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. 基本术语

2. 树

3. 二叉树

    几种特殊的二叉树

    二叉树的性质

    二叉树的存储结构

4. 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

基本术语

基本术语

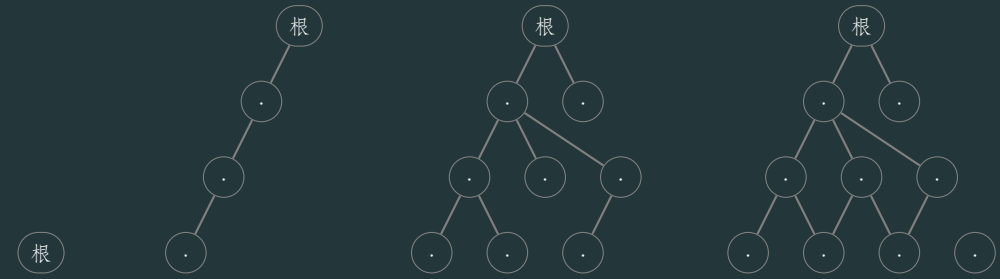
树 (tree) 与森林 (forest)

- **半线性 (semi-linear)** 结构一般指**树**
- 树由 n 个[1]**顶点 (vertex)**[2]与连接于其间的若干条**边 (edge)** 组成
- 空树既无结点亦无边
- 非空树应满足如下条件
  - 有且仅有 1 个特定结点为**根 (root)** 结点
  - 除根结点外的其余结点被分为 d 个[3]互不相交 的**子树 (subtree)**
  - 子树与根之间由边相连，但不形成**环 (ring)**
- 子树亦为树，满足上述性质（递归定义）
- m 棵[4]互不相交的树的集合为**森林**

---

[1] $n \in \mathbb{Z} \cap [0, +\infty)$
[2] 又名**结点 (node)**
[3] $d \in \mathbb{Z} \cap [0, +\infty)$
[4] $m \in \mathbb{Z} \cap [0, +\infty)$

基本术语



(a) 根树 **(b)** 左斜树 **(c)** 普通树 **(d)** 非树

图 1: 几种树与非树

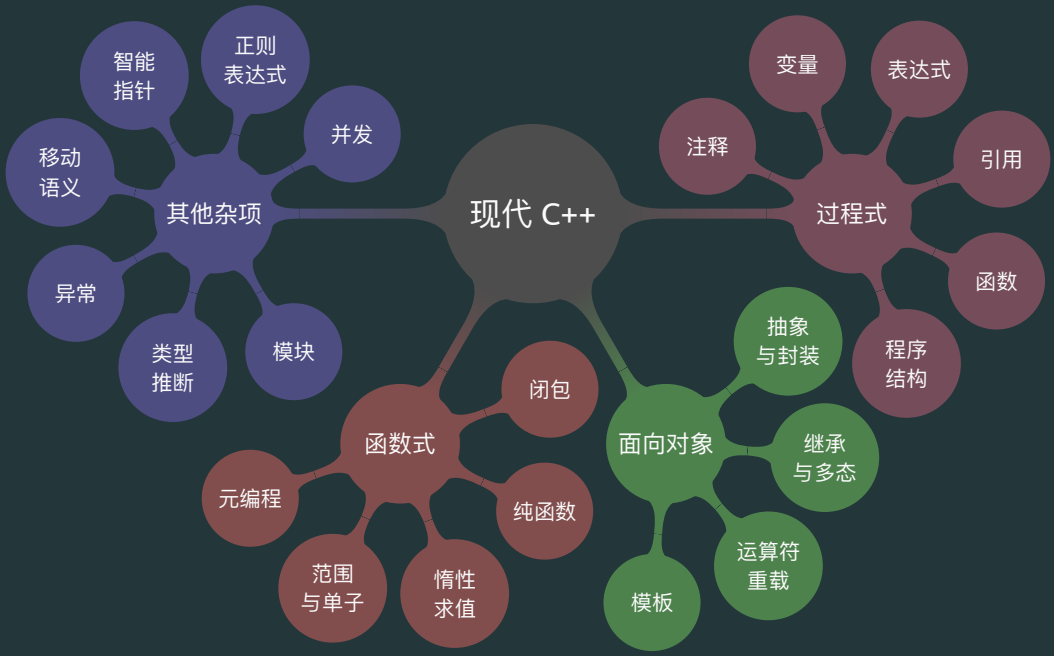

现代 C++
其他杂项
智能指针
正则表达式
并发
移动语义
异常
类型推断
模块
过程式
变量
表达式
注释
引用
函数
程序结构
面向对象
抽象与封装
继承与多态
运算符重载
模板
函数式
闭包
元编程
纯函数
范围与单子
惰性求值

图 2: 思维导图亦为树

树的特点

- 根结点无前驱，其余结点有且仅有 1 个直接前驱
- 所有结点均可有 n 个[5] 直接后继
- 前驱类似线性，后继则不同，故称半线性

[5] $n \in \mathbb{Z} \cap [0, +\infty)$

度 (degree)

- 结点的**度**指其子树个数
- 树的**度**指其最大结点度
- **叶 (leaf)** 结点度为 0，亦称**终端**结点
- 其余结点为**分支**结点

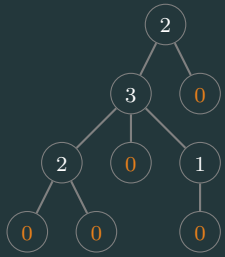


图 3: 结点的度

结点亲缘关系

- 结点的子树根为该结点的**子 (child)** 结点

$$b = child(a), \quad c = child(a)$$

- 该结点为子树根的**父 (parent)** 结点

$$a = parent(b), \quad a = parent(c)$$

- 同一结点的所有子结点互为**兄弟 (sibling)**

$$b = sibling(c), \quad c = sibling(b)$$

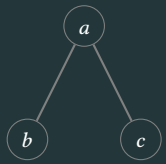


图 4: 结点间的关系[6]

[6] a 为 b 或 c 的父结点，b 或 c 为 a 的子结点，b 与 c 互为兄弟

基本术语

路径 (path)、深度 (depth) 与高度 (height)

- 若结点序列 $\{n_i\}_{i=0}^{k}$ 满足：

$$n_i = parent(n_{i+1}), \quad i = 0, 1, \cdots, k-1$$

则称之为自 n_0 至 n_k 的一条**路径**

  - 所过边数为路径**长度 (length)**
  - 若存在自 n_a 至 n_b 的路径，则该路径唯一，且 n_a 为 n_b 的**祖先 (ancestor)**，n_b 为 n_a 的**子孙 (descendant)**

- 结点**深度**[7]为根至其的路径长度
- 结点**高度**为其最大子孙深度[8]
  - 树的高度为其根的高度

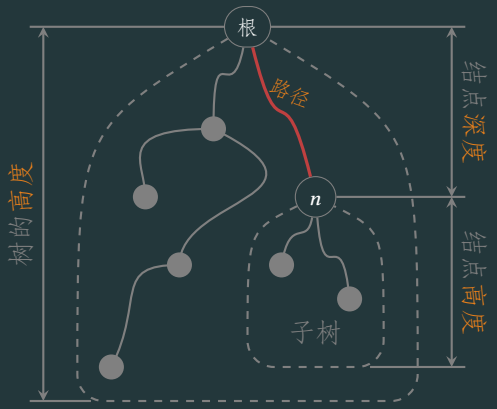


图 5: 路径、高度与深度

[7]又称结点所在**层数 (layer)**，根在第 0 层

[8]此处范围仅限于以其为根的子树内，一般为该子树最大叶深

基本术语

与线性结构的比较

线性	半线性
首元素无前驱	根结点无父结点
尾元素无后继	叶结点无子结点
其他元素单前驱单后继	其他结点单父结点多子结点

树

树的存储结构

- 可采用顺序或链式存储结构
- 每结点须记录：数据信息、与其他结点的逻辑关系

树

树的存储结构

- 可采用顺序或链式存储结构
- 每结点须记录：数据信息、与其他结点的逻辑关系

树的结点关系表示方法

- 父结点表示法：只记录父结点信息
- 子结点表示法：只记录子结点信息
- 父子结点表示法：同时记录父子结点信息
- 长子兄弟表示法：同时记录第一个子结点与兄弟结点信息

父结点表示法

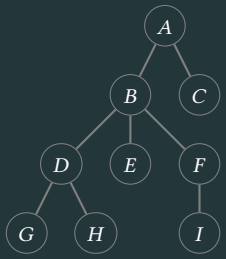
- 采用数组按层存储各结点
- 每结点包括数据信息与父结点序号

图 6: 父结点表示法

复杂度

- 空间：$O(n)$
- 时间：
  - 查找父结点 $O(1)$
  - 但查找子结点$O(n)$

```
typedef struct {
    DataType data; // 数据信息
    int parent; // 父结点序号
} TreeNode;
```



(a) 逻辑表示

0	A	-1
1	B	0
2	C	0
3	D	1
4	E	1
5	F	1
6	G	3
7	H	3
8	I	5

(b) 存储表示

图 7: 父结点表示法

子结点表示法

- 采用数组按层存储各结点
- 每结点包括数据信息与子结点序号链表

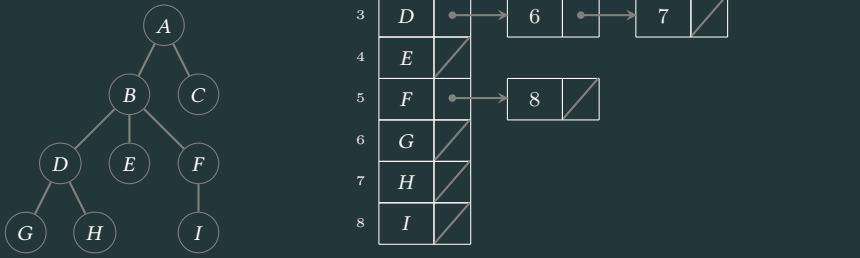
图 8: 子结点表示法

复杂度

- 空间：$O(n)$
- 时间：
  - 查找子结点 $O(d)$[9]
  - 但查找父结点$O(n)$

```c
typedef struct {
    DataType data; // 数据信息
    LinkedList children; // 子结点序号链表
} TreeNode;
```

[9]若该结点度数为 d



(a) 逻辑表示　　**(b)** 存储表示

图 9: 子结点表示法

父子结点表示法

- 结合上述二者

复杂度

- 空间：$O(n)$
- 时间：
  - 查找子结点 $O(d)$
  - 但查找父结点 $O(1)$

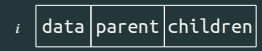
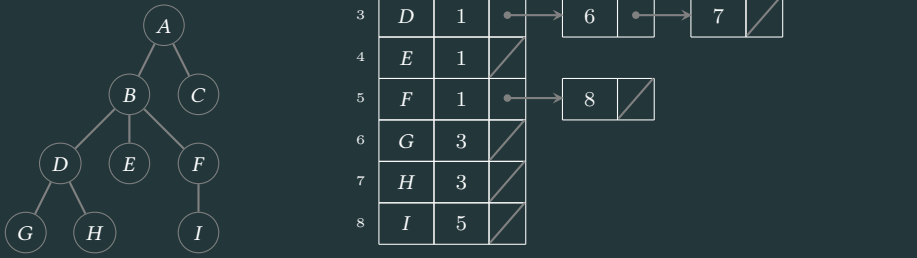


图 10: 父子结点表示法

```c
typedef struct {
    DataType data; // 数据信息
    int parent; // 父结点序号
    LinkedList children; // 子结点序号链表
} TreeNode;
```



(a) 逻辑表示　　**(b)** 存储表示

图 11: 父子结点表示法

父子结点表示法的性质

- **优势**：一定程度上兼顾了查找效率
- **不足**：插入/删除结点操作需大量修改链表，效率偏低

父子结点表示法的性质

- **优势**：一定程度上兼顾了查找效率
- **不足**：插入/删除结点操作需大量修改链表，效率偏低

基本术语

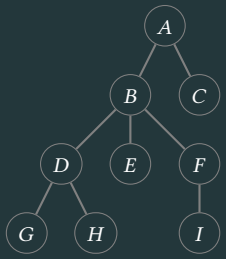
- 若同一结点的所有子结点间具备某种线性次序，则称之为**有序树 (ordered tree)**
- 有序树的任意非叶结点均有且仅有 1 个**长子 (eldest son)**

长子兄弟表示法

- 采用数组按层存储各结点
- 每结点包括
  - 数据信息
  - 长子结点序号
  - 首个兄弟结点序号

图 12: 长子兄弟表示法

```c
typedef struct {
    DataType data; // 数据信息
    int eldest_son; // 长子结点序号
    int sibling; // 兄弟结点序号
} TreeNode;
```



(a) 逻辑表示

0	A	1	-1
1	B	3	2
2	C	-1	-1
3	D	6	4
4	E	-1	5
5	F	8	-1
6	G	-1	7
7	H	-1	-1
8	I	-1	-1

(b) 存储表示

图 13: 长子兄弟表示法

二叉树

二叉树

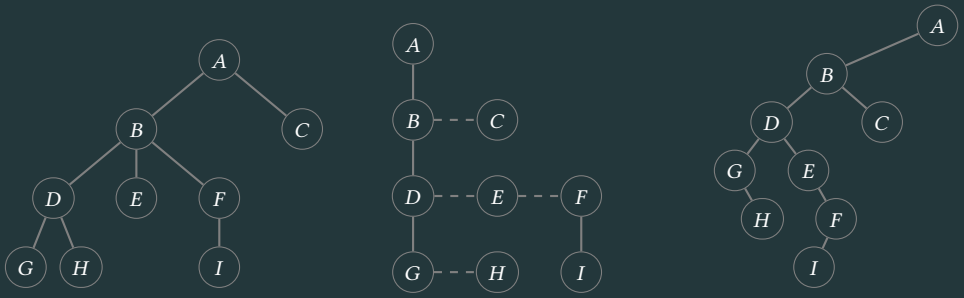
二叉树 (binary tree)

- 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



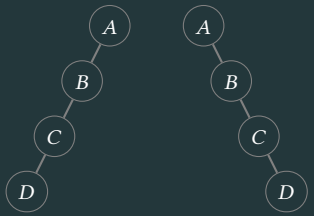
(a) 树表示 **(b)** 中间表示 **(c)** 二叉树表示

图 14: 将树转化为二叉树

几种特殊的二叉树

左斜树[10]

- 所有非叶结点均有且仅有 1 个左子树
- 每层均有且仅有 1 个结点
- 已退化为线性结构



(a) 左斜树　　**(b)** 右斜树

图 15: 斜树

[10]右斜树与之类似，只需将左改作右

几种特殊的二叉树

满二叉树 (full binary tree)

- 所有叶结点均在最后一层
- 所有非叶结点度均为 2

性质

- 同深度的二叉树中满二叉树结点最多
- 同深度的二叉树中满二叉树叶结点最多

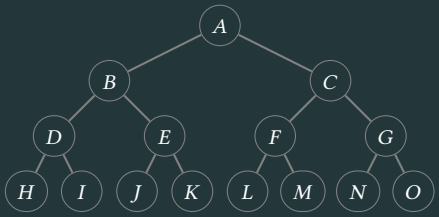
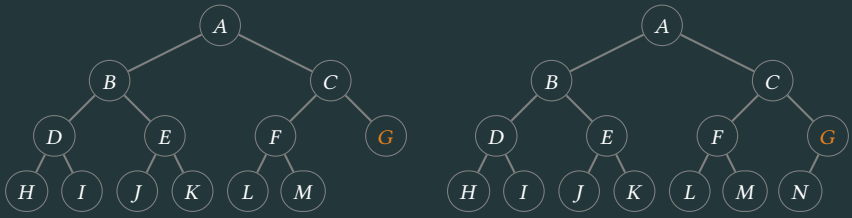


图 16: 满二叉树

几种特殊的二叉树



(a) 叶结点不在同层　　**(b)** 个别非叶结点度不为 2

图 17: 非满二叉树

几种特殊的二叉树

完全二叉树 (proper binary tree)

- 若去除最后一层结点，则为满二叉树
- 最后一层结点自左至右连续[11]排列

性质

- 同结点数的二叉树中完全二叉树最矮
- 满二叉树亦为完全二叉树的一种

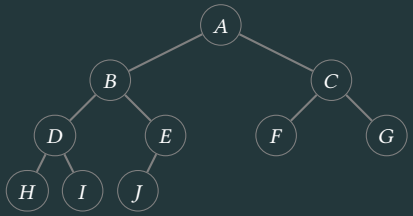


图 18: 完全二叉树

[11]指中间无空结点

二叉树的性质

性质甲

- 令 N 层二叉树第 k 层结点数为 $n_l(k)$，则有

$$1 \leq n_l(k) \leq 2^k,\ k \in \mathbb{Z} \cap [0, N)$$

二叉树的性质

性质甲

- 令 N 层二叉树第 k 层结点数为 $n_l(k)$，则有

$$1 \leq n_l(k) \leq 2^k,\ k \in \mathbb{Z} \cap [0, N)$$

证明

1. 左侧不等式显然成立
2. 右侧不等式当 $k = 0$ 时，第 0 层仅有根结点，故 $n_l(0) = 1 = 2^0$ 显然成立
3. 假设当 $k = n < N - 1$ 时成立，即 $n_l(n) \leq 2^n$，因所有结点的度均不大于 2，故

$$n_l(n+1) \leq 2 \cdot n_l(n) \leq 2 \cdot 2^n = 2^{n+1}$$

于是当 $k = n + 1$ 时，归纳假设成立 □

二叉树的性质

性质乙

- 令 k 层二叉树的结点总数为 $n(k)$，则有

$$k \leq n(k) \leq 2^k - 1$$

性质乙

- 令 k 层二叉树的结点总数为 $n(k)$，则有

$$k \leq n(k) \leq 2^k - 1$$

证明

1. 由性质甲，$1 \leq n_l(i) \leq 2^i,\ i \in \mathbb{Z} \cap [0, k)$
2. 经累加后，有

$$k = \sum_{i=0}^{k-1} 1 \leq n(k) = \sum_{i=0}^{k-1} n_l(i) \leq \sum_{i=0}^{k-1} 2^i = 2^k - 1$$

□

二叉树的性质

思考

- 满足 $n(k) = k$ 的二叉树一定是斜树么?
- 满足 $n(k) = 2^k - 1$ 的二叉树一定是满二叉树么?

二叉树的性质

性质丙

- 令二叉树中度为 d 的结点数为 n_d，$(d \in \{0, 1, 2\})$，则有

$$n_0 = n_2 + 1$$

二叉树的性质

性质丙

- 令二叉树中度为 d 的结点数为 n_d，$(d \in \{0,1,2\})$，则有

$$n_0 = n_2 + 1$$

证明

1. 一方面，结点总数由各种度的结点构成，故总结点数 n 满足

$$n = \sum_{d=0}^{2} n_d = n_0 + n_1 + n_2$$

2. 另一方面，每个非根结点均由 1 个结点生成，故度为 d 的结点可生成 d 个结点，即

$$n = \sum_{d=0}^{2} d \cdot n_d + 1 = n_1 + 2n_2 + 1$$

3. 综上得证 □

二叉树的性质

思考

- 有 n 个结点的完全二叉树有多少叶结点?

二叉树的性质

思考

- 有 n 个结点的完全二叉树有多少叶结点?

提示

- 在完全二叉树中，度为 1 的结点数不多于 1
- 当 n 为偶数时，$n_1 = 1$，$n_0 = n/2$，$n_2 = n/2 - 1$
- 当 n 为奇数时，$n_1 = 0$，$n_0 = (n+1)/2$，$n_2 = (n-1)/2$
- 故 $n_0 = \lceil n/2 \rceil$，$n_2 = \lfloor (n-1)/2 \rfloor$

二叉树的性质

性质丁

- 具有 n 个结点的完全二叉树层数 $k = \lfloor \log_2 n \rfloor + 1$

性质丁

- 具有 n 个结点的完全二叉树层数 $k = \lfloor \log_2 n \rfloor + 1$

证明

1. 由完全二叉树性质与性质乙可得，k 层完全二叉树结点数 n 满足

$$2^{k-1} - 1 < n \leq 2^k - 1 \text{ 或 } 2^{k-1} \leq n < 2^k$$

2. 取对数

$$k - 1 \leq \log_2 n < k \text{ 或 } \log_2 n < k \leq \log_2 n + 1$$

3. 注意到 $k \in \mathbb{Z}$，于是得证 □

二叉树的性质

完全二叉树按层序编号

- 可为完全二叉树结点按层序依次编号
- 约定根编号为 1，依次递增
- 称如此编号为 k 的结点为**结点** k

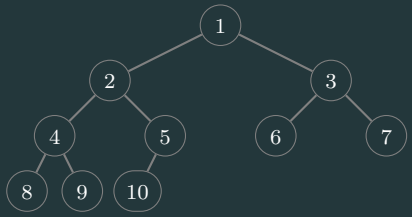


图 19: 为完全二叉树按层序编号

二叉树的性质

性质戊

- 为含有 n 个结点的完全二叉树按层序对结点编号[1][2]，则有
  1. 结点 k 的左右子结点序号分别为 $2k$ 与 $2k+1,\ k \in \mathbb{Z} \cap [1, \lfloor n/2 \rfloor]$
  2. 结点 k 的父结点序号为 $\lfloor k/2 \rfloor,\ k \in \mathbb{Z} \cap (1, n]$

[1][2]根结点为 1

二叉树的性质

性质戊

- 为含有 n 个结点的完全二叉树按层序对结点编号[12]，则有
  1. 结点 k 的左右子结点序号分别为 $2k$ 与 $2k+1,\ k \in \mathbb{Z} \cap [1, \lfloor n/2 \rfloor]$
  2. 结点 k 的父结点序号为 $\lfloor k/2 \rfloor,\ k \in \mathbb{Z} \cap (1, n]$

证明

1. 考察结论 1，当 $k = 1$ 时，显然其左右子结点序号分别为 2 与 3，成立
2. 假设当 $k = m$ 时成立，即结点 m 的左右子结点序号分别为 $2m$ 与 $2m+1$，
   - 因结点 $m+1$ 的左子结点必为结点 m 的右子结点的后继
   - 故结点 $m+1$ 的左子结点序号为 $(2m+1)+1 = 2(m+1)$
   - 且结点 $m+1$ 的右子结点序号为 $2(m+1)+1$

   则当 $k = m+1$ 时，假设成立，故结论 1 成立
3. 由结论 1 知结论 2 成立 □

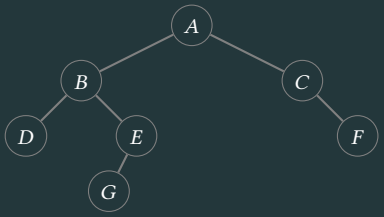
[12]根结点为 1

二叉树的存储结构

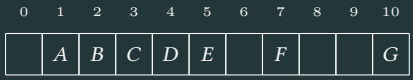
顺序存储

- 按完全二叉树层序编号方式为二叉树编号，跳过不存在结点的编号
- 以静态数组方式存储，留空不存在的结点

二叉树的存储结构



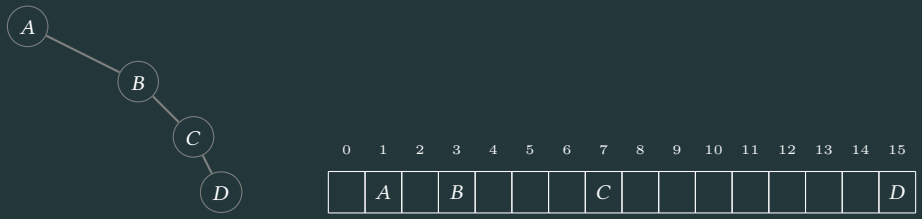
(a) 逻辑结构



(b) 物理结构

图 20: 二叉树的顺序存储示例

二叉树的存储结构



(a) 逻辑结构　　**(b)** 物理结构

图 21: 右斜树的顺序存储示例

二叉树的存储结构

顺序存储的特点

- 可利用性质戊快速访问各结点：$O(1)$
- 增删结点可能需要大幅调整存储
- 在存储含有稀疏结点的二叉树时需耗费大量存储空间
- 仅适合存储含有稠密结点的完全二叉树

二叉树的存储结构

链式存储

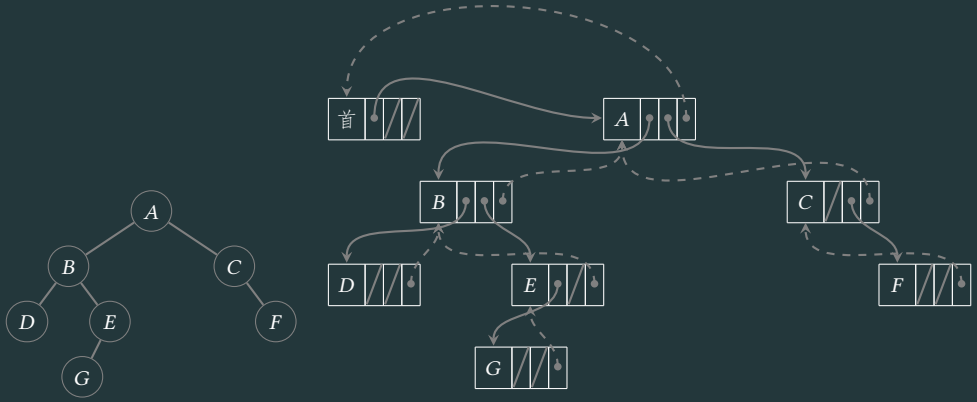
- 采用二/三叉链表表示结点
- 结点除存信息包括
  - 数据信息
  - 左、右子结点地址
  - 可选的父结点地址
- 为后续处理方便，设置虚拟首结点

图 22: 二叉树结点的链式存储结构

```c
typedef struct BinaryTreeNode {
    DataType data; // 数据信息
    struct BinaryTreeNode *left; // 左子结点地址
    struct BinaryTreeNode *right; // 右子结点地址
    struct BinaryTreeNode *parent; // 可选的父结点地址
} BinaryTreeNode;
```

```c
typedef struct {
    BinaryTreeNode *head; // 虚拟首结点
} BinaryTree;
```

二叉树的存储结构



(a) 逻辑结构　　**(b)** 物理结构

图 23: 二叉树的链式存储示例

未完待续 . . .

小结

小结

-

问与答