

17 第六讲 图（上）

笔记本： 浙江大学《数据结构》

创建时间： 2025/4/12 16:18

更新时间： 2025/4/13 14:14

作者： panhengye@163.com

URL: <https://www.doubao.com/chat/2913230631981826>

什么是图（Graph）？

表示“多对多”的关系

线性表和树可以看作是图的特殊情况

包含：

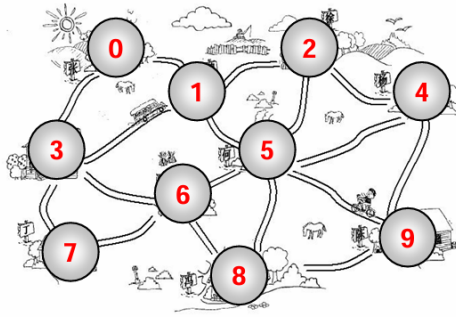
- 一组顶点：通常用 V (vertex) 表示顶点集合
- 一组边：通常用 E (edge) 表示边的集合
 - $(v, w) \rightarrow$ 无向边
 - $\langle v, w \rangle \rightarrow$ 有向边（单行线）
 - 不考虑重边和自回路

抽象数据类型定义

- **类型名称：**图（Graph）
- **数据对象集：** $G(V, E)$ 由一个非空的有限顶点集合 V 和一个有限边集合 E 组成。
- **操作集：**对于任意图 $G \in \text{Graph}$ ，以及 $v \in V$ ， $e \in E$
 - `Graph Create()`：建立并返回空图；
 - `Graph InsertVertex(Graph G, Vertex v)`：将 v 插入 G ；
 - `Graph InsertEdge(Graph G, Edge e)`：将 e 插入 G ；
 - `void DFS(Graph G, Vertex v)`：从顶点 v 出发深度优先遍历图 G ；
 - `void BFS(Graph G, Vertex v)`：从顶点 v 出发宽度优先遍历图 G ；
 - `void ShortestPath(Graph G, Vertex v, int Dist[])`：计算图 G 中顶点 v 到任意其他顶点的最短距离；
 - `void MST(Graph G)`：计算图 G 的最小生成树；
 -

邻接矩阵表示法

$$G[i][j] = \begin{cases} 1 & \text{若 } \langle v_i, v_j \rangle \text{ 是 } G \text{ 中的边} \\ 0 & \text{否则} \end{cases}$$



	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_0	0	1	0	1	0	0	0	0	0	0
v_1	1	0	1	1	0	1	0	0	0	0
v_2	0	1	0	0	1	1	0	0	0	0
v_3	1	1	0	0	0	0	1	1	0	0
v_4	0	0	1	0	0	1	0	0	0	1
v_5	0	1	1	0	1	0	1	0	1	1
v_6	0	0	0	1	0	1	0	1	1	0
v_7	0	0	0	1	0	0	1	0	0	0
v_8	0	0	0	0	0	1	1	0	0	1
v_9	0	0	0	0	1	1	0	0	1	0



Copyright © 2014, 浙江大学计算机科学与技术学院
All Rights Reserved

如何节省一半的空间呢？

- 将问题转化为如何求总容量，观察这个矩阵：设行数为 i ，总数为 n
 - $i = 1$ 时, $n = 1$
 - $i = 2$ 时, $n = 3$
 - $i = 3$ 时, $n = 6$
 - $i = 4$ 时, $n = 10$
- 观察 n 之间的关系，发现是 $a_1 = 2, d = 1$ 的等差数列
- 所以原数列是二阶等差数列（差分的差分是常数），其通项公式应为二次函数形式

设通项公式为 $a_n = An^2 + Bn + C$ ，代入前三项的值：

1. 当 $n = 1$ 时: $A(1)^2 + B(1) + C = 1$, 即 $A + B + C = 1$;
2. 当 $n = 2$ 时: $A(2)^2 + B(2) + C = 3$, 即 $4A + 2B + C = 3$;
3. 当 $n = 3$ 时: $A(3)^2 + B(3) + C = 6$, 即 $9A + 3B + C = 6$.

解方程组：

- 用第 2 式减第 1 式: $(4A + 2B + C) - (A + B + C) = 3 - 1$, 得 $3A + B = 2$;
- 用第 3 式减第 2 式: $(9A + 3B + C) - (4A + 2B + C) = 6 - 3$, 得 $5A + B = 3$;
- 再用新得到的两式相减: $(5A + B) - (3A + B) = 3 - 2$, 得 $2A = 1$, 即 $A = \frac{1}{2}$;
- 代入 $3A + B = 2$: $3 \times \frac{1}{2} + B = 2$, 得 $B = \frac{1}{2}$;
- 代入第 1 式: $\frac{1}{2} + \frac{1}{2} + C = 1$, 得 $C = 0$.

因此，通项公式为: $a_n = \frac{1}{2}n^2 + \frac{1}{2}n = \frac{n(n+1)}{2}$

但是如何解决查找问题呢？

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_0	0	1	0	1	0	0	0	0	0	0
v_1	1	0	1	1	0	1	0	0	0	0
v_2	0	1	0	0	1	1	0	0	0	0
v_3	1	1	0	0	0	0	1	1	0	0
v_4	0	0	1	0	0	1	0	0	0	1
v_5	0	1	1	0	1	0	1	0	1	1
v_6	0	0	0	0	0	1	0	1	1	0
v_7	0	0	0	1	0	0	1	0	0	0
v_8	0	0	0	0	0	1	1	0	0	1
v_9	0	0	0	0	1	1	0	0	1	0

用一个长度为 $N(N+1)/2$ 的1维数组A存储 $\{G_{00}, G_{10}, G_{11}, \dots, G_{n-1, 0}, \dots, G_{n-1, n-1}\}$, 则 G_{ij} 在A中对应的下标是:

$$(i * (i+1) / 2 + j)$$

对于网络, 只要把 $G[i][j]$ 的值定义为边 $\langle v_i, v_j \rangle$ 的权重即可。

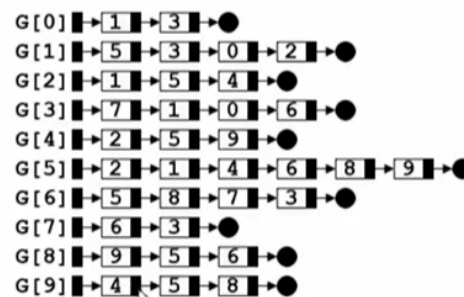
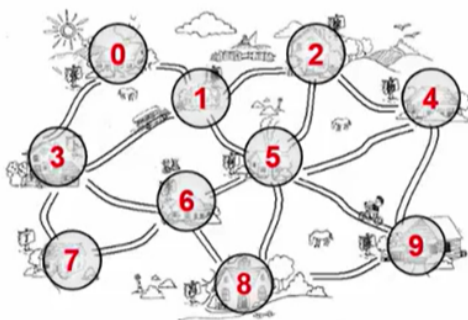
问题: v_i 和 v_j 之间若没有边该怎么表示?

思考邻接矩阵表示法的利弊

- 好处
 - 直观易懂
 - 方便检查任意一对顶点间是否存在边
 - 方便检查任意一顶点的所有“邻接点” (有边直接相连的顶点)
 - 方便计算任意顶点的度 (degree)
 - 概念:
 - 从该点出发: 出度
 - 指向该点: 入度
 - 使用:
 - 无向图: 对应行 (或列) 非0元素的个数
 - 有向图: 对应行非0元素的个数是“出度”; 对应列非0元素的个数是“入度”
- 坏处
 - 有大量的0:
 - 存放稀疏图的时候浪费空间
 - 存放稠密图 (尤其是完全图) 很合算
 - 浪费时间

邻接表表示法

- 邻接表: $G[N]$ 为指针数组, 对应矩阵每行一个链表, 只存非0元素



优缺点:

- 优点

- 方便找任一顶点的所有“邻接点”
 - 节约稀疏图的空间
 - 方便计算无向图的度
- 缺点
 - 对于有向图，只能计算“出度”；需要构造“逆邻接表”来方便计算“入度”
 - 检查任意一对顶点间是否存在边时非常困难

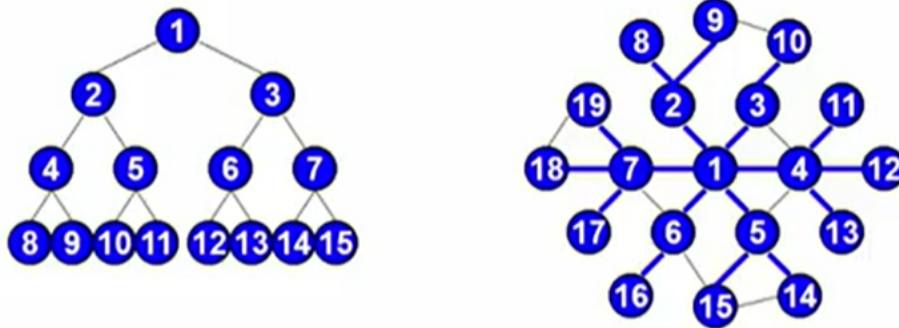
图的遍历

深度优先搜索 - depth first search

对应树的遍历中的先序

广度优先搜索 - breadth first search

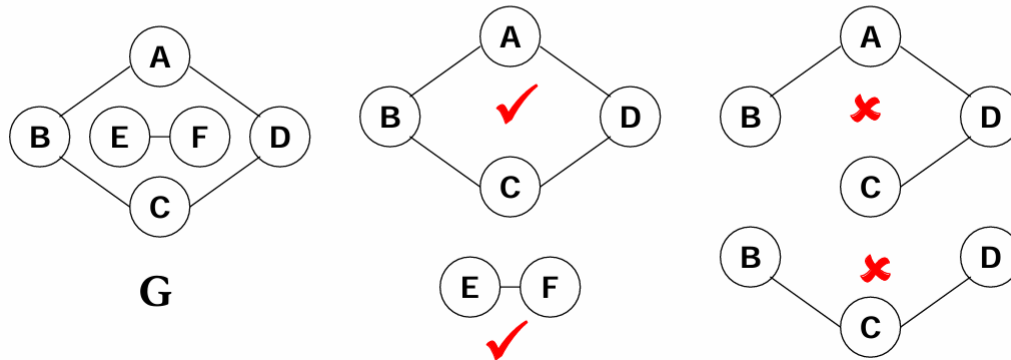
对应树中的层序遍历



为什么要有两种遍历方法？

- **连通**：如果从V到W存在一条（无向）**路径**，则称V和W是连通的
- **路径**：V到W的路径是一系列顶点 $\{V, v_1, v_2, \dots, v_n, W\}$ 的集合，其中任一对相邻的顶点间都有图中的边。**路径的长度**是路径中的边数（如果带权，则是所有边的权重和）。如果V到W之间的所有顶点都不同，则称**简单路径**
- **回路**：起点等于终点的路径
- **连通图**：图中任意两顶点均连通

- **连通分量**：无向图的极大连通子图
 - 极大顶点数：再加1个顶点就不连通了
 - 极大边数：包含子图中所有顶点相连的所有边



那么对于有向图呢？

- **强连通**：有向图中顶点 V 和 W 之间存在双向路径，则称 V 和 W 是强连通的
- **强连通图**：有向图中任意两顶点均强连通
- **强连通分量**：有向图的极大强连通子图

