06-B1

二叉搜索树

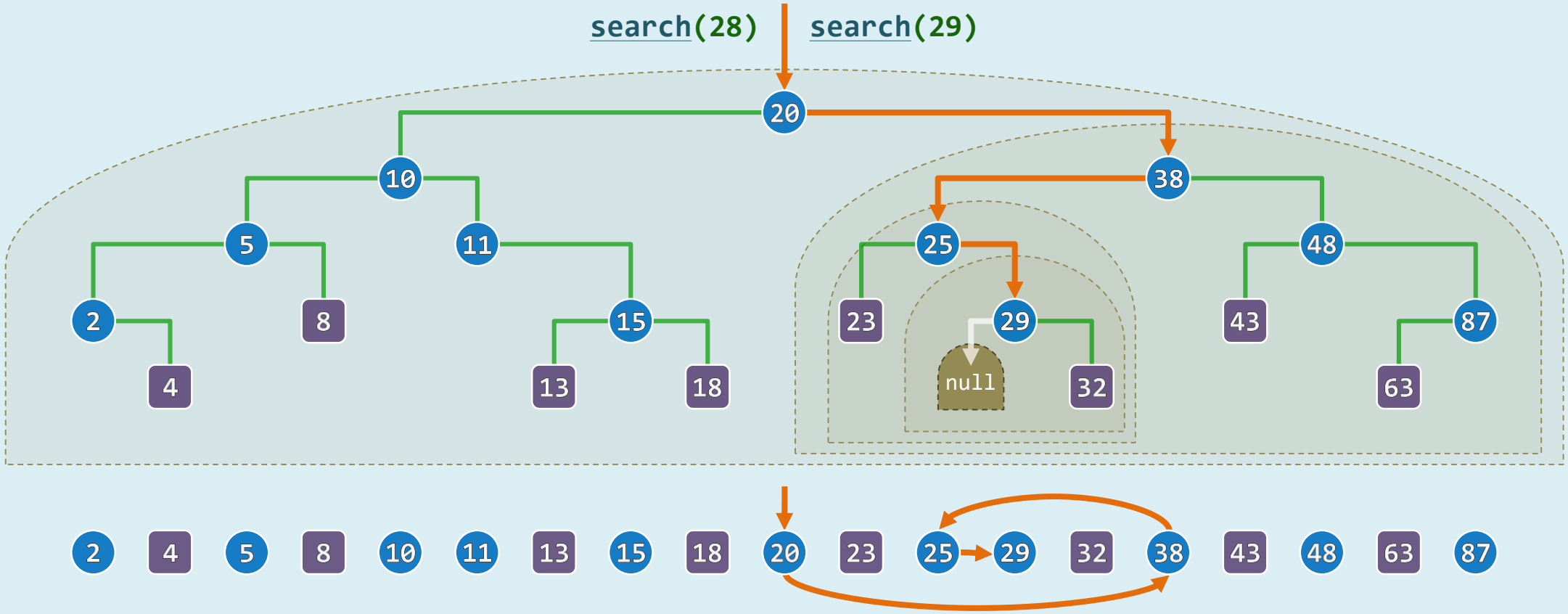
算法及实现：查找

邓俊辉

deng@tsinghua.edu.cn

为学日益，为道日损，损之又损，以至于无为，无为而无不为。

减而治之



❖ 从根节点出发，逐步地缩小查找范围，直到

- 发现目标（成功），或抵达空树（失败）

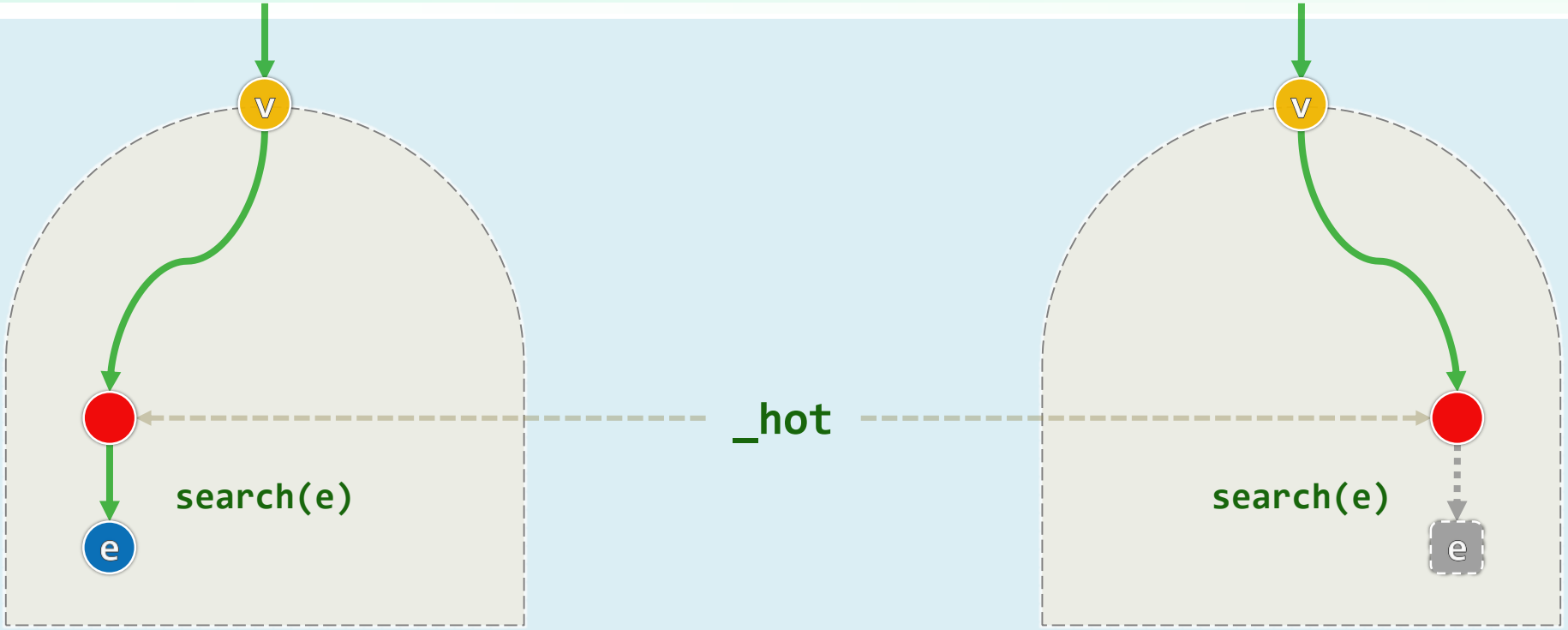
❖ 对照中序遍历序列可见，整个过程可视作是在

仿效有序向量的二分查找

实现

```
❖ template <typename T> BinNodePosi<T> & BST<T>::search( const T & e ) {
    if ( !_root || e == _root->data ) //空树，或恰在树根命中
       { _hot = NULL; return _root; }
    for ( _hot = _root; ; ) { //否则，自顶而下
       BinNodePosi<T> & v = ( e < _hot->data ) ? _hot->lc : _hot->rc; //深入一层
       if ( !v || e == v->data ) return v; _hot = v; //一旦命中或抵达叶子，随即返回
    } //返回目标节点位置的引用，以便后续插入、删除操作
} //无论命中或失败，_hot均指向v之父亲（v是根时，hot为NULL）
```

返回的引用值



- 查找成功时，指向一个关键码为e且真实存在的节点
- 失败时，指向最后一次试图转向的空节点NULL——随后可视需要进行修改

  此时，不妨假想地将该空节点转换为一个数值为e的哨兵节点