

本节内容

二叉树
存储结构

王道考研/CSKAOYAN.COM

2

知识总览

二叉树的存储结构

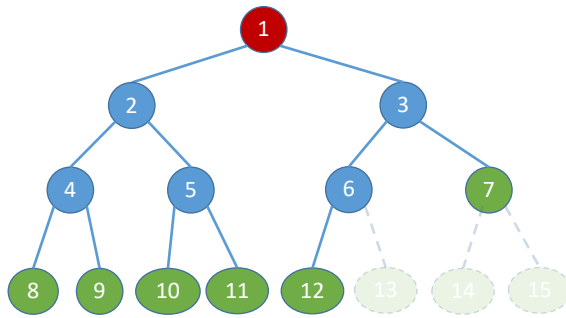
顺序存储

链式存储

王道考研/CSKAOYAN.COM

3

二叉树的顺序存储



```
#define MaxSize 100
struct TreeNode {
    ElemType value; // 结点中的数据元素
    bool isEmpty;    // 结点是否为空
};
```

```
TreeNode t[MaxSize];
```

定义一个长度为 `MaxSize` 的数组 `t`，按照从上至下、从左至右的顺序依次存储**完全二叉树**中的各个结点



`t[0]` `t[1]` `t[2]`

可以让第一个位置空缺，保证数组下标和结点编号一致

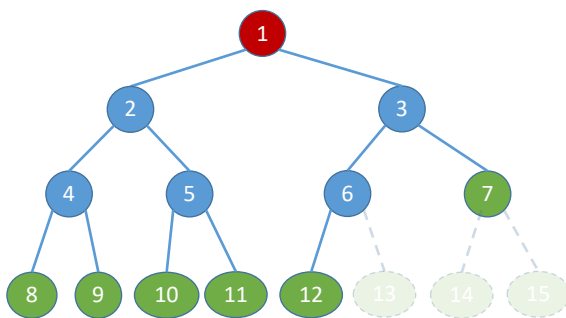
初始化时所有结点标记为空

```
for (int i=0; i<MaxSize; i++){
    t[i].isEmpty=true;
}
```

王道考研/CSKAOYAN.COM

4

二叉树的顺序存储



几个重要常考的基本操作：

- i 的左孩子 $---2i$
- i 的右孩子 $---2i+1$
- i 的父节点 $---\lfloor i/2 \rfloor$
- i 所在的层次 $---\lceil \log_2(n+1) \rceil$ 或 $\lfloor \log_2 n \rfloor + 1$

若**完全二叉树**中共有 n 个结点，则

- 判断 i 是否有左孩子？ $---2i \leq n$?
- 判断 i 是否有右孩子？ $---2i+1 \leq n$?
- 判断 i 是否是叶子/分支结点？ $---i > \lfloor n/2 \rfloor$?



`t[0]` `t[1]` `t[2]`

王道考研/CSKAOYAN.COM

5

二叉树的顺序存储

如果不是完全二叉树，
依然按层序将各节点
顺序存储，那么...

无法从结点编号反映
出结点间的逻辑关系

- i 的左孩子 $2i$
- i 的右孩子 $2i+1$
- i 的父节点 $\lfloor i/2 \rfloor$

^	1	2	3	4	5	6	7	8	^	^	^	^	^	^	^	
t[0]	t[1]	t[2]													

王道考研/CSKAOYAN.COM

6

二叉树的顺序存储

二叉树的顺序存储中，一定要把二叉
树的结点编号与完全二叉树对应起来

- i 的左孩子 $2i$
- i 的右孩子 $2i+1$
- i 的父节点 $\lfloor i/2 \rfloor$

若非完全二叉树中共有n个结点，则

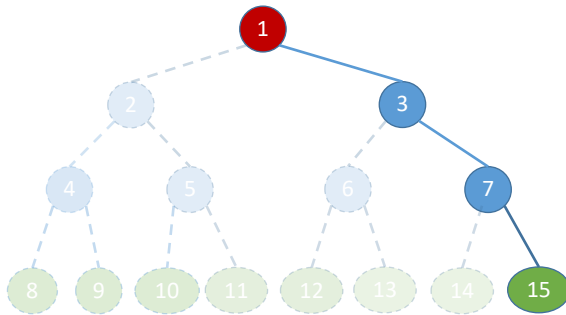
- 判断i是否有左孩子? $2i \leq n?$
- 判断i是否有右孩子? $2i+1 \leq n?$
- 判断i是否是叶子/分支结点? $i > \lfloor n/2 \rfloor?$

^	1	2	3	^	5	6	7	^	^	^	11	12	^	^	^	^
t[0]	t[1]	t[2]													

王道考研/CSKAOYAN.COM

7

二叉树的顺序存储

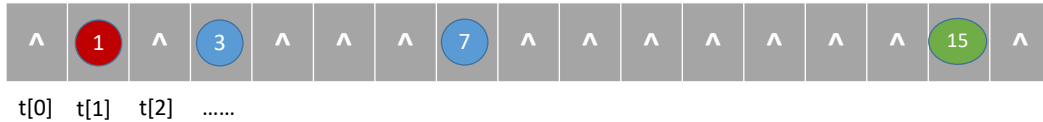


★ 二叉树的顺序存储中，一定要把二叉树的结点编号与完全二叉树对应起来

- i 的左孩子 —— $2i$
- i 的右孩子 —— $2i+1$
- i 的父节点 —— $\lfloor i/2 \rfloor$

最坏情况：高度为 h 且只有 h 个结点的单支树（所有结点只有右孩子），也至少需要 2^h-1 个存储单元

结论：二叉树的顺序存储结构，只适合存储完全二叉树



王道考研/CSKAOYAN.COM

8

二叉树的链式存储

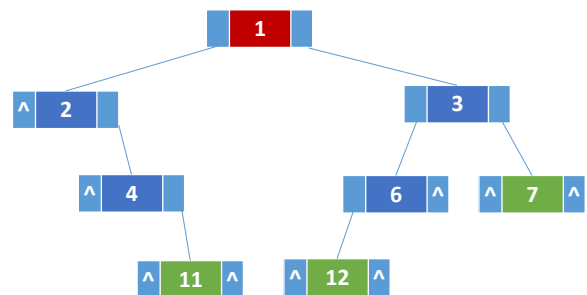
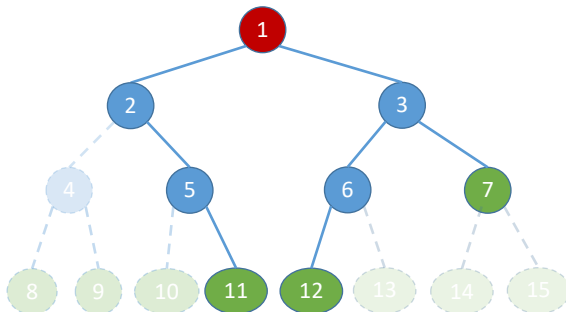
//二叉树的结点（链式存储）

```
typedef struct BiTNode{
    ElemType data;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

//数据域
//左、右孩子指针

可以用于构造
线索二叉树

★ n 个结点的二叉链表共有 $n+1$ 个空链域



王道考研/CSKAOYAN.COM

9

二叉树的链式存储

```
struct ElemType{
    int value;
};

typedef struct BiTNode{
    ElemType data;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

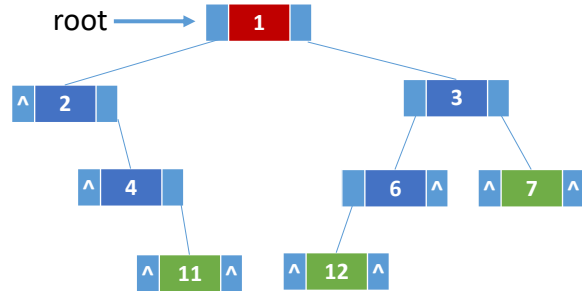
```
//定义一棵空树
BiTree root = NULL;
```

//插入根节点

```
root = (BiTree) malloc(sizeof(BiTNode));
root->data = {1};
root->lchild = NULL;
root->rchild = NULL;
```

//插入新结点

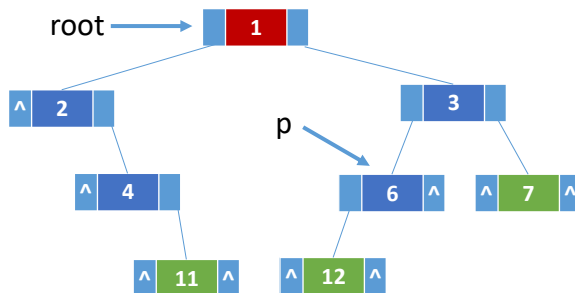
```
BiTNode * p = (BiTNode *) malloc(sizeof(BiTNode));
p->data = {2};
p->lchild = NULL;
p->rchild = NULL;
root->lchild = p; //作为根节点的左孩子
```



王道考研/CSKAOYAN.COM

10

二叉树的链式存储



找到指定结点 p 的左/右孩子——超简单

如何找到指定结点 p 的父结点?

只能从根开始遍历寻找

Tips: 根据实际需求决定要不要加父结点指针

//二叉树的结点 (链式存储)

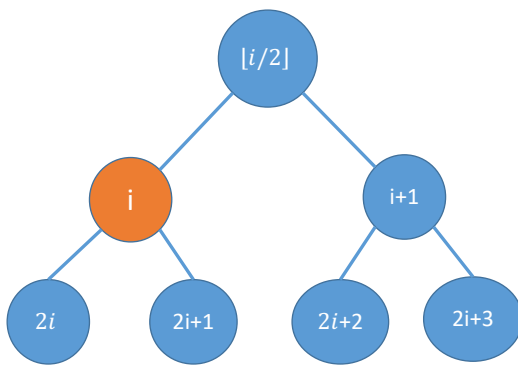
```
typedef struct BiTNode{
    ElemType data; //数据域
    struct BiTNode *lchild,*rchild; //左、右孩子指针
    struct BiTNode *parent; //父节点指针
}BiTNode,*BiTree;
```

三叉链表——方便找父结点

王道考研/CSKAOYAN.COM

11

知识回顾与重要考点



思考：如果结点从0开始编号，该怎么算

★ 二叉树的顺序存储中，一定要把二叉树的结点编号与完全二叉树对应起来

- i 的左孩子 —— $2i$
- i 的右孩子 —— $2i+1$
- i 的父节点 —— $[i/2]$

最坏情况：高度为 h 且只有 h 个结点的单支树（所有结点只有右孩子），也至少需要 2^h-1 个存储单元

结论：二叉树的顺序存储结构，只适合存储完全二叉树

王道考研/CSKAOYAN.COM

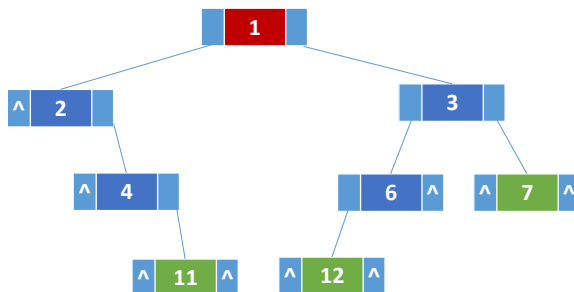
12

知识回顾与重要考点

//二叉树的结点（链式存储）

```
typedef struct BiTNode{
    ElemType data;           //数据域
    struct BiTNode *lchild, *rchild; //左、右孩子指针
}BiTNode, *BiTree;
```

★ n 个结点的二叉链表共有 $n+1$ 个空链域



王道考研/CSKAOYAN.COM

13