数据结构与算法 课程复习提纲（二）

> 李明哲
>
> 计算学部金牌讲师团

第5章 树与二叉树

- 树的基本概念
- 二叉树
  - 二叉树的定义及其主要特征
  - 二叉树的顺序存储结构和链式存储结构
  - 二叉树的遍历
  - 线索二叉树的基本概念和构造
- 树、森林
  - 树的存储结构
  - 森林与二叉树的转换
  - 树和森林的遍历
- 树与二叉树的应用
  - 哈夫曼树和哈夫曼编码
  - 并查集及其应用

5.1 树的基本概念

5.1.1 树的定义

- 树是一种分层结构

5.1.2 基本术语

- 树中一个结点的孩子个数称为该结点的度，树中结点的最大度数称为树的度
- 度大于 0 的结点称为分支结点（非终端节点），度为 0 的结点称为叶结点（终端节点）
- 树的层次从树根开始定义，根节点为第 1 层，它的子节点为第 2 层，以此类推；双亲在同一层的结点互为堂兄弟
- 结点的深度是从根节点开始自顶向下逐层累加的
- 结点的高度是从叶节点开始自底向上逐层累加的
- 树的高度（或深度）是树中结点的最大层数
- 有序树和无序树
  - 注意：有序树只是子树有序，并没有规定子树的位置
  - 例：度为 2 的有序树不是二叉树（因为若有序树的某个结点只有一个儿子，则无法区分它是左子树还是右子树）
- 路径和路径长度
  - 树中两个结点之间的**路径**是由这两个结点之间所经过的**结点序列**构成的，而**路径长度**是路径上所经过的**边的个数**

    - 注意：由于树中分支是有向的（从双亲指向孩子），所以树中的路径是从上向下的，同一双亲的两个孩子之间不存在路径
    - 树的路径长度是指树根到每个结点的路径长的总和
- 度为 m 的树中第 i 层上至多有 m^{i-1} 个结点 $(i \geqslant 1)$
- 高度为 h 的 m 叉树至多有 $\frac{(m^h-1)}{(m-1)}$ 个结点
- 具有 n 个结点的 m 叉树的最小高度为 $\lceil log_m(n(m-1)+1) \rceil$

5.1.3 树的性质

5.2 二叉树的概念

5.2.1 二叉树的定义及其主要特性

- 平衡二叉树：树上任意一个结点的左子树和右子树的深度之差不超过 1
- $n_0 = n_2 + 1$

5.2.2 二叉树的存储结构

- 顺序存储结构：建议从数组下标 1 开始存储树中的结点

5.3 二叉树的遍历和线索二叉树

5.3.1 二叉树的遍历

- 按后序非递归算法遍历二叉树，访问一个结点 p 时，栈中结点恰好是 p 结点的所有祖先；从栈底到栈顶结点再加上 p 结点，刚好构成从根节点到 p 结点的一条路径

5.3.2 线索二叉树

- 标志域为 0 指示结点的孩子，标志域为 1 指示结点的前驱或后继
- 在后序线索二叉树上找后继时需要知道结点双亲，即采用带标志域的三叉链表作为存储结构

5.4 树、森林

5.4.1 树的存储结构

5.4.2 树、森林与二叉树的转换

- 由于根节点没有兄弟，所以对应的二叉树没有右子树
- 树转二叉树
- 森林转二叉树
    - 每棵树转二叉树
    - 每棵树树根间加一条线
    - 以第一棵树为轴心顺时针旋转 45°

5.4.3 树和森林的遍历

- 树的遍历：
    - 先根遍历
    - 后根遍历

- 森林的遍历：
  - 先序遍历
  - 中序遍历（后序遍历）
    - 中序遍历森林中第一棵树的根节点的子树森林
    - 访问第一棵树的根节点
    - 中序遍历除去第一棵树之后剩余的树构成的森林
- 二叉遍历的对应关系
  - -树---森林-二叉树
  - 先根--先序--先序
  - 后根--中序--中序

5.5 树与二叉树的应用

5.5.1 哈夫曼树和哈夫曼编码

- 初始结点最终都为叶结点，且权值越小的结点到根结点的路径长度越大
- 构造过程中共新建了 $n-1$ 个结点，因此哈夫曼树的结点总数为 $2n-1$
- 哈夫曼树中不存在度为 1 的结点

5.5.2 并查集

- 未作路径优化的并查集时间复杂度为 $O(n)$

第6章 图

- 图的基本概念
- 图的存储及基本操作
  - 邻接矩阵
  - 邻接表
  - 邻接多重表
  - 十字链表
- 图的遍历
  - 深度优先搜索
  - 广度优先搜索
- 图的基本应用
  - 最小（代价）生成树
  - 最短路径
  - 拓扑排序
  - 关键路径

6.1 图的基本概念

6.1.1 图的定义

- 一般当图 G 满足 $|E| < |V|log|V|$ 时，可以将 G 视为稀疏图
- 顶点 v_p 到顶点 v_q 之间的一条路径是指顶点序列 $v_p, v_{i_1}, v_{i_2}, \ldots, v_{i_m}, v_q$
- 路径序列中，顶点不重复出现的路径称为简单路径；除第一个顶点和最后一个顶点外，其余顶点不重复出现的回路称为简单回路

6.2 图的存储及基本操作

6.2.1 邻接矩阵法

- 设图 G 的邻接矩阵为 A， A^n的元素 $A^n[i][j]$ 等于由顶点 i 到顶点 j 的长度为 n 的路径的数目

6.2.2 邻接表法

- 无向图所需存储空间为 $O(|V| + 2|E|)$
- 有向图所需存储空间为 $O(|V| + |E|)$
- 邻接矩阵和邻接表对于不同的操作各有优势

6.2.3 十字链表

- 十字链表是**有向图**的一种链式存储结构

6.2.4 邻接多重表

- 邻接多重表是**无向图**的一种链式存储结构

6.2.5 图的基本操作

6.3 图的遍历

- 对于同样一个图，基于邻接矩阵的遍历所得到的 DFS 序列和 BFS 序列是唯一的，基于邻接表的遍历所得到的 DFS 序列和 BFS 序列是不唯一的

6.3.1 广度优先搜索

- 空间复杂度： $O(|V|)$
- 时间复杂度：
  - 邻接矩阵： $O(|V| + |E|)$
  - 邻接表： $O(|V|^2)$
- 可以求解非带权图的单源最短路径问题

6.3.2 深度优先搜索

- 空间复杂度： $O(|V|)$
- 时间复杂度：
  - 邻接矩阵： $O(|V| + |E|)$
  - 邻接表： $O(|V|^2)$

6.3.3 图的遍历与图的连通性

6.4 图的应用

6.4.1 最小生成树

- $Prim$ 算法的时间复杂度为 $O(|V|^2)$，不依赖于 $|E|$，因此它适用于求解边稠密的图的最小生成树
- $Kruskal$ 算法通常采用堆来存放边的集合，因此每次选择最小权值的边只需 $O(log_2|E|)$ 的时间；此外求解等价类可以采用并查集的数据结构；故构造最小生成树的时间复杂度为 $O(|E|log_2|E|)$，因此它适合于边稀疏而顶点较多的图
- **当带权连通图的任意一个环中所包含的边的权值均不相同时，其 MST 是唯一的**

6.4.2 最短路径

- $Dijkstra$ 算法：
  - 使用邻接矩阵表示时，时间复杂度为 $O(|V|^2)$
  - 使用带权邻接表表示时，虽然修改 `dist[]` 的时间可以减少，但由于在 `dist[]` 中选择最小分量的时间不变，时间复杂度仍为 $O(|V|^2)$
  - 点对点最短路径问题的求解和单源最短路径问题的求解一样复杂，时间复杂度也为 $O(|V|^2)$
  - $Dijkstra$ **算法无法处理含有负边权的最短路问题**
- $Floyd$ 算法
  - 时间复杂度为 $O(|V|^3)$，但常数较小，对中等规模的输入仍有效
  - 算法允许图中有带负权值的边，但**不允许有包含带负权值的边组成的回路**
  - 也可以用单源最短路径算法来解决每对顶点之间的最短路径问题，即轮流将每个顶点作为源点，并且在所有边权值均非负时，运行一次 $Dijkstra$ 算法，其时间复杂度为 $O(|V|^3)$

6.4.3 有向无环图描述表达式

6.4.4 拓扑排序

- 采用邻接表存储时拓扑排序的时间复杂度为 $O(|V|+|E|)$，采用邻接矩阵存储时拓扑排序的时间复杂度为 $O(|V|^2)$
- 由于 AOV 网中各顶点的地位平等，每个顶点编号是人为的，因此可以按拓扑排序的结果重新编号，生成 AOV 网的新的邻接存储矩阵，这种邻接矩阵可以是三角矩阵；但对于一般的图来说，若其邻接矩阵是三角矩阵，则存在拓扑序列，反之则不一定成立

6.4.5 关键路径

- AOE 网中的边有权值；而 AOV 网中的边无权值，仅表示顶点之间的前后关系
- 在 AOE 网中仅有一个入度为 0 的顶点，称为开始顶点（源点），它表示整个工程的开始；网中也存仅存在一个出度为 0 的顶点，称为结束顶点（汇点），它表示整个工程的结束
- 从源点到汇点的所有路径中，具有最大路径长度的路径称为关键路径，而把关键路径上的活动称为关键活动
- 相关参量定义：
  - 事件 v_k 的最早发生时间 $ve(k)$
    - 从源点 v_1 到顶点 v_k 的最长路径长度

        - $ve(\text{源点})=0$
        - $ve(k)=Max\{ve(j)+Weight(v_j,v_k)\}$， v_k 为 v_j 的任意后继
    - 事件 v_k 的最迟发生时间 $vl(k)$
        - $vl(\text{汇点})=ve(\text{汇点})$
        - $ve(k)=Min\{vl(j)-Weight(v_k,v_j)\}$， v_k 为 v_j 的任意前驱
    - 活动 a_i 的最早开始时间 $e(i)$
        - 指该活动弧的起点所表示的事件的最早发生时间
        - 若边 $<v_k,v_j>$ 表示活动 a_i，则有 $e(i)=ve(k)$
    - 活动 a_i 的最迟开始时间 $l(i)$
        - 指该活动弧的起点所表示的事件的最迟发生时间与该活动所需时间之差
        - 若边 $<v_k,v_j>$ 表示活动 a_i，则有 $l(i)=vl(j)-Weight(v_k,v_j)$
    - 一个活动 a_i 的最迟开始时间 $l(i)$ 和其最早开始时间 $e(i)$ 的差额 $d(i)=l(i)-e(i)$
        - 若一个活动的时间余量为 0 则说明该活动必须要如期完成，否则就会拖延整个工程的进展，所以称 $l(i)-e(i)=0$ 即 $l(i)=e(i)$ 的活动 a_i 是关键活动
- 求解关键路径的算法步骤如下：
    - 从源点出发，令 $ve(\text{源点})=0$，按拓扑有序求其余顶点的最早发生时间 $ve()$
    - 从汇点出发，令 $vl(\text{汇点})=ve(\text{汇点})$，按逆拓扑有序求其余顶点的最迟发生时间 $vl()$
    - 根据各顶点的 $ve()$ 值求所有弧的最早开始时间 $e()$
    - 根据各顶点的 $vl()$ 值求所有弧的最迟开始时间 $l()$
    - 求 AOE 网中所有活动的差额 $d()$，找出所有 $d()=0$ 的活动构成关键路径
- 注意：
    - 关键路径上的所有活动都是关键活动，可以通过加快关键活动来缩短整个工程的工期，但也不能任意缩短关键活动，因为一旦缩短到一定的程度，该关键活动就可能会变成非关键活动
    - 网中的关键路径并不唯一，且对于有几条关键路径的网，只提高一条关键路径上的关键活动速度并不能缩短整个工程的工期，只有加快那些包括所有关键路径上的关键活动才能达到缩短工期的目的