

## 二叉树

### 先序遍历：观察

真君曰：“昔吕洞宾居庐山而成仙，鬼谷子居云梦而得道，今或无此吉地么？”璞曰：“有，但当遍历耳。”

一桩事情的真相与奥妙，通常并不藏在最深的地方，有时就在表面。只不过，一般人视若无睹。要想成为一个好的算命先生，首先就必须学会观察...

警幻冷笑道：“贵省女子固多，不过择其紧要者录之。下边二厨则又次之。余者庸常之辈，则无册可录矣。”宝玉听说，再看下首二厨上，果然写着“金陵十二钗副册”，又一个写着“金陵十二钗又副册”。

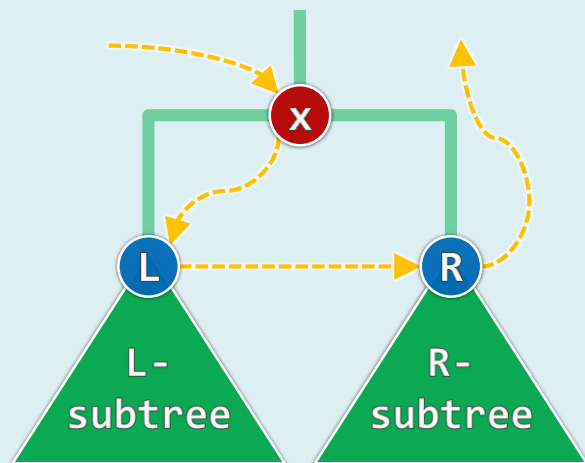
邓俊辉

deng@tsinghua.edu.cn

遍历：按照**某种次序**访问树中各节点，每个节点被访问**恰好一次**

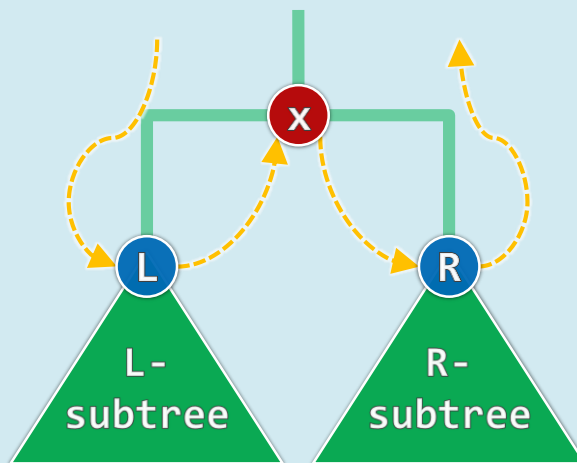
$$\diamond \boxed{T} = \boxed{L} \cup \boxed{x} \cup \boxed{R}$$

先序  
preorder



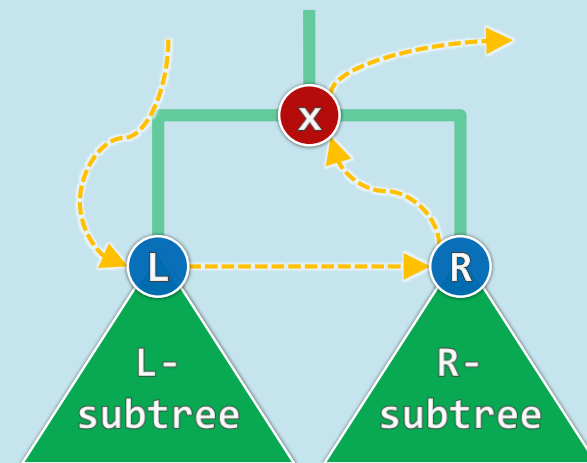
$\boxed{x} \mid \boxed{L} \mid \boxed{R}$

中序  
inorder



$\boxed{L} \mid \boxed{x} \mid \boxed{R}$

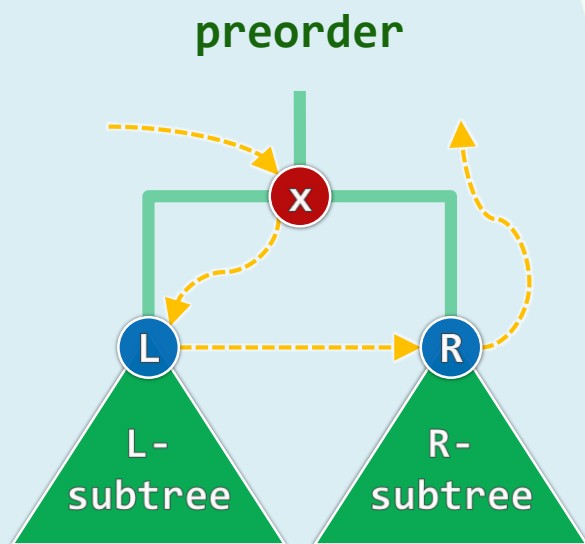
后序  
postorder



$\boxed{L} \mid \boxed{R} \mid \boxed{x}$

❖ 遍历：结果 ~ 过程 ~ 次序 ~ 策略

## 先序实例：统计规模



```
template <typename T>
```

```
Rank BinNode<T>::size() { //后代总数
```

```
Rank s = 1; //计入本身
```

```
if (lc) s += lc->size(); //递归计入左子树规模
```

```
if (rc) s += rc->size(); //递归计入右子树规模
```

```
return s;
```

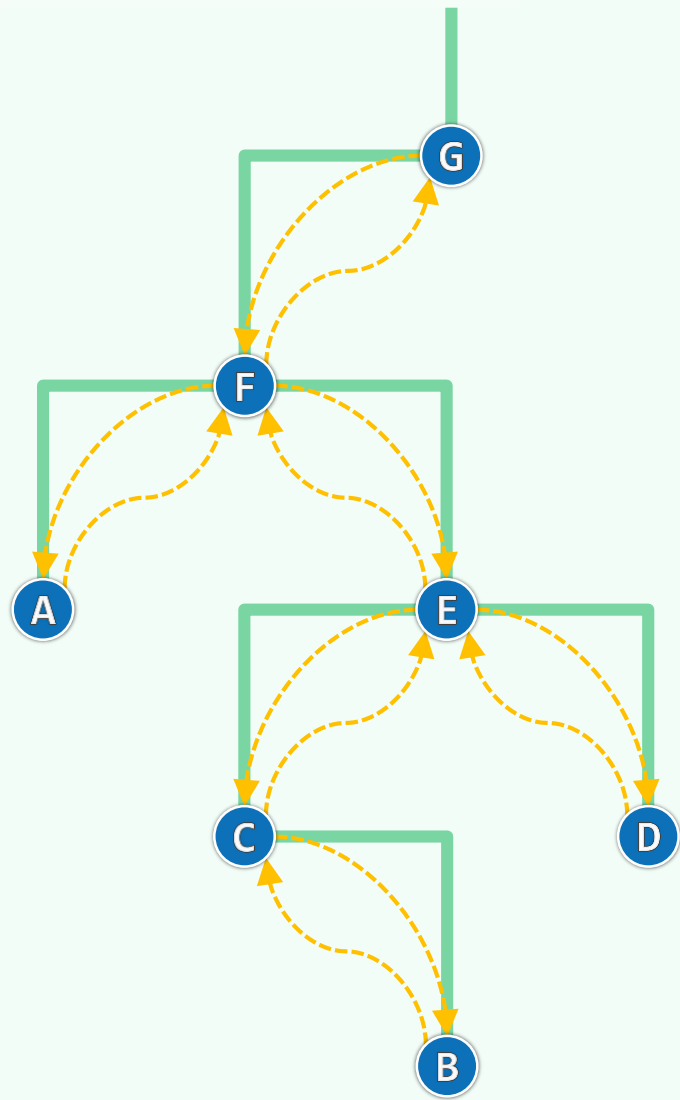
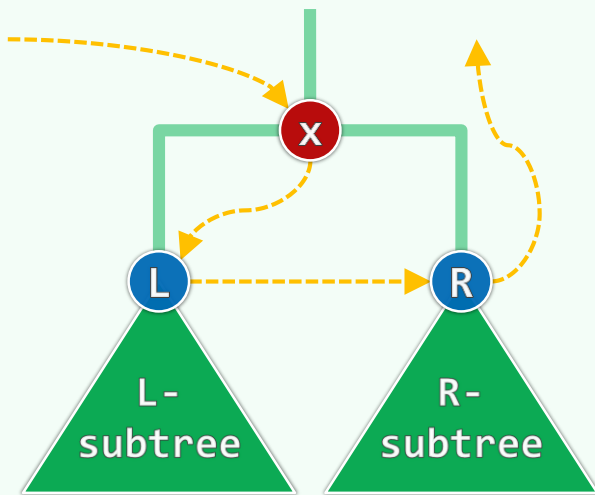
```
} //懒惰策略,  $O(n = |size|)$ 
```

# 递归实现

❖ 应用：先序输出文件树结构： `c:\> tree.com c:\windows`

❖ `template <typename T, typename VST>`

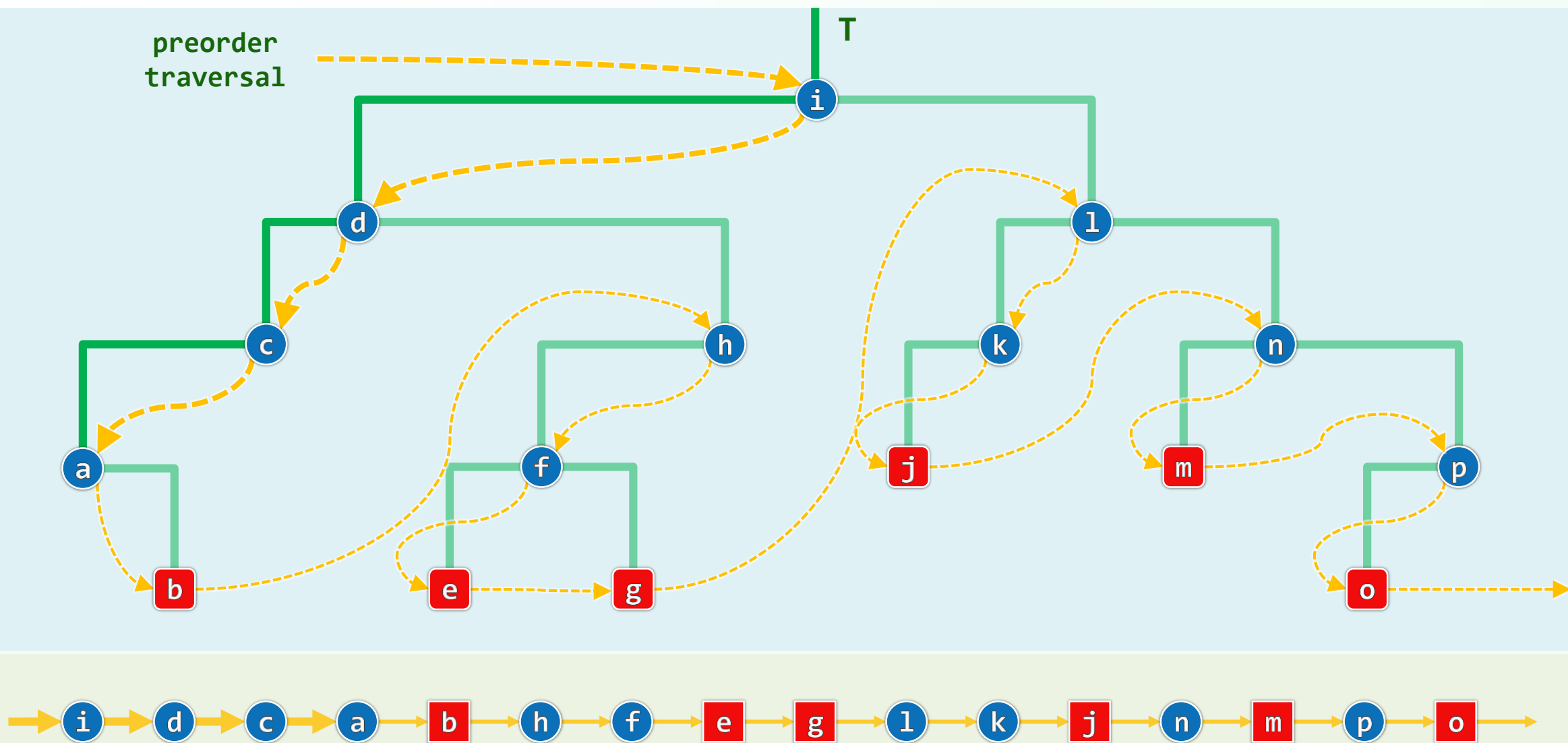
```
void traverse( BinNodePosi<T> x, VST & visit ) {  
    if ( ! x ) return;  
    visit( x->data );  
    traverse( x->lc, visit );  
    traverse( x->rc, visit );  
} //O(n)
```



❖ 制约：使用**默认**的Call Stack，允许的递归**深度**有限

❖ 挑战：不依赖递归机制，能否实现先序遍历？如何实现？效率如何？

# 观察



# 藤缠树

❖ 沿着左侧藤，整个遍历过程可分解为：

- 自上而下访问藤上节点，再
- 自下而上遍历各右子树

❖ 各右子树的遍历彼此独立

自成一个子任务

