浙江传媒学院
COMMUNICATION UNIVERSITY OF ZHEJIANG

数据结构与算法

Data Structures and Algorithms

谢昊
xiehao@cuz.edu.cn

第二章

半线性结构
Semi-Linear Structures

大纲

1. 二叉树

    二叉树的基本操作

2. 小结

表 1: 线性结构的优势与不足

	顺序列表	**链式列表**
访问元素	$O(1)$	$O(n)$
增删元素	$O(n)$	$O(1)$

表 1: 线性结构的优势与不足

	顺序列表	**链式列表**
访问元素	$O(1)$	$O(n)$
增删元素	$O(n)$	$O(1)$

半线性结构：可去二者之糟粕，取二者之精华

二叉树

二叉树

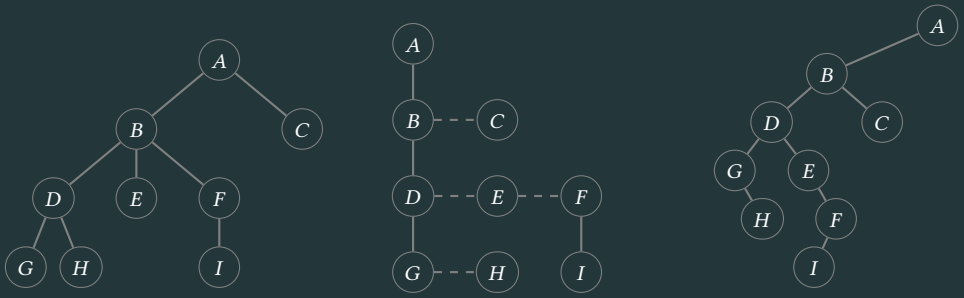
二叉树 (binary tree)

- 度不大于 2 的有序树
- 子结点可按左右区分

将树转化为二叉树

- 令长子为左子结点、首个兄弟为右子结点
- 任何树均可按此法转化为二叉树
- 因二叉树的表示与运算相对方便，故树的问题均可转化为二叉树形式进行研究

二叉树



(a) 树表示 **(b)** 中间表示 **(c)** 二叉树表示

图 1: 将树转化为二叉树

二叉树的基本操作

二叉树的抽象数据类型

```
ADT BinaryTree {
    数据：
        数据对象：D = {a_k | a_k ∈ 数据元素集合, k ∈ ℤ ∩ [0, n)}
        逻辑关系：若 D = ∅，则 R = ∅；否则 R = {⟨a_i, a_j⟩ | i < j; i, j ∈ ℤ ∩ [0, n); a_i, a_j ∈ D_l ∪ {a_0} 或 D_r ∪ {a_0}}[1]
    操作：
        create_binary_tree()
            构造并初始化一个空二叉树t
        insert_left_binary_tree(p, d)
            在二叉树中结点p下插入值为d的左子结点，右子结点可简单类比，略
        remove_left_binary_tree(p)
            在二叉树中删除结点p的左子树，右子树可简单类比，略
        traverse_binary_tree(t)
            以某种方式遍历二叉树t的所有结点
}
```

[1] a_0 为根结点，D_l 与 D_r 分别为其左右子树结点集合，且 $D_l \cap D_r = \varnothing$

二叉树的基本操作

二叉树的初始化

- 利用结点初始化方法
- 注意空指针处理

```
BinaryTreeNode *create_binary_tree_node(DataType d) {
    BinaryTreeNode *n =
            malloc(sizeof(BinaryTreeNode));
    if (n) {
        n->data = d;
        n->left = n->right = n->parent = NULL;
    }
    return n;
}
```

```
BinaryTree *create_binary_tree() {
    BinaryTree *t = malloc(sizeof(BinaryTree));
    if (t) {
        t->head = create_binary_tree_node(0);
    }
    return t;
}
```

二叉树的基本操作

为二叉树中的指定结点插入左子结点[2]

- 将原左子树作为新结点的左子树
- 注意更新双向链接关系的顺序
- 可将操作拆分为两部分
- 类似于双向链表的结点插入

```
BinaryTreeNode *attach_left_binary_tree(
        BinaryTreeNode *p, BinaryTreeNode *n) {
    assert(n && p);
    if (p->left) {
        n->left = p->left, p->left->parent = n;
    }
    p->left = n, n->parent = p;
    return n;
}
```

```
bool insert_left_binary_tree(
        BinaryTreeNode *p, DataType d) {
    if (!p) {
        printf("Wrong insertion place!\n");
        return false;
    }
    BinaryTreeNode *n = create_binary_tree_node(d);
    return n && attach_left_binary_tree(p, n);
}
```

[2]右子结点可直接类比，略，下同

二叉树的基本操作

删除二叉树中的指定结点的左子树

- 注意更新双向链接关系的顺序
- 可将操作拆分为两部分
- 拆除的子树须通过遍历释放

```
BinaryTreeNode *detach_left_binary_tree(
        BinaryTreeNode *p) {
    assert(p && p->left);
    BinaryTreeNode *c = p->left;
    p->left = c->parent = NULL;
    return c;
}
```

```
bool remove_left_binary_tree(BinaryTreeNode *p) {
    if (!p) {
        printf("Wrong removal place!\n");
        return false;
    }
    BinaryTreeNode *c = detach_left_binary_tree(p);
    return cleanup_binary_tree_by_node(c);
}
```

未完待续 . . .

小结

小结

-

问与答