数据结构之图

# 图

## 图的定义

图G由两个集合V(顶点Vertex)和E(边Edge)组成，定义为G=(V,E)。

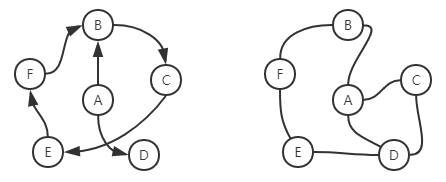
## 图的表示

**邻接矩阵**：原理就是用两个数组，一个数组保存顶点集，一个数组保存边集。

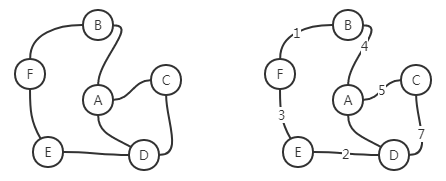
**邻接表**：邻接表是图的一种链式存储结构。这种存储结构类似于树的孩子链表。对于图G中每个顶点Vi，把所有邻接于Vi的顶点Vj链成一个单链表，这个单链表称为顶点Vi的邻接表。

## 图的分类

1. 有向图 和无向图



1. 有权图 和 无权图



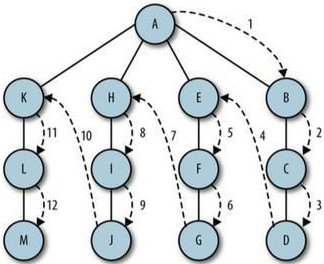
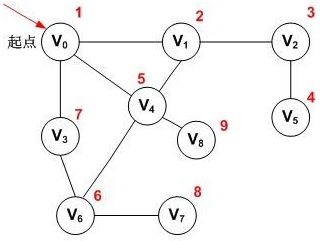
## 解决问题

1. 图的遍历（DFS和BFS）：联通分量、Flood Fill、寻路、走迷宫、迷宫生成、无权图的最短路径、环的判断等
2. 最小生成树问题（Minimum Spanning Tree）： Prim算法、Kruskal算法等
3. 最短路径问题（Shortest Path）：Dijkstra算法、Bellman-Ford算法等
4. 拓扑排序（Topological sorting）

### 深度优先搜索遍历（Deep First Search）

深度优先搜索DFS遍历类似于树的先序遍历。遍历过程的节点用栈存！

1. 从图G中任意一顶点V为初始出发点，首先访问出发点V，并将其标记为已访问过。
2. 从V出发搜索V的每个邻接点W，若W未曾访问过，则以W作为新的出发点出发，继续进行深度优先遍历，直到图中所有和V有路径相通的顶点都被访问到。
3. 若图中仍有顶点未被访问，则另选一个未曾访问的顶点作为起点，重复上述步骤，直到图中所有顶点都被访问到为止。



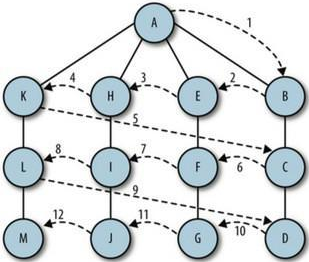
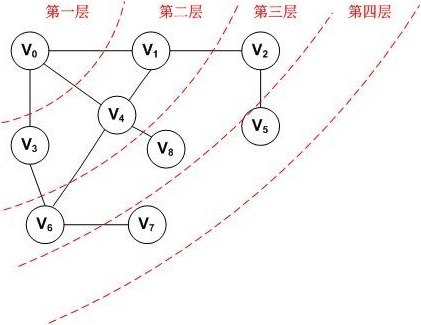
注：1）红色数字代表遍历的先后顺序；

1. 用邻接矩阵存储，时间复杂度为O(n2)；用邻接表存储，时间复杂度为O(n+e)；

### 广度优先搜索遍历（Breadth First Search）

广度优先搜索遍历BFS类似于树的按层次遍历。遍历过程的节点用队列存！

1. 从图G中任意一顶点V为初始出发点，首先访问出发点V，并将其标记为已访问过。
2. 从V出发搜索V的每个未被访问的邻接点（W1，W2，...，Wi），并标记为已访问。
3. 依次以Wi为顶点重复2，直到图中所有和V有路径相通的顶点都被访问到。
4. 若图中仍有顶点未被访问，则另选一个未曾访问的顶点作为起点，重复上述步骤，直到图中所有顶点都被访问到为止。



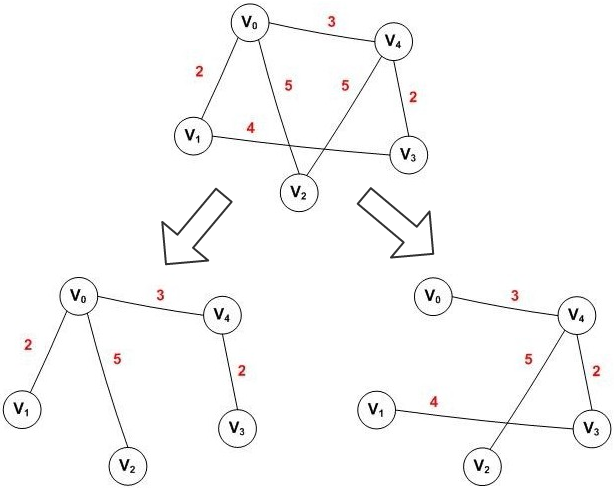
注：用邻接矩阵存储，时间复杂度为O(n2)；用邻接表存储，时间复杂度为O(n+e)；

### 最小生成树

生成树：将图中所有顶点以最少的边连通的子图。

所谓最小生成树，用一句话总结就是：权值和最小的生成树就是最小生成树。

如下图，左边生成树的权值和是12，右边生成树的权值和是14。故左边生成树是最小生成树。



注：那么如何构造最小生成树呢？可以使用普里姆算法。

<https://segmentfault.com/a/1190000010794621#articleHeader3>

<https://blog.csdn.net/u010700335/article/details/44941851>

<https://www.cnblogs.com/CherishFX/p/4608880.html>