

12-F3

优先级队列

左式堆：合并算法

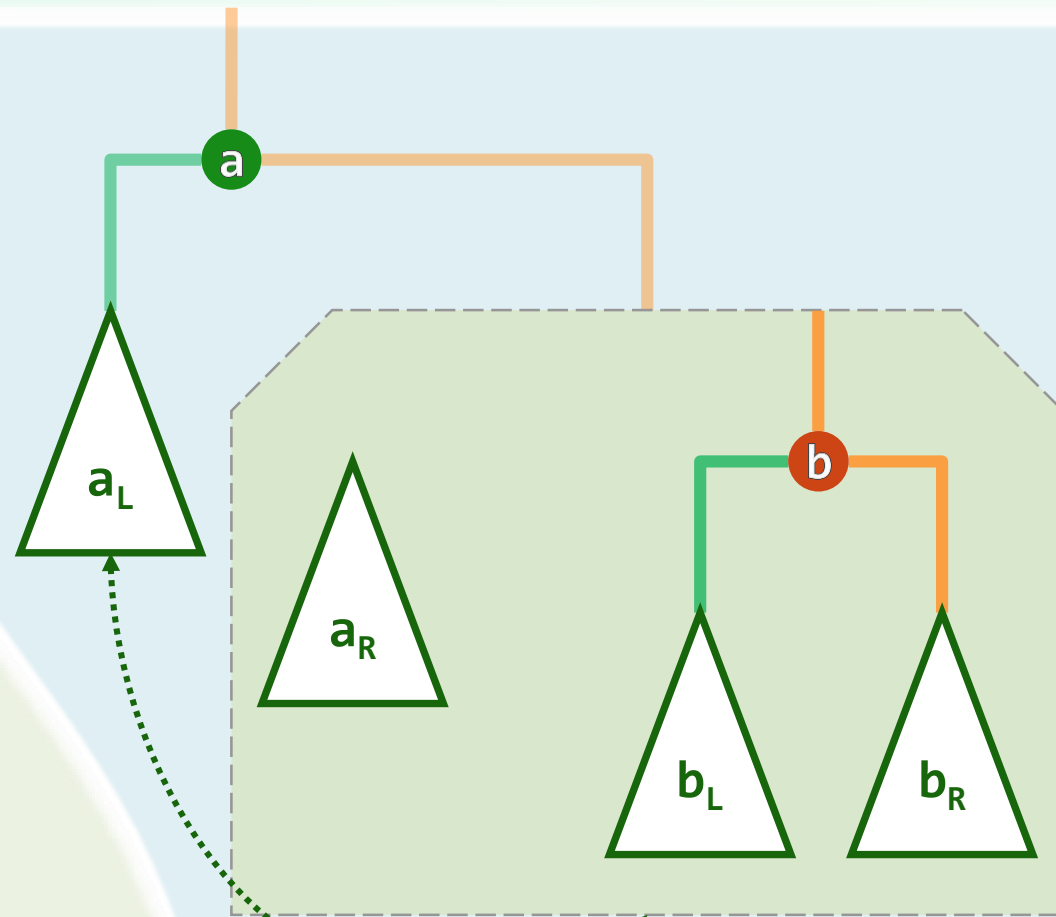
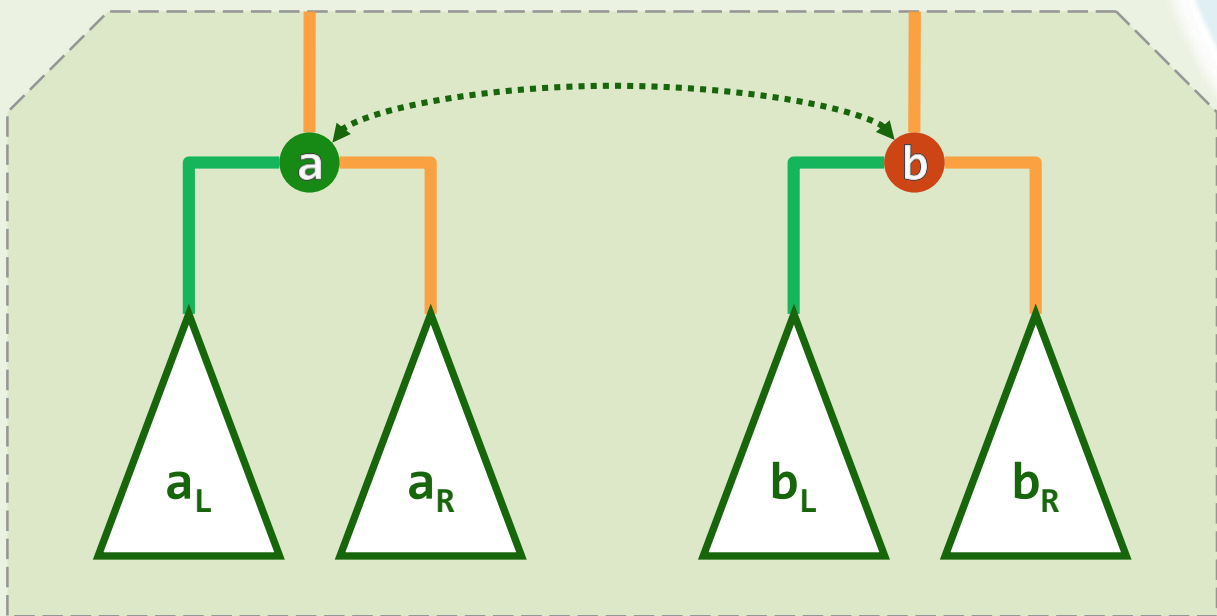
邓俊辉

deng@tsinghua.edu.cn

左之左之，君子宜之；右之右之，君子有之

递归：前处理 + 后处理

Before:
compare priority &
swap if necessary

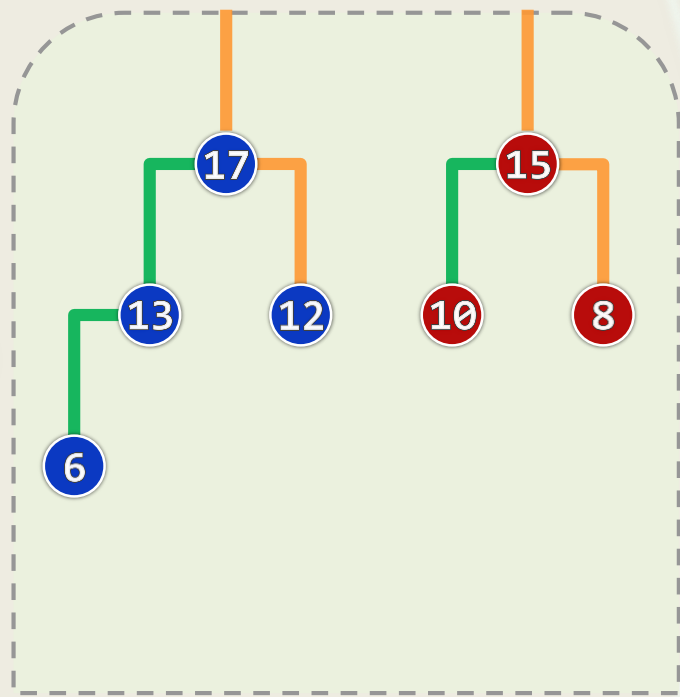


After:
compare NPL &
flip if necessary

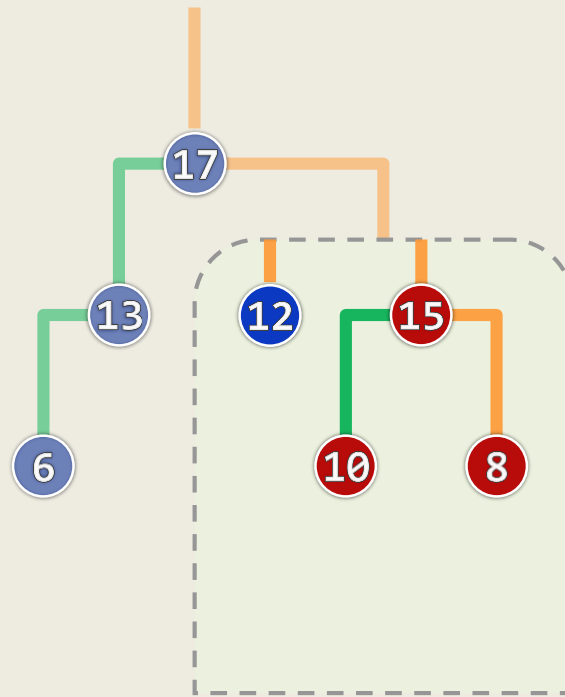
递归实现

```
template <typename T> BinNodePosi<T> merge( BinNodePosi<T> a, BinNodePosi<T> b ) {  
    if ( !a ) return b; if ( !b ) return a; //递归基  
    if ( lt( a->data, b->data ) ) swap( a, b ); //确保a>=b  
    ( a->rc = merge( a->rc, b ) )->parent = a; //将a的右子堆, 与b合并  
    if ( !a->lc || (a->lc->npl < a->rc->npl) ) //若有必要  
        swap( a->lc, a->rc ); //交换a的左、右子堆, 以确保左子堆的npl不小  
    a->npl = a->rc ? 1 + a->rc->npl : 1; //更新a的npl  
    return a; //返回合并后的堆顶  
}
```

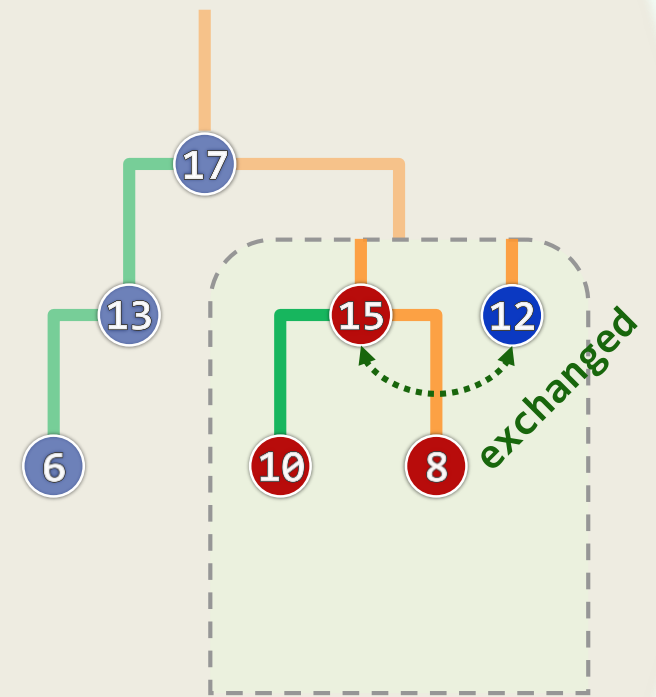
实例 (1/5)



(a)

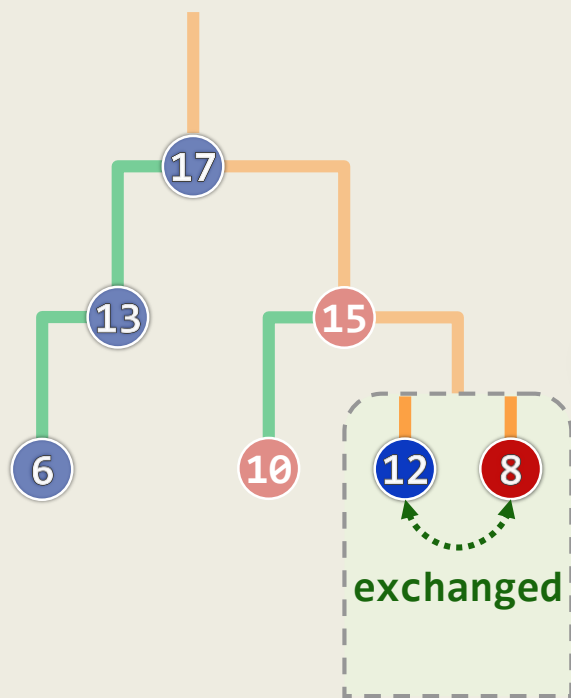


(b)

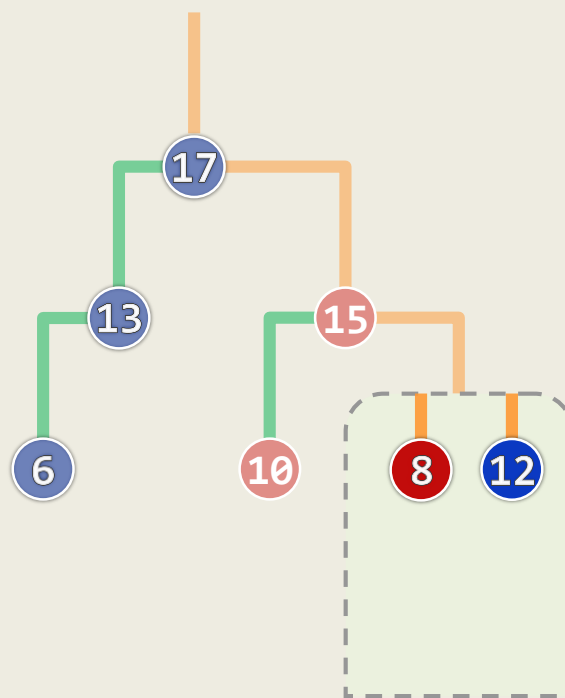


(c)

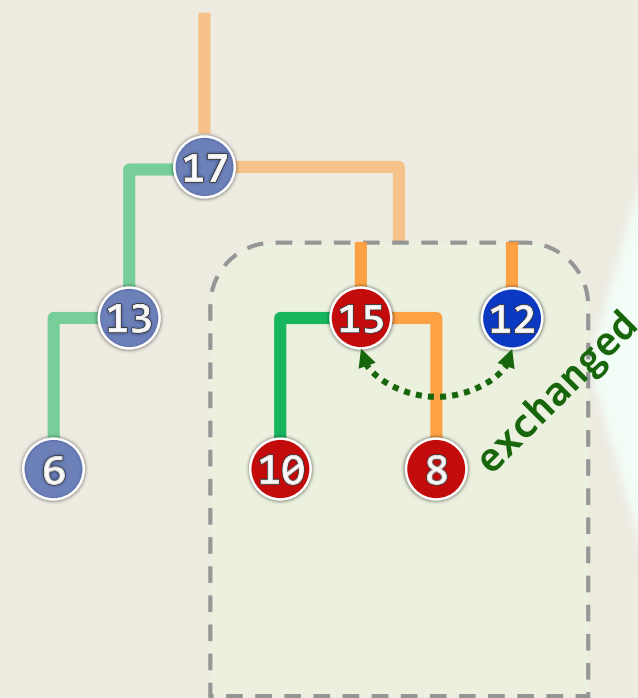
实例 (2/5)



(e)

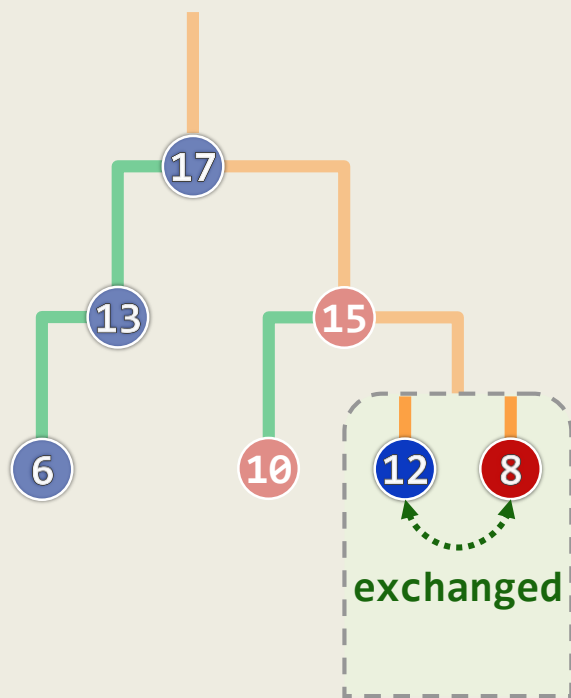


(d)

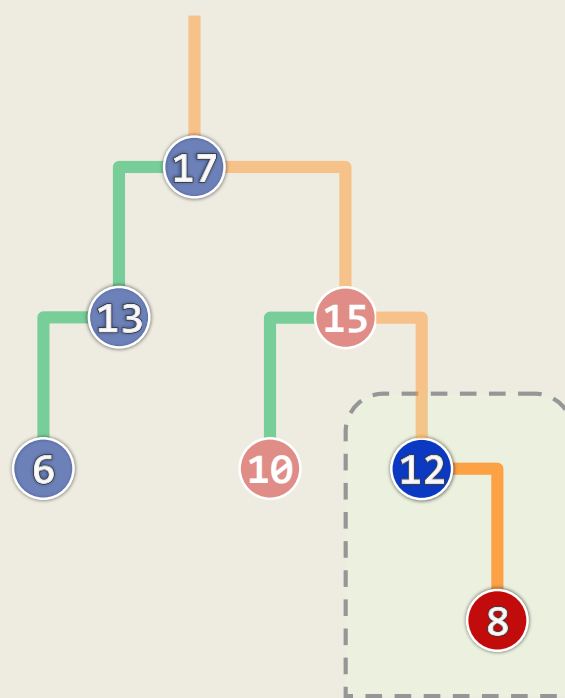


(c)

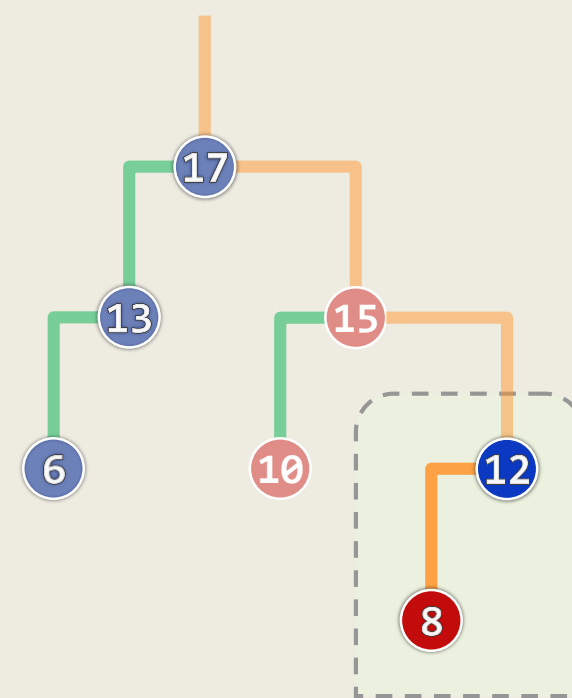
实例 (3/5)



(e)

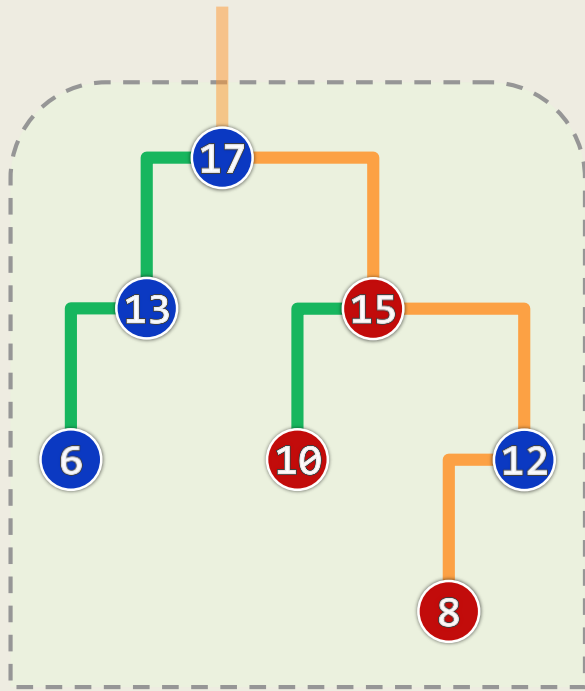


(f)

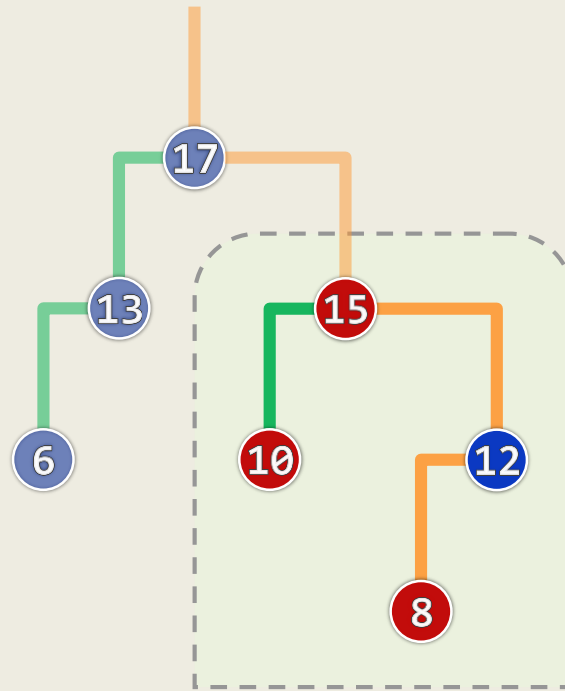


(g)

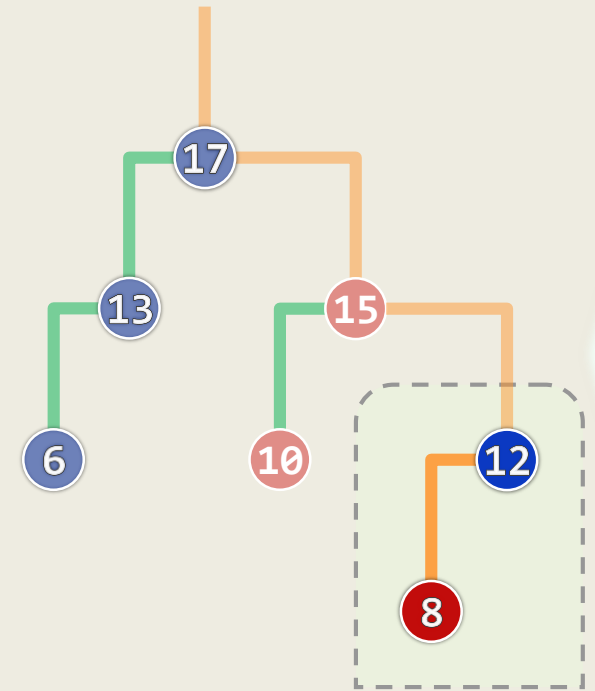
实例 (4/5)



(i)

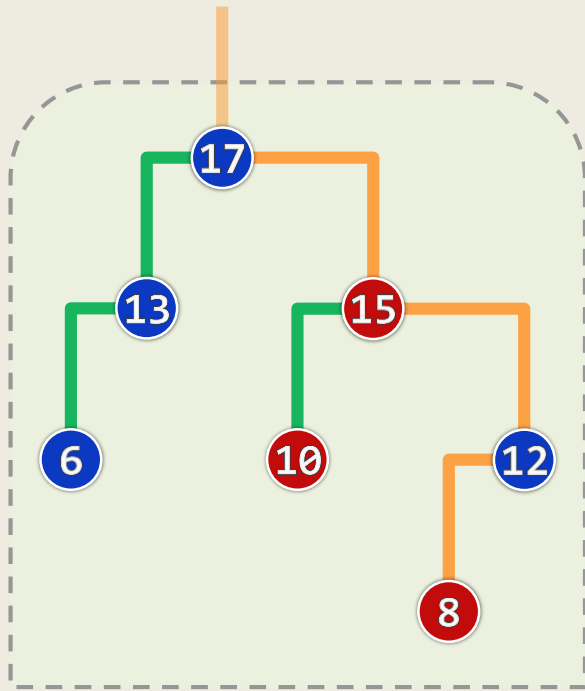


(h)

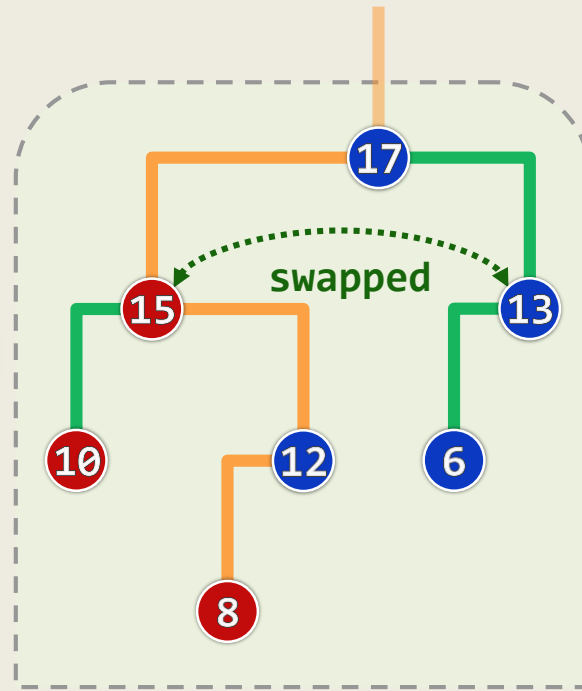


(g)

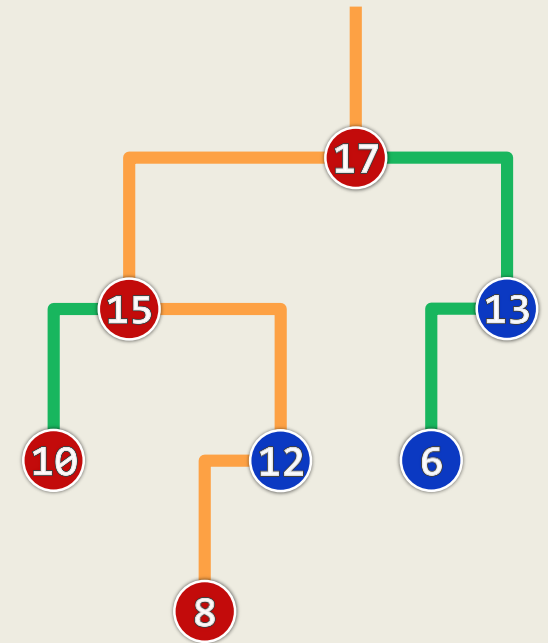
实例 (5/5)



(i)



(j)



(k)

迭代实现

```
template <typename T> BinNodePosi<T> merge( BinNodePosi<T> a, BinNodePosi<T> b ) {  
    if ( !a ) return b; if ( !b ) return a; //退化情况  
    if ( lt( a->data, b->data ) ) swap( a, b ); //确保a>=b  
    for ( ; a->rc; a = a->rc ) //沿右侧链做二路归并, 直至堆a->rc先于b变空  
        if ( lt( a->data, b->data ) ) { b->parent = a; swap( a->rc, b ); } //接入b  
    (a->rc = b)->parent = a; //直接接入b的残余部分 (必然非空)  
    for ( ; a; b = a, a = a->parent ) { //从a出发沿右侧链逐层回溯 (b == a->rc)  
        if ( !a->lc || (a->lc->npl < a->rc->npl) ) swap( a->lc, a->rc ); //确保npl合法  
        a->npl = a->rc ? a->rc->npl + 1 : 1; //更新npl  
    }  
    return b; //返回合并后的堆顶  
} //merge()
```