大学计算机-Python算法实践

Part One

第一讲 线性数据结构

线性数据结构

- 线性数据结构是计算机组织数据的一种方式。
- 线性结构是一个数据元素集合
  - 有一个唯一的首元素
  - 有一个唯一的尾元素
  - 除了首元素和尾元素，所有的元素都有一个唯一的前驱
  - 除了首元素和尾元素，所有的元素都有一个唯一的后继
- 常见的线性数据结构：数组、栈、队列、链表等

- 数组
  - Python语言没有提供数组数据类型，通常直接使用列表作为数组。
  - 列表支持数据要求的几种核心操作
    - 创建数组
    - 索引访问
    - 索引赋值
    - 迭代遍历

- 栈
  - 栈（Stack）是一种特殊的列表
    - 栈内的元素只能通过列表的一端访问（栈顶）。
    - 栈是后进先出的数据结构
  - 栈的核心操作
    - 入栈（Push）
    - 出栈（Pop）

- 栈的实现：列表
  - 列表从最后的位置添加和移除元素都非常方便高效，可天然地快捷实现栈的操作
    - 列表的append()方法对应于入栈操作(push)
    - 列表的pop()方法对应于出栈操作(pop)

```
s = []
s.append(1)
s.append(2)
print(s)    #输出[1, 2]
s.pop()
print(s)    #输出[1]
s.pop()
print(s)    #输出[]
```

- 队列
  - 队列（Queue）也是一种特殊的列表
    - 队列插入元素的操作只能在队尾进行
    - 队列删除元素的操作只能在队首进行
    - 队列是先进先出的数据结构
  - 队列的核心操作
    - 尾部添加
    - 首部删除

- 队列的实现：deque
  - 队列用列表实现并不适合（在首部添加数据只能使用insert方法需要移动其他数据）
  - collections.deque是Python提供的双端队列，支持从任意一端添加删除数据，速度快
  - deque支持的方法
    - **append(x)** 尾部添加元素
    - appendleft(x) 首部添加元素
    - pop() 尾部弹出元素
    - **popleft()** 首部弹出元素
    - clear() 清空队列
    - reverse() 反转队列

队列的实现

```
from collections import deque
dq = deque()
dq.append(1)
dq.append(2)
print(dq)              #输出deque([1, 2])
dq.popleft()
print(dq)              #输出deque([2])
dq.popleft()
print(dq)              #输出deque([])
```

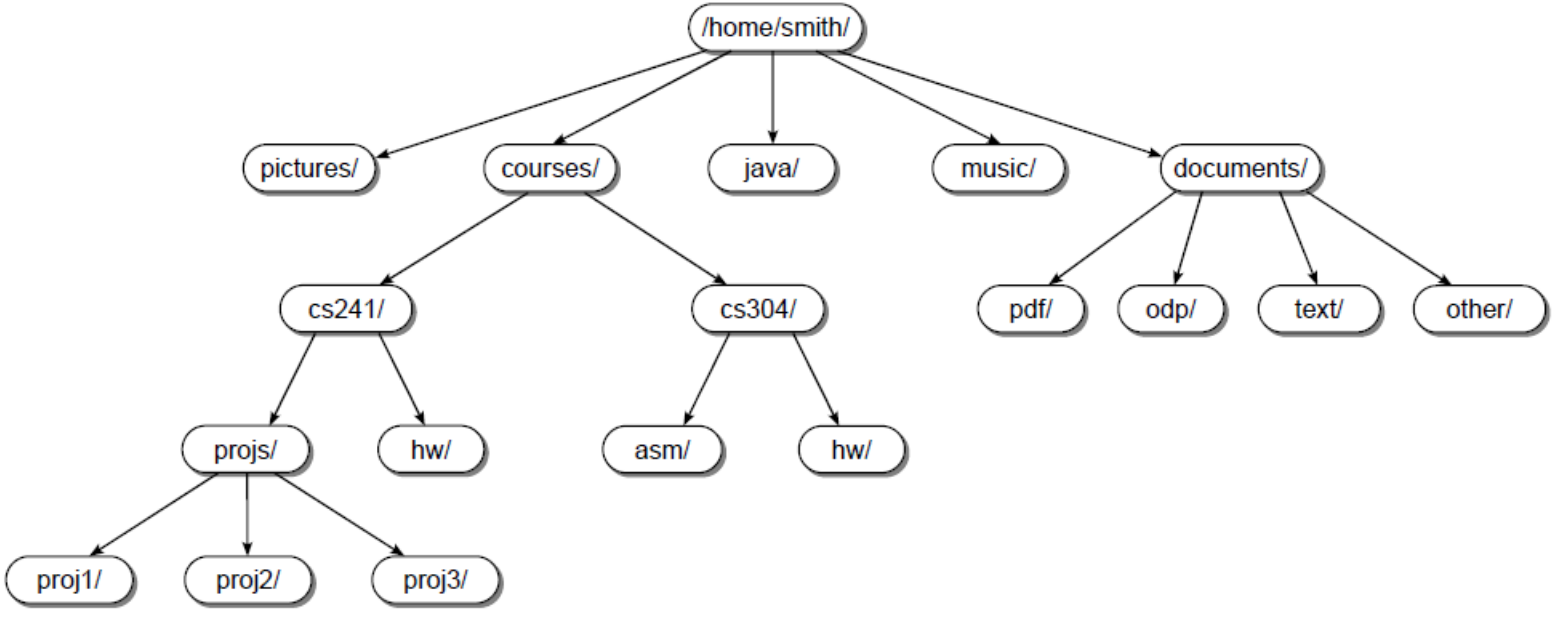
Part Two

第二讲 树的概念

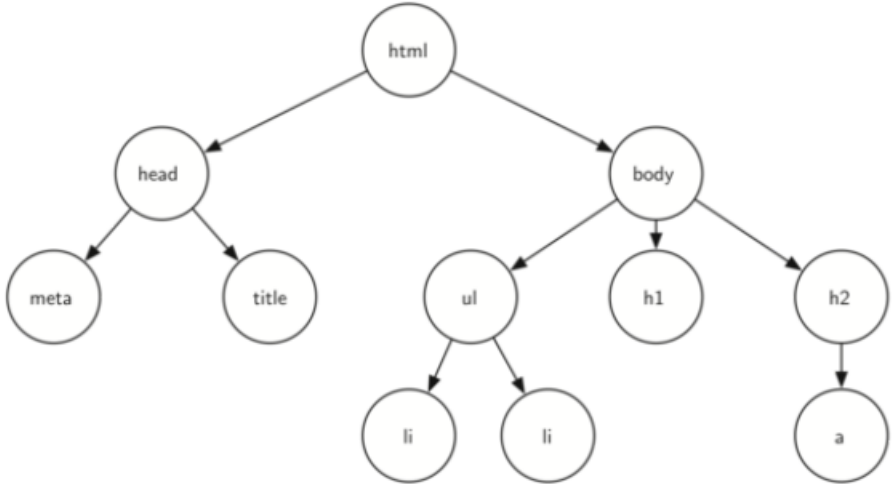
树的概念

- 树：
  - 树不是一种线性结构，是非线性的。
  - 树在计算机科学里应用广泛，包括操作系统，图形学，数据库和计算机网络等。
  - 树和真正的树有许多相似的地方，也包括根、树枝和叶子，它们的不同在于计算机中的树的根在顶层而它的叶子在底部。

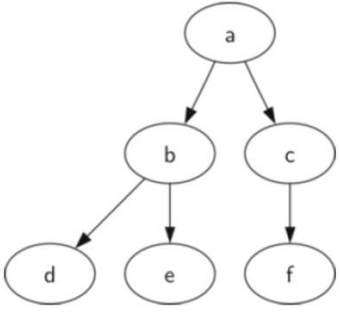
- 树的实例：


/home/smith/
pictures/
courses/
java/
music/
documents/
cs241/
cs304/
pdf/
odp/
text/
other/
projs/
hw/
asm/
hw/
proj1/
proj2/
proj3/

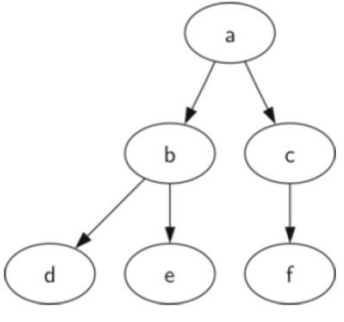
- 树的实例：



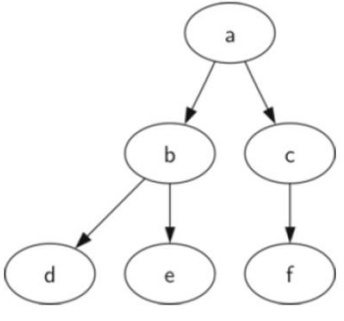
- 树的术语：
  - 节点（Node）
    - 树中的每一个数据元素称为一个节点，节点是树的基本构成部分。
  - 边（Edge）
    - 边也是树的基本构成部分。边有方向，连接两个节点，并表示它们之间的联系。
    - 除了根节点外每个节点都有且只有一条与其他节点相连的入边（指向该节点的边），每个节点可能有许多条出边（从该节点指向其他节点的边）。
  - 根节点（Root）
    - 根节点是树中唯一一个没有入边的节点。



- 树的术语：
  - 路径（Path）
    - 路径是由边连接起来的节点的有序排列。
  - 子节点集（Children）
    - 当一个节点的入边来自另一个节点时，称前者是后者的子节点，同一个节点的所有子节点构成子节点集。
  - 父节点（Parent）
    - 一个节点是它出边所连接的所有节点的父节点。
  - 兄弟节点（Sibling）
    - 同一个节点的所有子节点互为兄弟节点。



- 树的术语：
  - 子树（Subtree）
    - 子树是一个父节点的某个子节点的所有边和后代节点所构成的集合。
  - 叶节点（Leaf Node）
    - 没有子节点的节点成为称为叶节点。
  - 层数（Level）
    - 一个节点的层数是指从根节点到该节点的路径中的边的数目。
  - 高度（Height）
    - 树的高度等于所有节点的层数的最大值。



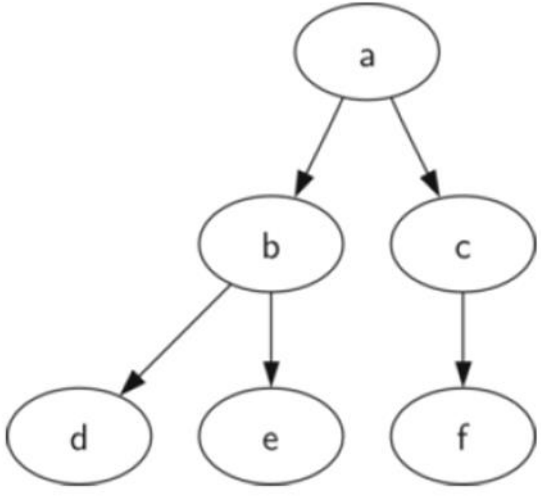
树的概念

- 树的定义：
  - 树是节点和连接节点的边的集合，它有以下特征：
    - 有一个节点被设计为根节点。
    - 除了根节点，每一个节点都通过一条边与它唯一的父节点相连。
    - 可以沿着唯一的路径从根节点到每个节点。
    - 如果这个树的每个节点都至多有两个子节点，称之为二叉树。

Part Three

第三讲 二叉树

- 二叉树的定义
  - 二叉树是由n（n≥0）个结点组成的有限集合、每个结点最多有两个子树的有序树。它或者是空集，或者是由一个根和称为左、右子树的两个不相交的二叉树组成。
  - 结点的度和树的度
    - 每个结点具有的子树个数称为结点的度，树中所有结点的度的最大值称为树的度
  - 二叉树的度为2

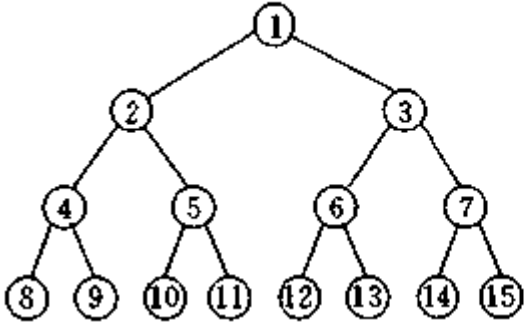


- 二叉树的特点：
  - 二叉树是有序树，即使只有一个子树，也必须区分左、右子树；
  - 二叉树的每个结点的度不能大于2，只能取0、1、2三者之一；
  - 二叉树中所有结点的形态有5种：空结点、无左右子树的结点、只有左子树的结点、只有右子树的结点和具有左右子树的结点；

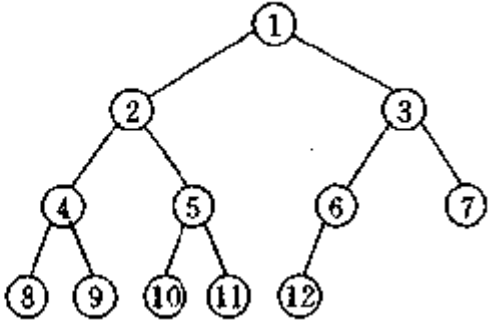
- 二叉树的性质：
  - 二叉树的第i层至多有2^{i-1}个结点；
  - 对任何一棵二叉树T，如果其叶子结点数为N0，度为2的结点数为N2，则N0=N2+1。

N0+N1+N2=N
N2*2+N1=N-1

- 二叉树的性质：
  - 满二叉树：当树中每一层都满时，则称此树为满二叉树。
  - 完全二叉树：在一棵二叉树中，除最后一层外，若其余层都是满的，并且最后一层或者是满的，或者是右边缺少连续若干个结点，则称此树为完全二叉树。
  - 满二叉树是完全二叉树的特例。
  - 深度为h的满二叉树的结点数为2^h-1

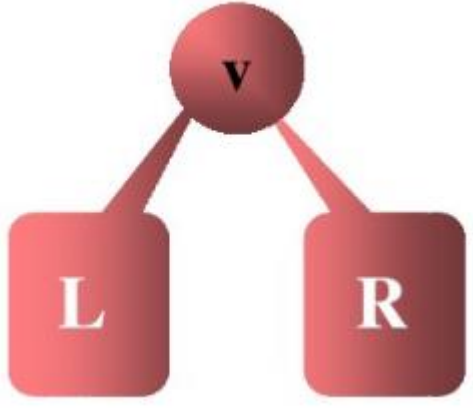


满二叉树



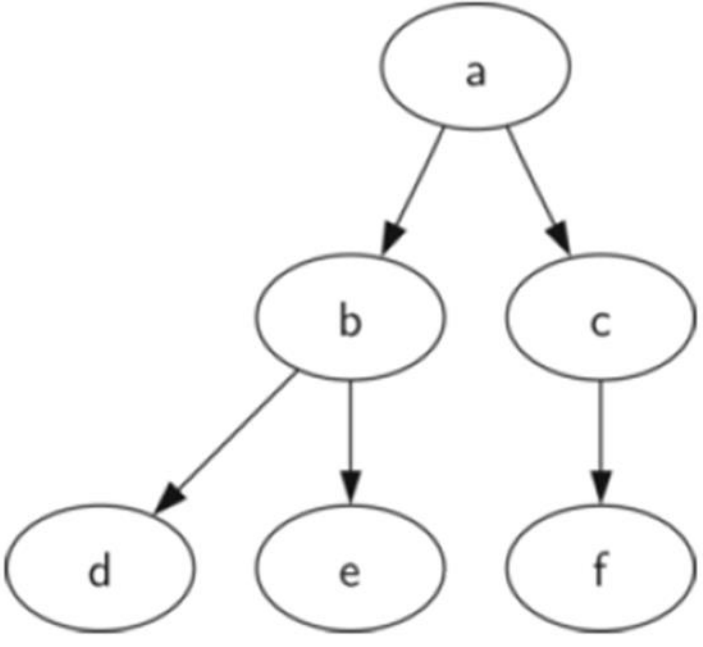
完全二叉树

- 二叉树的遍历：
  - 按照一定次序访问树中所有结点，并且每个结点的值仅被访问一次的过程。
  - 可能的三种遍历次序：
    - 先序遍历：vLR
    - 中序遍历：LvR
    - 后序遍历：LRv



二叉树

- 先序遍历：abdecf
- 中序遍历：dbeafc
- 后序遍历：debfca



大学计算机-Python算法实践