08-B1

二叉搜索树

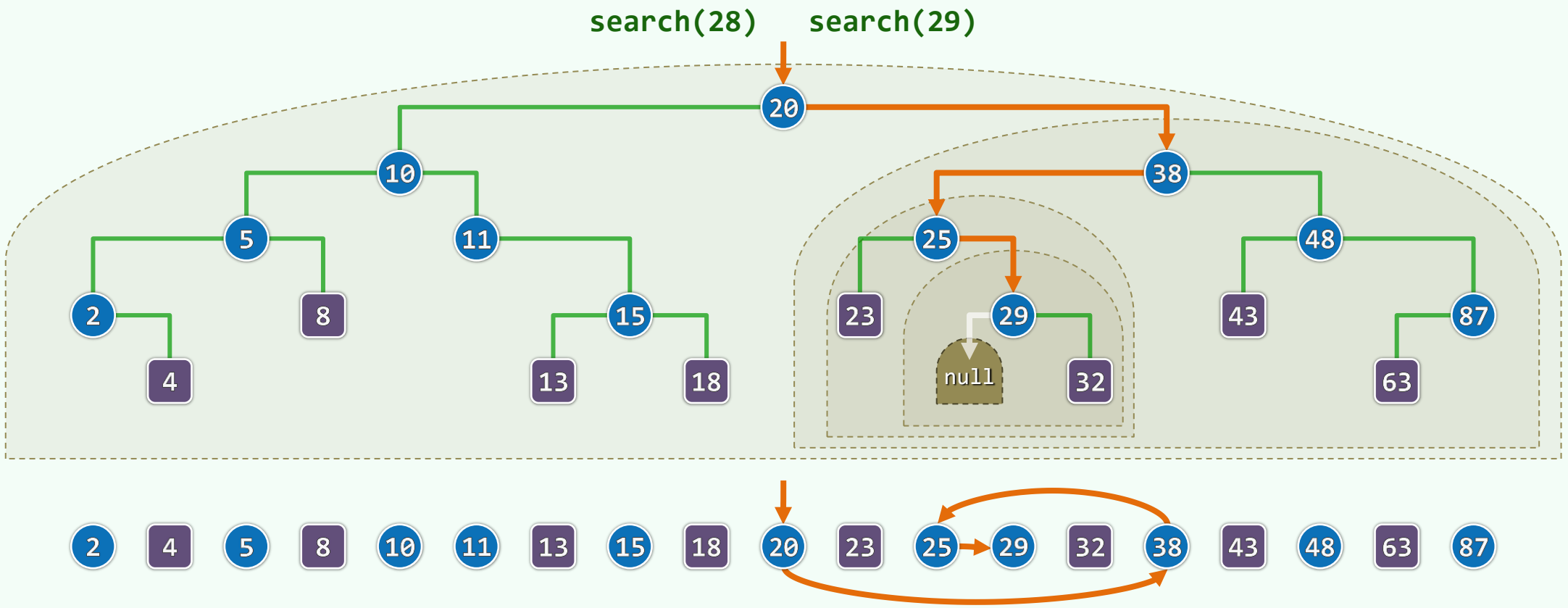
算法及实现：查找

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)
    { return searchIn( _root, e, _hot = NULL ); } //从根节点启动查找
❖static BinNodePosi(T) & searchIn( //当前（子）树根、目标关键码、记忆热点
    BinNodePosi(T) & v,  const T & e, BinNodePosi(T) & hot) {
    if ( !v || e == v->data ) return v; hot = v; //在树根v处命中
    while (1) { //自顶而下
       BinNodePosi(T) & c = (e < hot->data) ? hot->lc : hot->rc; //确定方向
       if ( !c || e == c->data ) return c; hot = c; //命中返回，或者深入一层
    } //无论命中或失败，hot均指向v之父亲（或为NULL）
  } //运行时间正比于返回节点v的深度，不超过树高O(h)
```

接口语义约定

❖ 返回的引用值

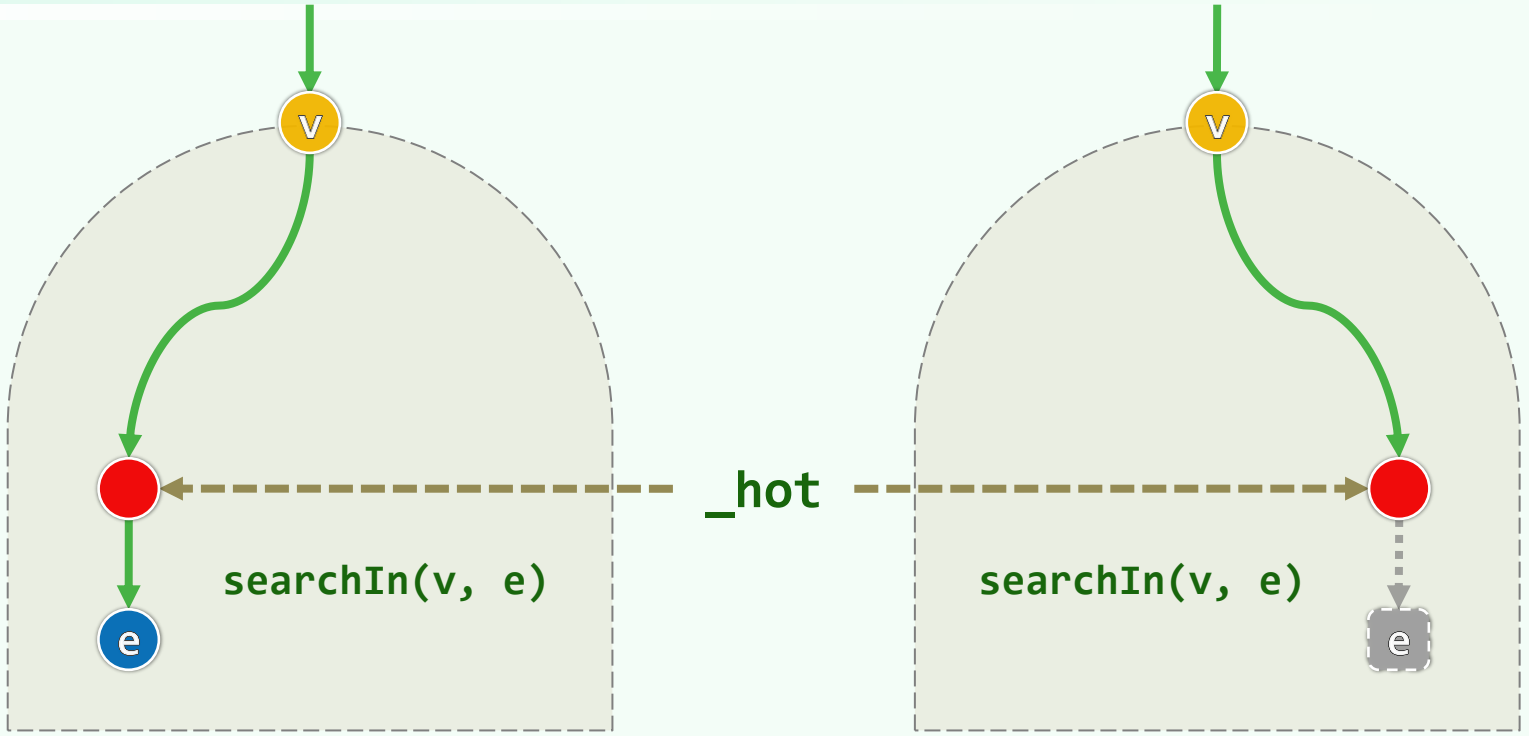
- 成功时，指向一个

  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

  （随后可视需要修改）



❖ 失败时，不妨假想地将此空节点，转换为一个数值为e的哨兵节点

如此，依然满足BST的充要条件；而且更重要地...

❖ 无论成功与否：返回值总是等效地指向命中节点，而_hot总是指向命中节点的父亲