# 5.二叉树

## （b）树的表示

邓俊辉

deng@tsinghua.edu.cn

# 接口

| 节点 | 功能 |
| --- | --- |
| root() | 根节点 |
| parent() | 父节点 |
| firstChild() | 长子 |
| nextSibling() | 兄弟 |
| insert(i, e) | 将e作为第i个孩子插入 |
| remove(i) | 删除第i个孩子（及其后代） |
| traverse() | 遍历 |

# 父节点

❖ 观察：除根外，任一节点有且仅有一个父节点

❖ 构思：将节点组织为序列，各节点分别记录

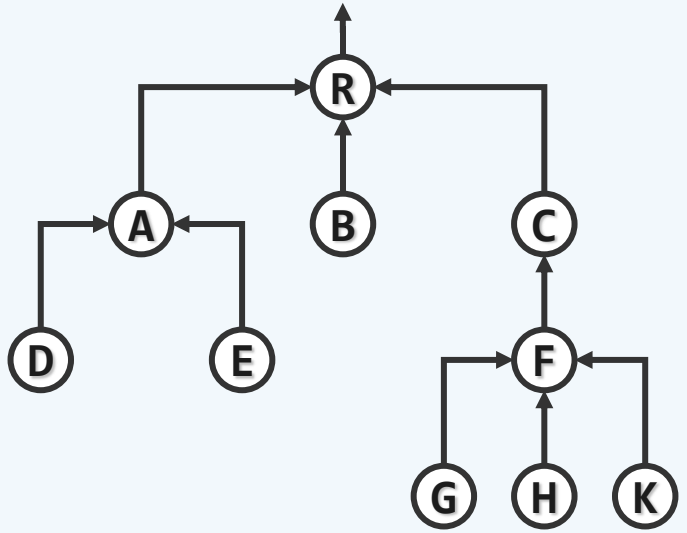
data　本身信息

parent　父节点的秩或位置

❖ 树根（0）也有“虚构的”父节点

parent(0) = -1 或

parent(0) = NULL

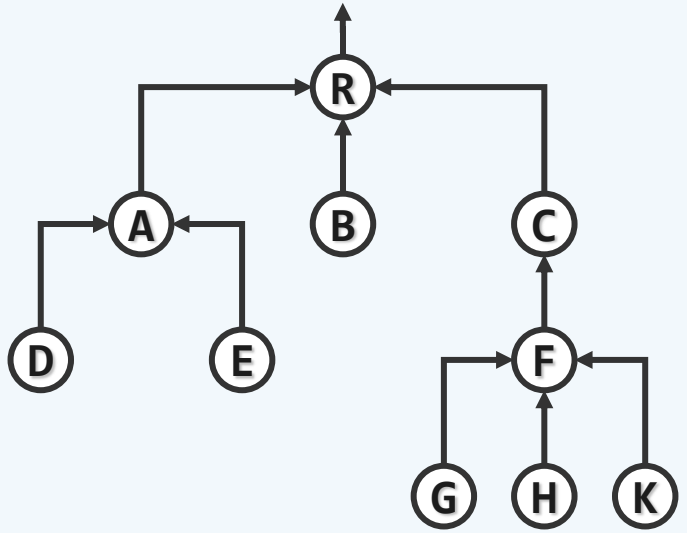



| rank | data | parent |
|---|---|---|
| 0 | R | -1 |
| 1 | A | 0 |
| 2 | B | 0 |
| 3 | C | 0 |
| 4 | D | 1 |
| 5 | E | 1 |
| 6 | F | 3 |
| 7 | G | 6 |
| 8 | H | 6 |
| 9 | K | 6 |

# 父节点

- 空间性能：O(n)
- 时间性能
  - ☺ parent()：O(1)
  - ☹ root()：O(n)或O(1)
  - ☹ firstChild()：O(n)
  - ☹ nextSibling()：O(n)
- 动态操作姑且不论，首先

  如何加速对孩子、兄弟的查找？

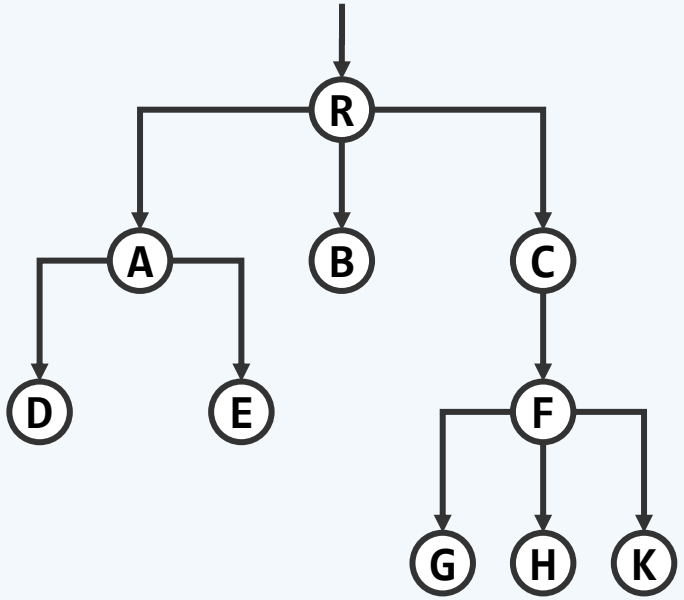



| rank | data | parent |
|---|---|---|
| 0 | R | -1 |
| 1 | A | 0 |
| 2 | B | 0 |
| 3 | C | 0 |
| 4 | D | 1 |
| 5 | E | 1 |
| 6 | F | 3 |
| 7 | G | 6 |
| 8 | H | 6 |
| 9 | K | 6 |

# 孩子节点

- 同一节点的所有孩子，组织为一个序列
- 序列的长度，分别等于对应节点的度数

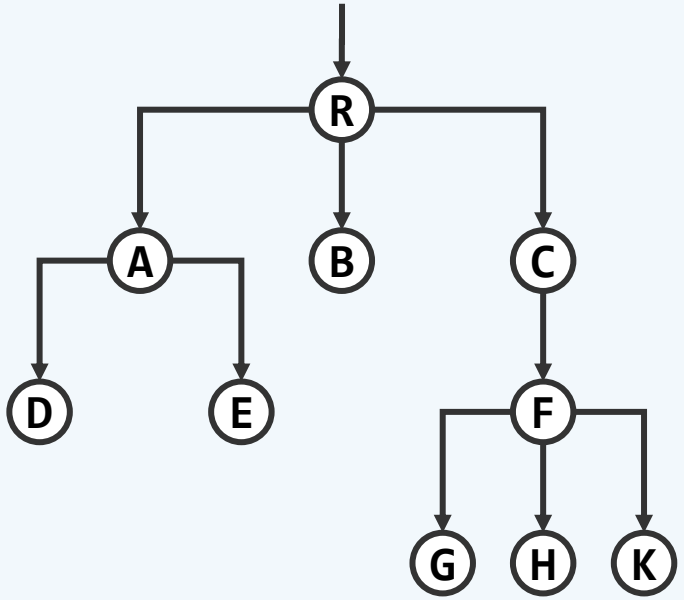
| | data | children |
|---|---|---|
| 0 | A | → 3 → 5 → ^ |
| 1 | B | ^ |
| 2 | C | → 6 → ^ |
| 3 | D | ^ |
| 4 | R | → 0 → 1 → 2 → ^ |
| 5 | E | ^ |
| 6 | F | → 7 → 8 → 9 → ^ |
| 7 | G | ^ |
| 8 | H | ^ |
| 9 | K | ^ |




- 现在，所有孩子都可很快找出，但parent()却很慢...

# 父节点 + 孩子节点

| | data | parent | children |
|---|---|---|---|
| 0 | A | 4 | 3 → 5 → ^ |
| 1 | B | 4 | ^ |
| 2 | C | 4 | 6 → ^ |
| 3 | D | 0 | ^ |
| 4 | R | -1 | 0 → 1 → 2 → ^ |
| 5 | E | 0 | ^ |
| 6 | F | 2 | 7 → 8 → 9 → ^ |
| 7 | G | 6 | ^ |
| 8 | H | 6 | ^ |
| 9 | K | 6 | ^ |



# 长子 + 兄弟

- 每个节点均设两个引用
  - 纵：firstChild()
  - 横：nextSibling()
- 如此
  - 对于度数为d的节点
  - 可在O(d + 1)时间内遍历其所有孩子
- 若再设置parent引用
  - 则parent()接口也仅需O(1)时间