

05-F4

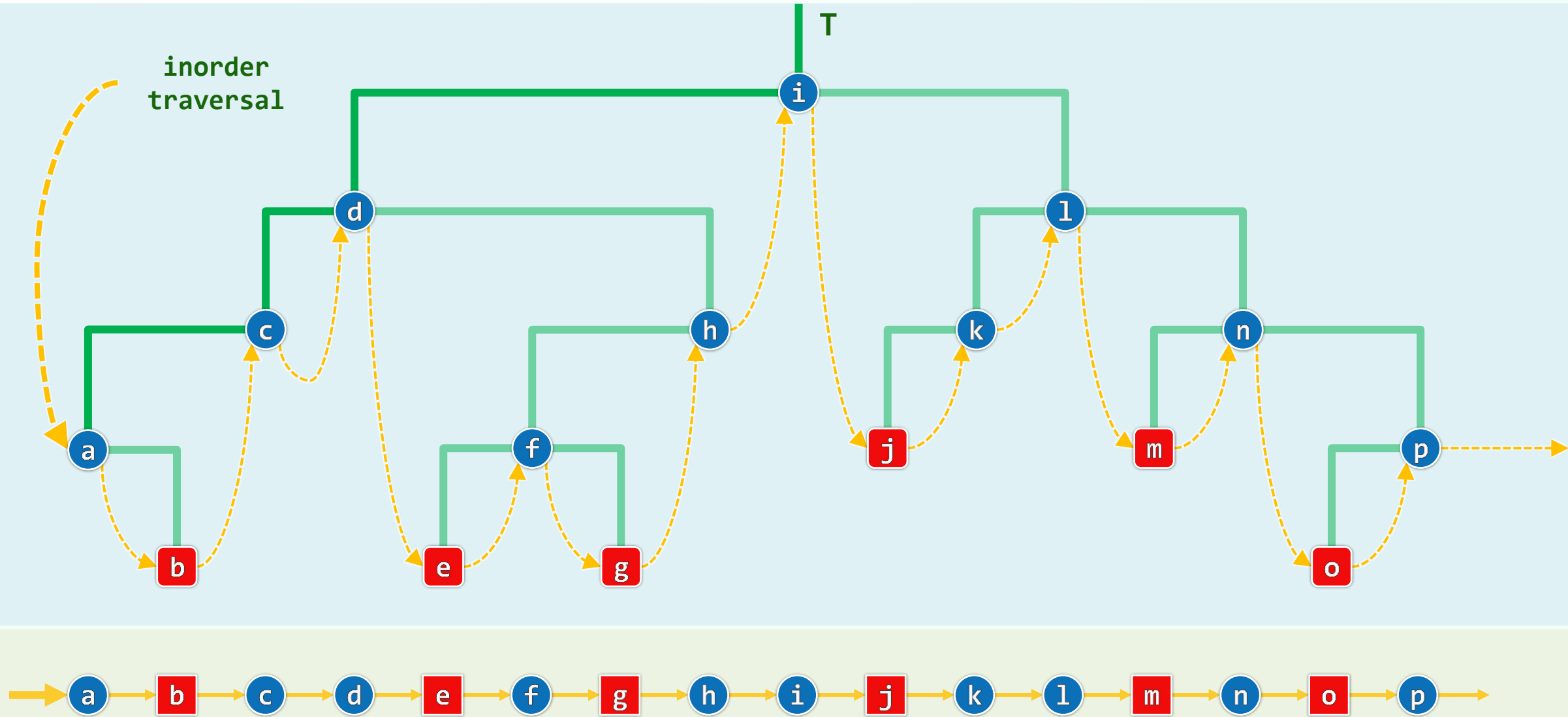
二叉树

中序遍历：后继与前驱

邓俊辉

deng@tsinghua.edu.cn

```
for ( BinNodePosi<T> t = first(); t; t = t->succ() )
```



## 直接后继：右后代

//在中序遍历意义下的直接后继

//稍后将被BST::remove中的removeAt()调用

```
template <typename T>
```

```
BinNodePosi<T> BinNode<T>::succ() {
```

```
    BinNodePosi<T> s = this;
```

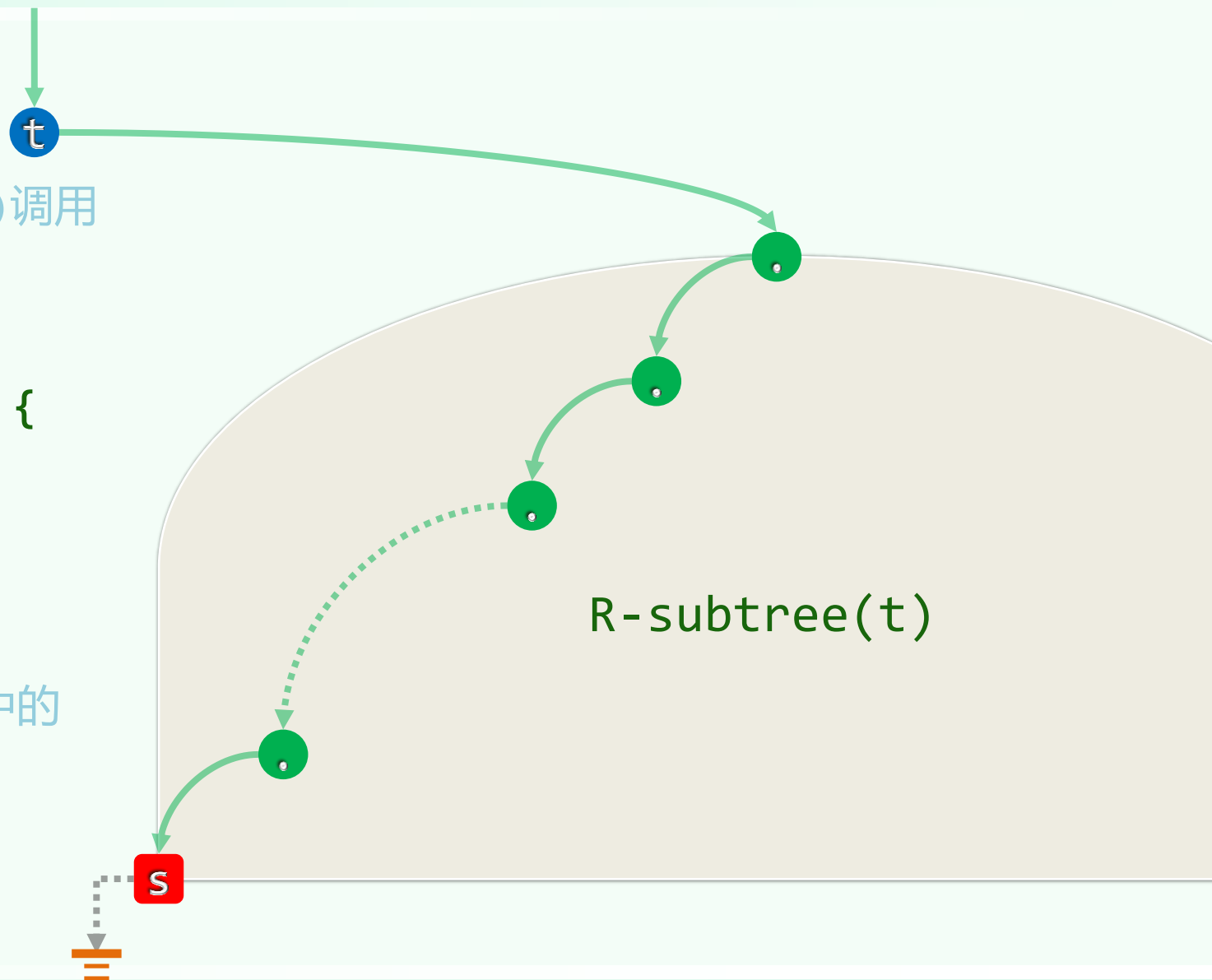
```
    if ( rc ) { //若有右孩子, 则
```

```
        s = rc; //直接后继必是右子树中的
```

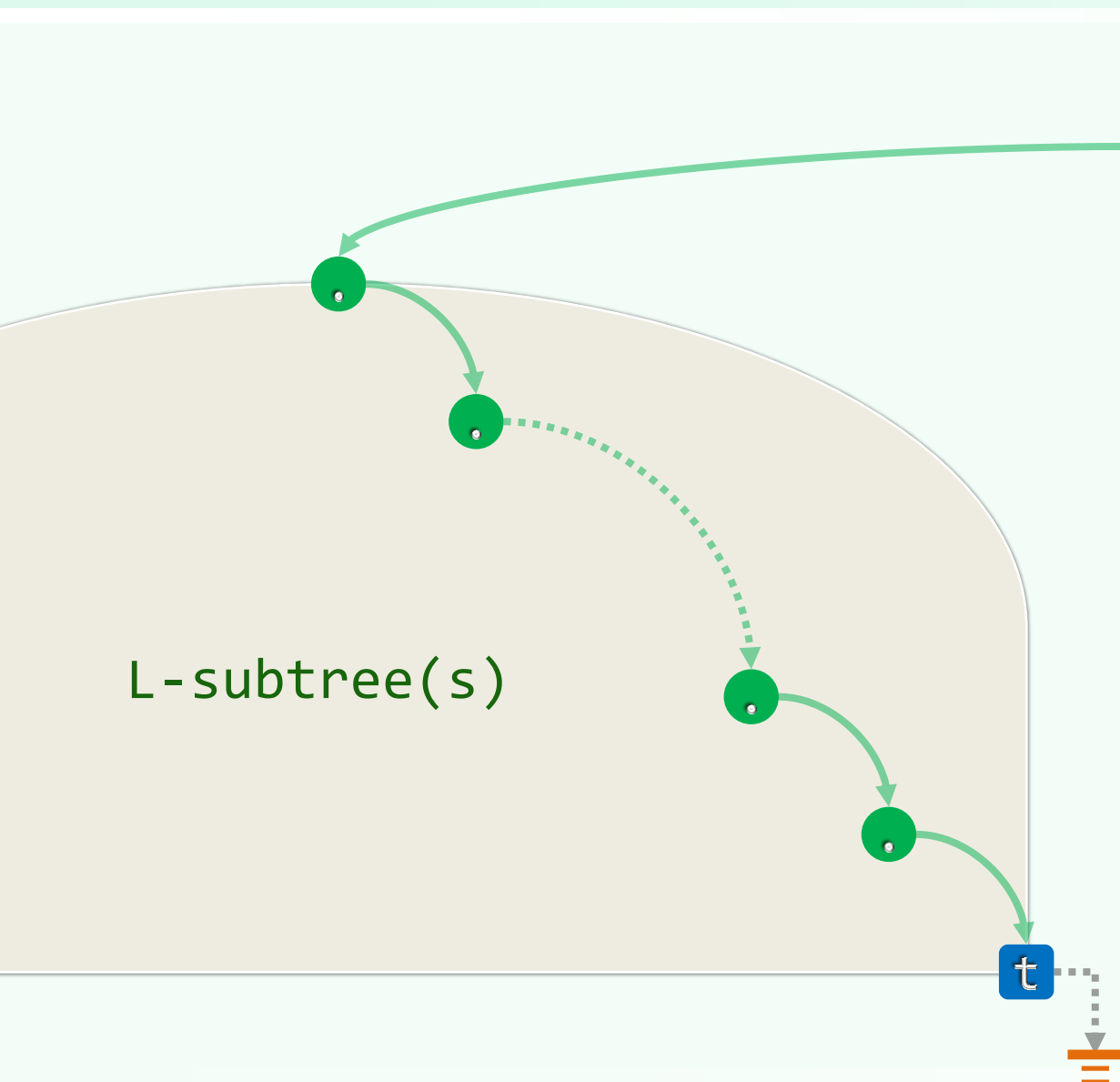
```
        while ( HasLChild( * s ) )
```

```
            s = s->lc; //最小节点
```

```
    } else /* ... */
```



## 直接后继：左父亲



```
} else { //否则
```

```
//后继应是 “以当前节点为直接前驱者”
```

```
while ( IsRChild( * s ) )
```

```
    s = s->parent; //不断朝左上移动
```

```
//最后再朝右上移动一步
```

```
    s = s->parent; //可能是NULL
```

```
}
```

```
return s; //两种情况下，运行时间分别为
```

```
} //当前节点的高度与深度，不过 $O(h)$ 
```