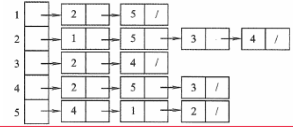


6.2图的存储及基本操作

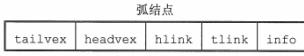
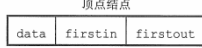
邻接矩阵法

- 邻接矩阵存储
 - 是指用一个一维数组存储图中顶点的信息,用一个二维数组存储图中边的信息 (即各顶点之间的邻接关系)
 - 存储顶点之间邻接关系的二维数组称为邻接矩阵
 - 结构示意
$$A[i][j] = \begin{cases} 1, & \text{若}(v_i, v_j) \text{或} \langle v_i, v_j \rangle \text{是} E(G) \text{中的边} \\ 0, & \text{若}(v_i, v_j) \text{或} \langle v_i, v_j \rangle \text{不是} E(G) \text{中的边} \end{cases}$$
 - 带权图示意
$$A[i][j] = \begin{cases} w_{ij}, & \text{若}(v_i, v_j) \text{或} \langle v_i, v_j \rangle \text{是} E(G) \text{中的边} \\ 0 \text{或} \infty, & \text{若}(v_i, v_j) \text{或} \langle v_i, v_j \rangle \text{不是} E(G) \text{中的边} \end{cases}$$
 - 邻接矩阵表示法的空间复杂度为 $O(n^2)$,其中 n 为图的顶点数 $|V|$
- 特点
 - 无向图的邻接矩阵一定是一个对称矩阵 (并且唯一)。因此,在实际存储邻接矩阵时只需存储上 (或下) 三角矩阵的元素
 - 对于无向图,邻接矩阵的第 i 行 (或第 i 列) 非零元素(或非0元素) 的个数正好是第 i 个顶点的度 $TD(v_i)$
 - 对于有向图,邻接矩阵的第 i 行 (或第 i 列) 非零元素 (或非0元素) 的个数正好是第 i 个顶点的出度 $OD(v_i)$
 - 用邻接矩阵法存储图,很容易确定图中任意两个顶点之间是否有边相连。但是,要确定图中有多少条边,则必须按行、按列对每个元素进行检测,所花费的时间代价很大
 - 稠密图适合使用邻接矩阵的存储表示

邻接表法

- 当一个图为稀疏图时,使用邻接矩阵法要浪费大量的存储空间,而图的邻接表法结合了顺序存储和链式存储方法, 减少了不必要的浪费
- 结构
 - 对图 G 中的每个顶点建立一个单链表,第 i 个单链表中的结点表示依附于顶点 v_i 的边,这个单链表就称为顶点 v_i 的边表 (对于有向图则称为出边表)
 - 边表的头指针和顶点的数据信息采用顺序存储 (称为顶点表),所以在邻接表中存在两种结点:顶点表结点和边表结点
- 示意图
- 特点
 - 若 G 为无向图,则所需的存储空间为 $O(|V| + 2|E|)$;若 G 为有向图,则所需的存储空间为 $O(|V| + |E|)$
 - 对于稀疏图,采用邻接表表示将极大地节省存储空间
 - 在邻接表中,给定一顶点,能很容易地找出它的所有邻边,因为只需要读取它的邻接表
 - 在有向图的邻接表表示中,求一个给定顶点的出度只需计算其邻接表中的结点个数
 - 但求其顶点的入度则需要遍历全部的邻接表
 - 图的邻接表表示并不唯一

十字链表

- 十字链表是有向图的一种链式存储结构。在十字链表中,对应于有向图中的每条弧有一个结点,对应于每个顶点也有一个结点
- 时间复杂度 $O(|V| + |E|)$
- 弧结点域结构
 - 尾域 (tailvex) : 弧尾
 - 头域 (headvex) : 弧头
 - 链域 (hlink) : 指向弧头相同的下一条弧
 - 链域 (tlink) : 指向弧尾相同的下一条弧
 - info 域 : 指向该弧的相关信息
- 顶点结点域结构
 - data域存放顶点相关的数据信息
 - firstin和firstout 两个域分别指向以该顶点为弧头或弧尾的第一个弧结点
- 只能存有向图

邻接多重表

- 邻接多重表是无向图的另一种链式存储结构
- 时间复杂度 $O(|V| + |E|)$
- 边结构
 - mark为标志域,可用以标记该条边是否被搜索过
 - ivex和jvex为该边依附的两个顶点在图中的位置
 - ilink指向下一条依附于顶点ivex的边;
 - jlink指向下一条依附于顶点jvex 的边
 - info为指向和边相关的各种信息的指针域
- 顶点结构
 - data域存储该顶点的相关信息
 - firstedge域指示第一条依附于该顶点的边
- 只能存无向图