[B]初始时赋为 8，在算法的执行过程中还会出现的值有（ ）。

BCD



A. 3　　B. 7　　C. 6　　D. 5