10-C2

高级搜索树

红黑树：结构

这时，我看见两只大蚂蚁，一只红不棱登，另一只个儿特大，差不离有半英寸长，是黑不溜秋的，它们两个正在相互凶殴...

玉帝即传旨宣托塔李天王，教：“把照妖镜来照这厮谁真谁假，教他假灭真存。”

邓俊辉

deng@tsinghua.edu.cn

红与黑

- 1972, R. Bayer

  Symmetric Binary B-Tree

- 1978, L. Guibas & R. Sedgewick

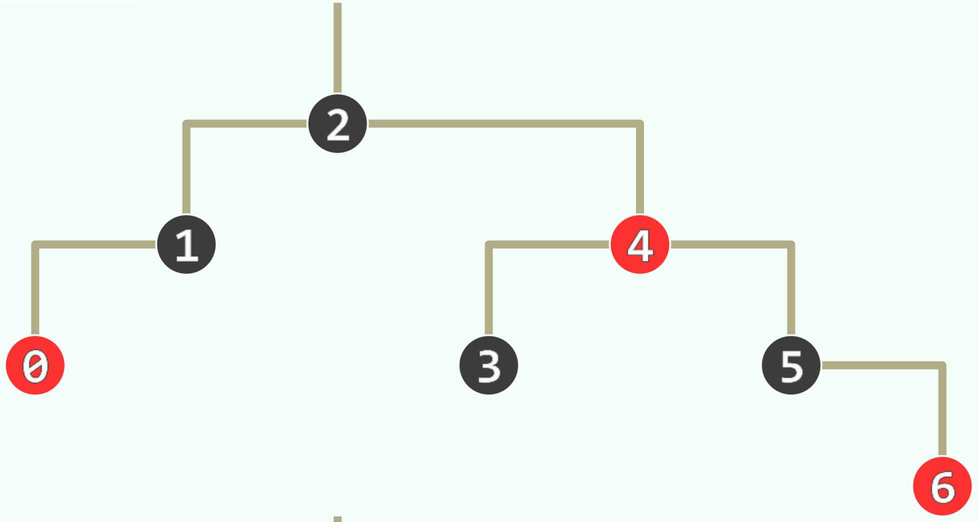
  Red-Black Tree

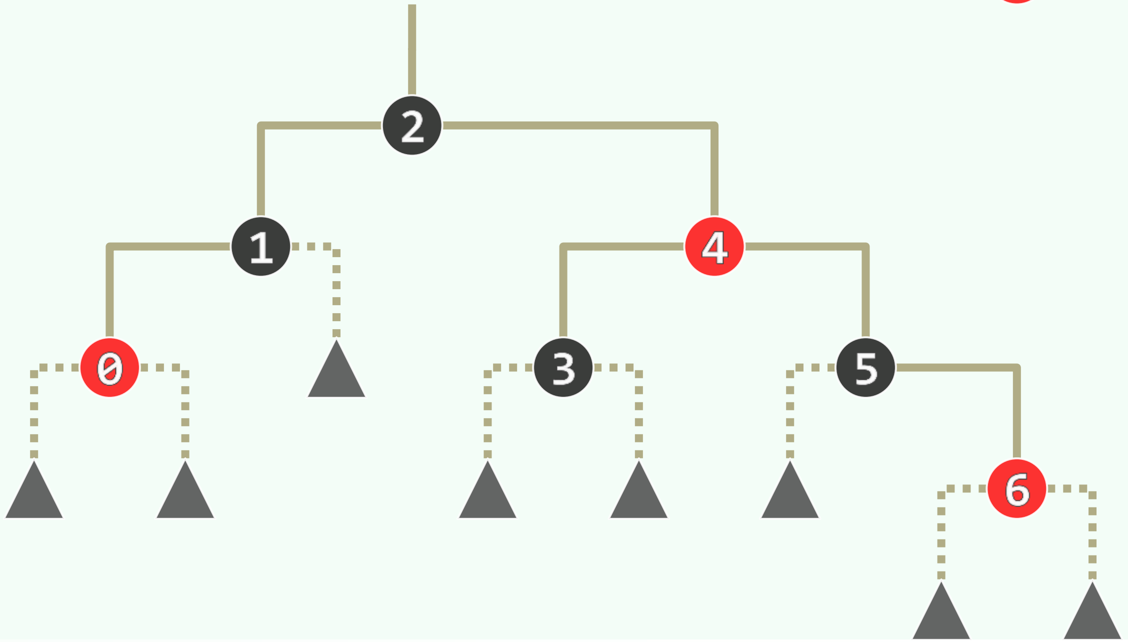
- 1982, H. Olivie

  Half-Balanced Binary Search Tree

- **由红、黑两类节点组成的BST**

  统一增设外部节点NULL，使之成为真二叉树





规则

1） 树根：必为黑色

2） 外部节点：均为黑色

3） 其余节点：若为红，则只能有黑孩子

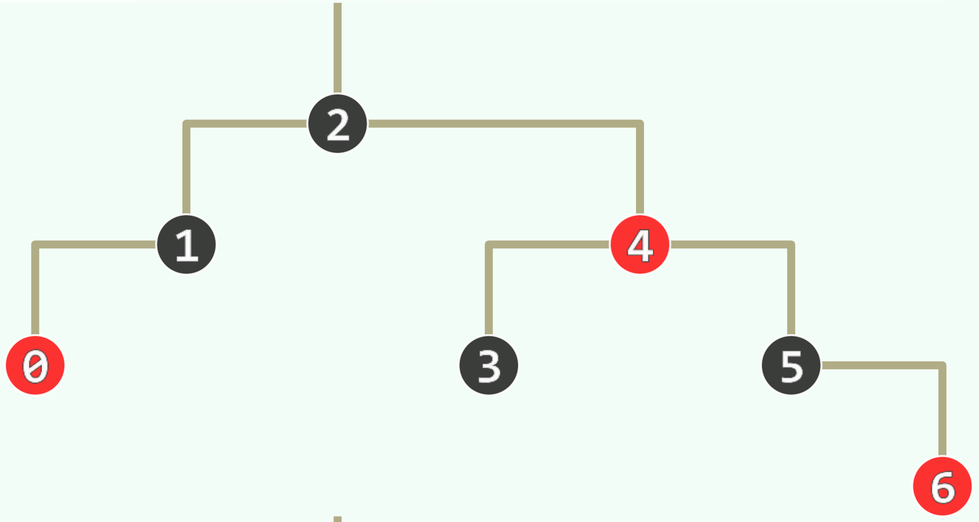
//红之子、之父必黑

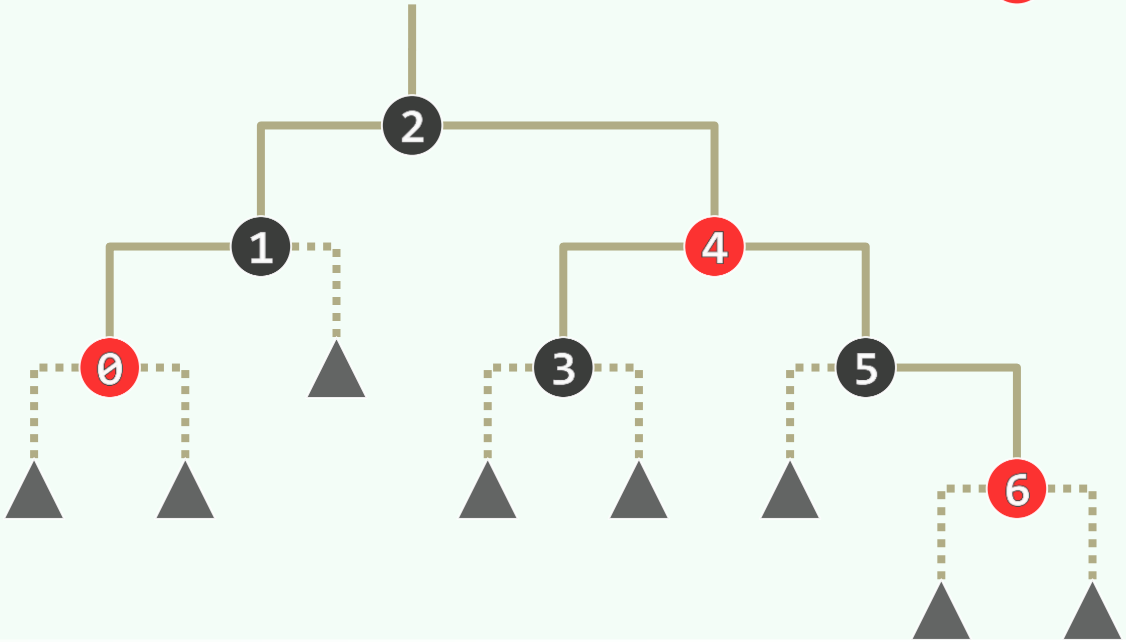
4） 外部节点到根：途中黑节点数目（黑深度）相等

❖ 节点的颜色，只能显式地记录？

❖ 以上定义颇为费解，有直观解释吗？

❖ 如此定义的BST，也是BBST？



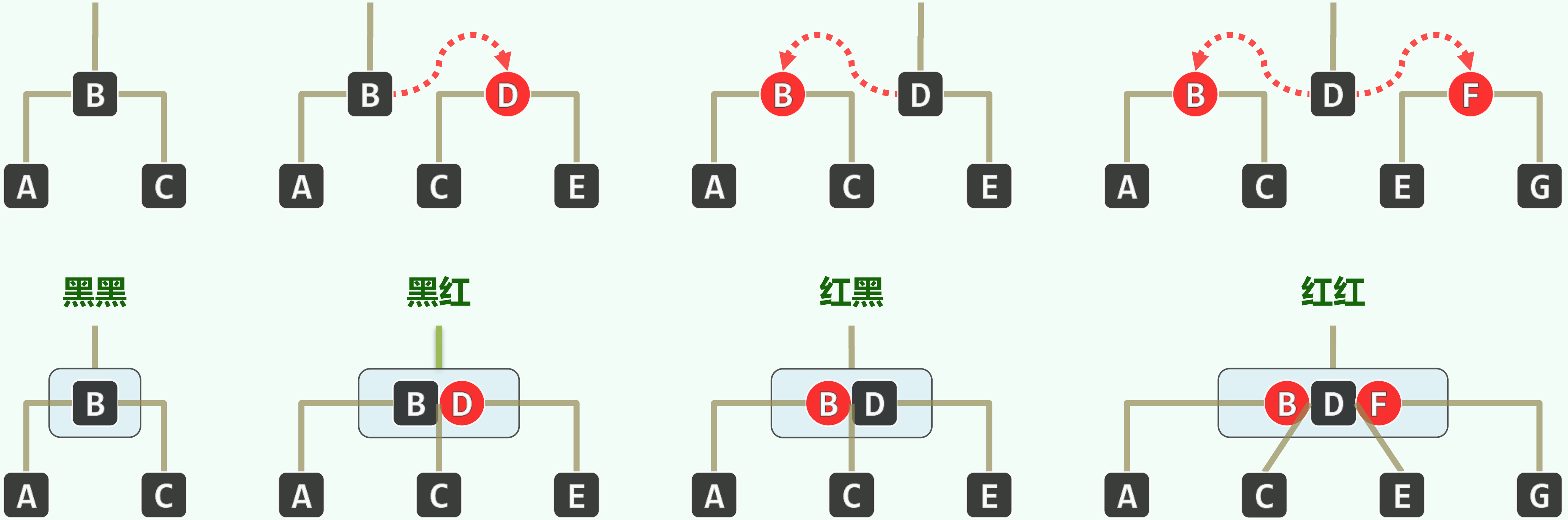


红黑树 = (2,4)树

❖ 将红节点**提升**至与其（黑）父亲等高——于是每棵红黑树，都对应于一棵(2,4)-树

❖ 将黑节点与其红孩子（们）视作关键码，并合并为B-树的**超级节点**...

❖ 无非四种组合，分别对应于**4阶B-树**的一类内部节点 //反过来呢？



红黑树 $\in$ BBST

❖ 包含n个内部节点的红黑树T，高度 $h = \mathcal{O}(\log n)$ //既然B-树是平衡的，由等价性红黑树也应是

$$\log_2(n+1) \le h \le 2 \cdot \log_2(n+1)$$

❖ 若T高度为h，红/黑高度为R/H，则

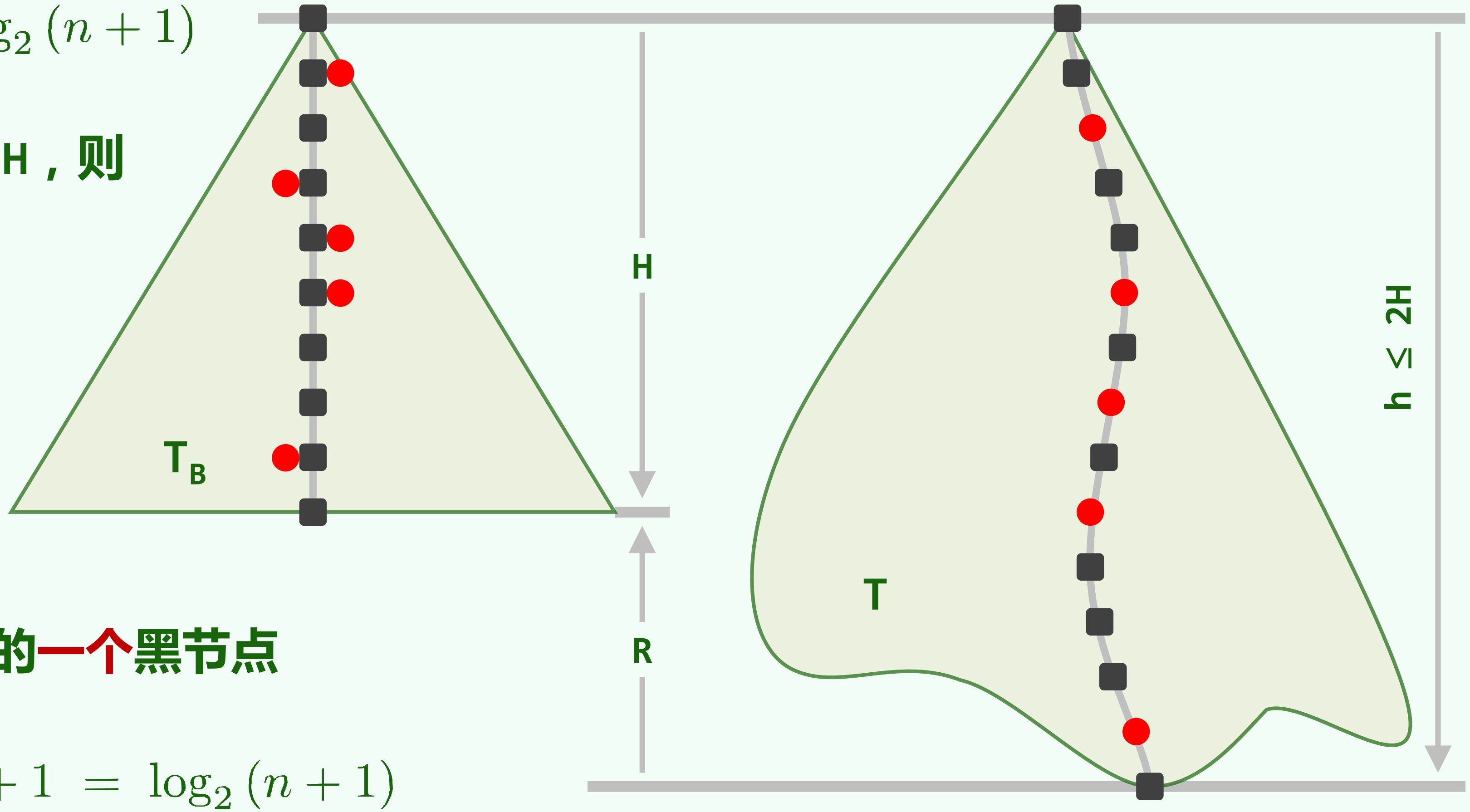
$$h \le R + H \le 2 \cdot H$$

❖ 若T所对应的B-树为T_B

则H即是T_B的高度

❖ T_B的每个节点，都恰好包含T的一个黑节点

❖ 于是，$H \le \log_{\lceil 4/2 \rceil} \frac{n+1}{2} + 1 = \log_2(n+1)$



RedBlack

❖
```
template <typename T> class RedBlack : public BST<T> { //红黑树
public:        //BST::search()等其余接口可直接沿用
               BinNodePosi(T) insert( const T & e ); //插入(重写)
               bool remove( const T & e ); //删除(重写)
protected:     void solveDoubleRed( BinNodePosi(T) x ); //双红修正
               void solveDoubleBlack( BinNodePosi(T) x ); //双黑修正
               int updateHeight( BinNodePosi(T) x ); //更新节点x的高度
};
```

❖
```
template <typename T> int RedBlack<T>::updateHeight( BinNodePosi(T) x ) {
   x->height = max( stature( x->lc ), stature( x->rc ) );
   if ( IsBlack( x ) ) x->height++; return x->height; //只计黑节点
}
```