

12-B2

优先级队列

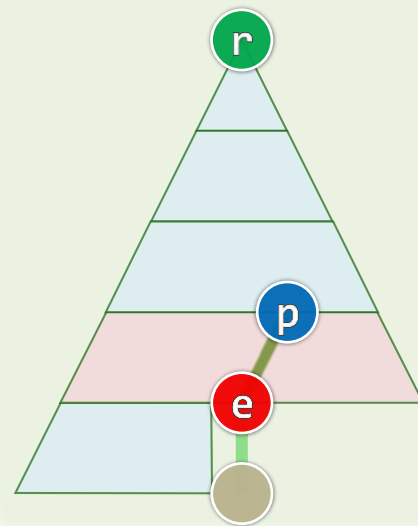
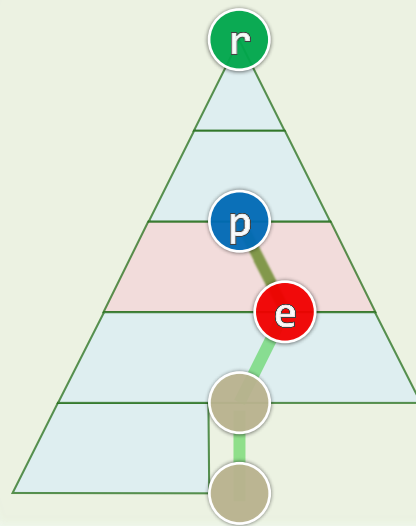
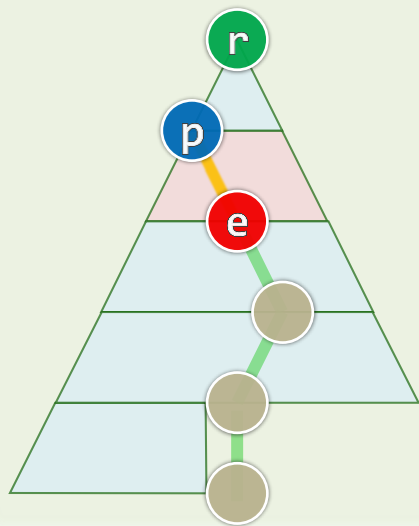
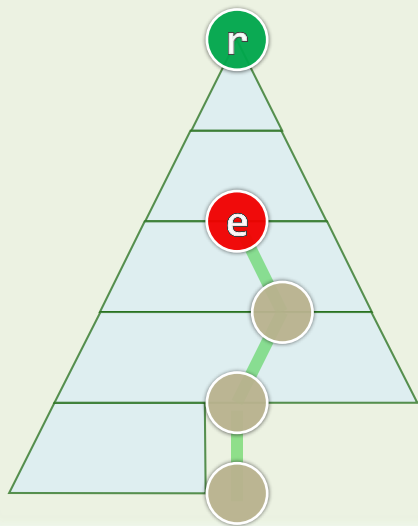
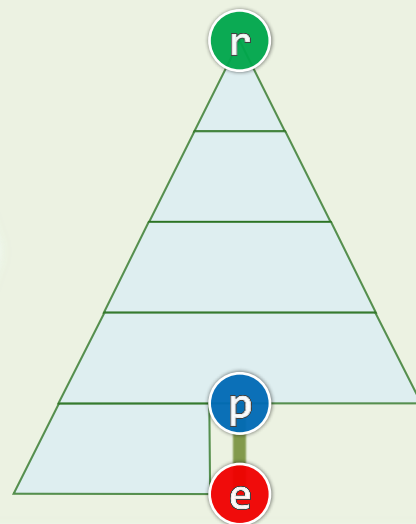
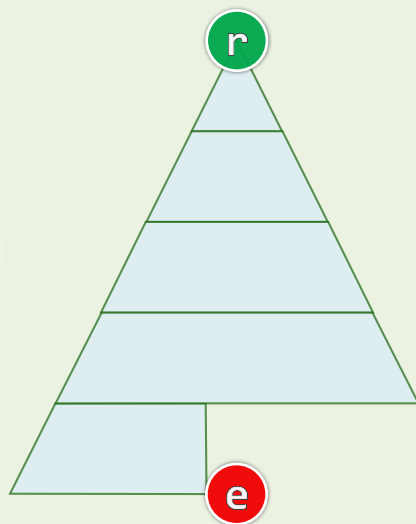
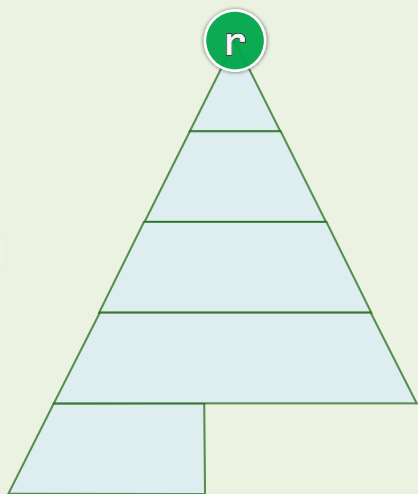
完全二叉堆：插入

时迁看见土地庙后一株大柏树，便把两只腿夹定，一节节爬将上去树头顶，骑马儿坐在枝柯上。

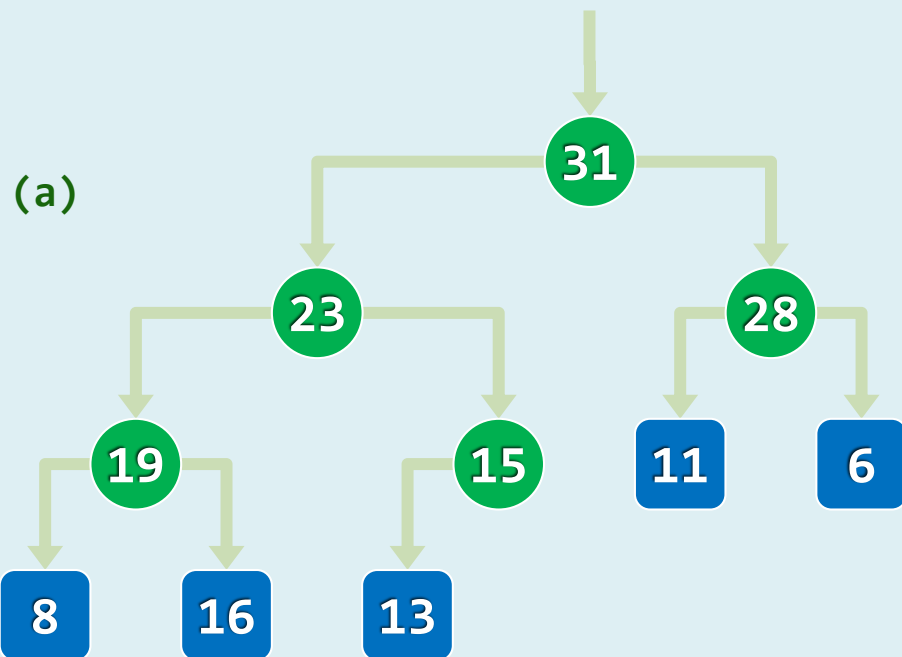
邓俊辉

deng@tsinghua.edu.cn

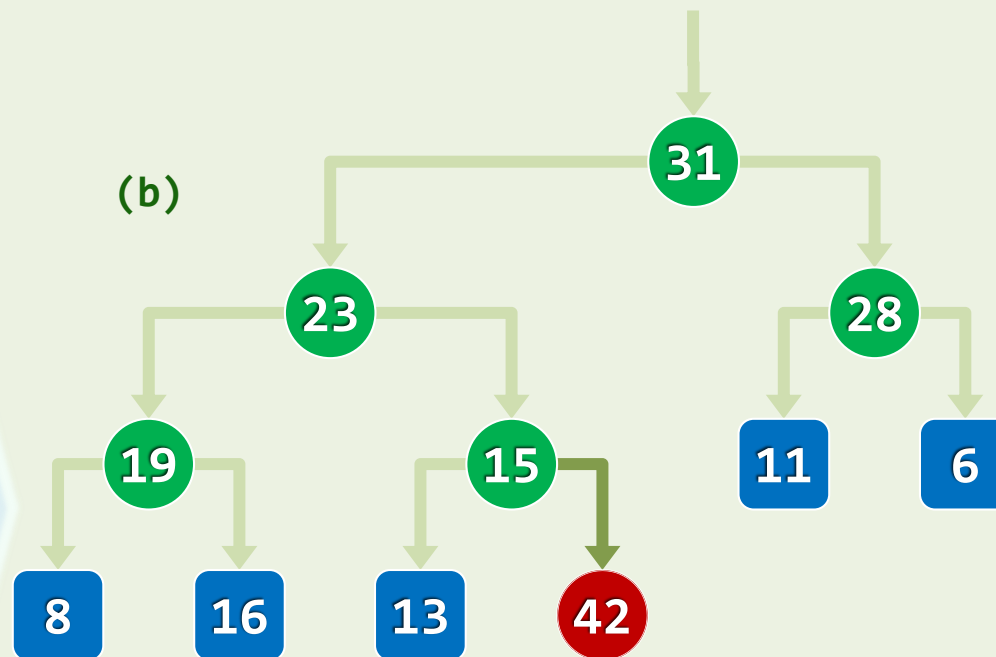
算法：逐层上滤



实例 (1/5)

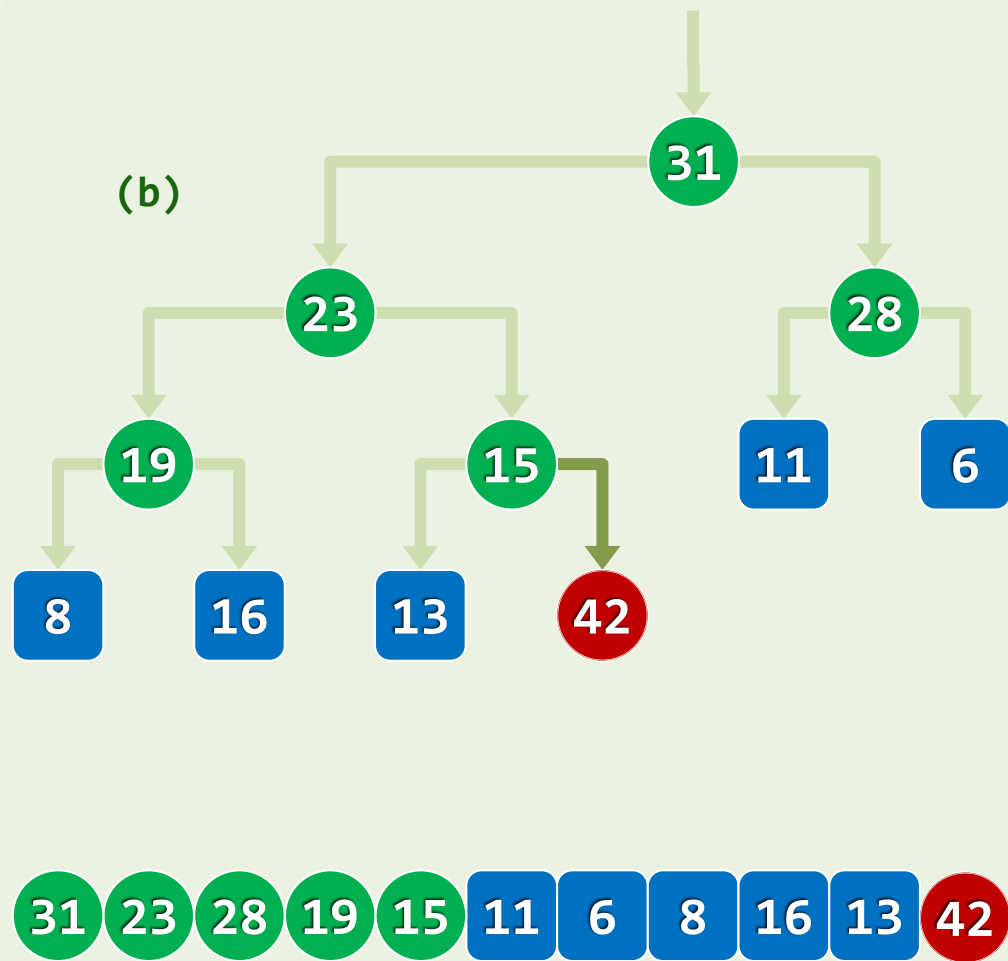
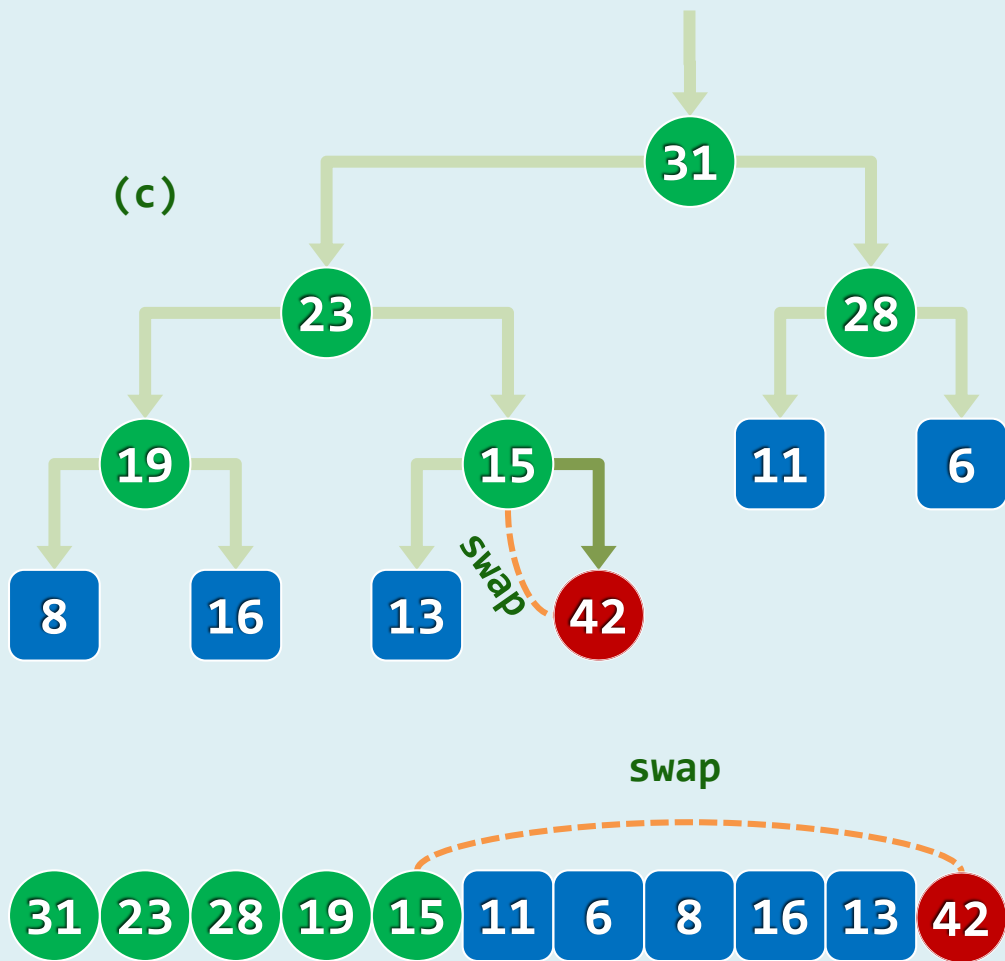


31 23 28 19 15 11 6 8 16 13

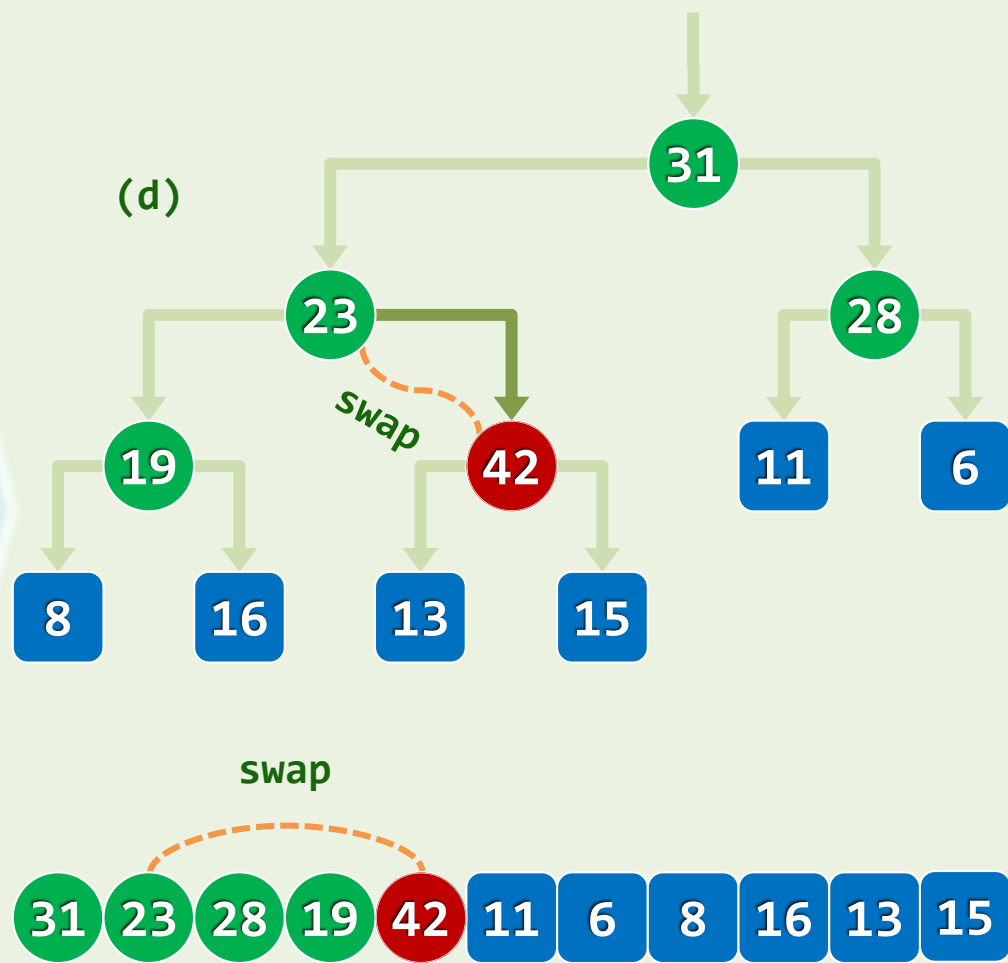
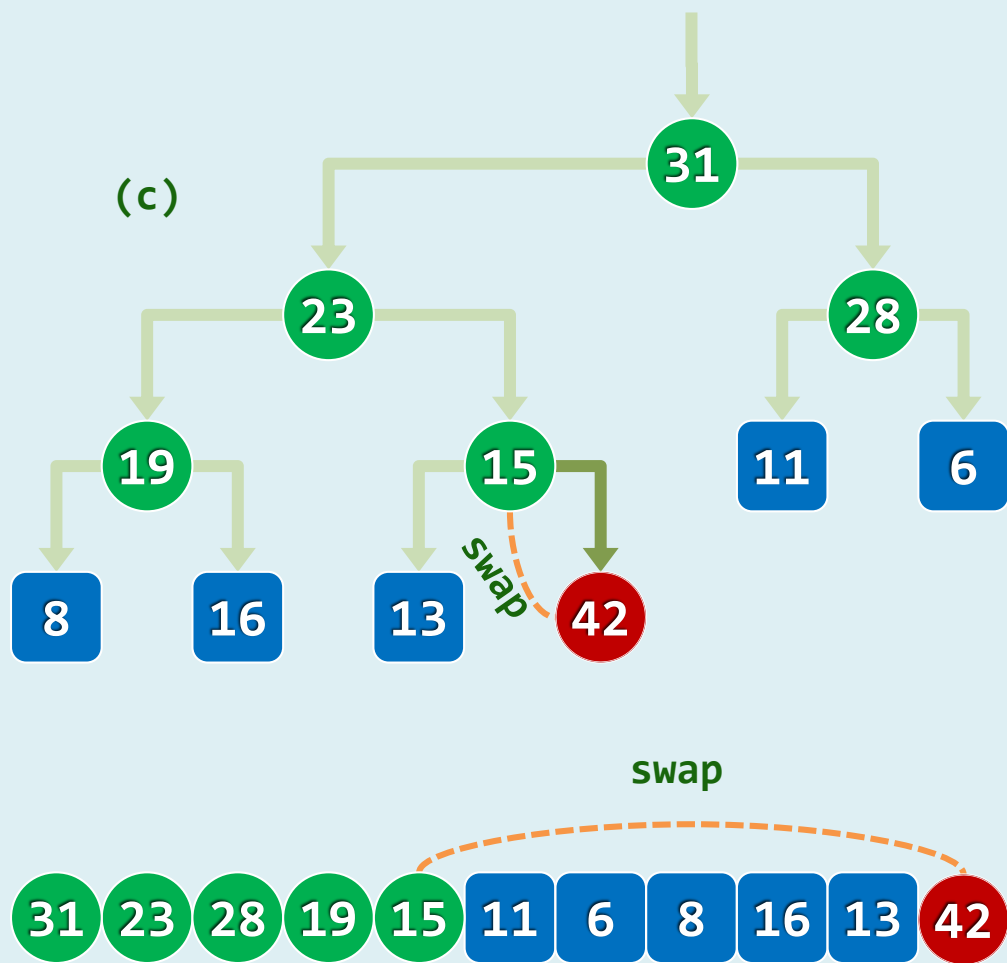


31 23 28 19 15 11 6 8 16 13 42

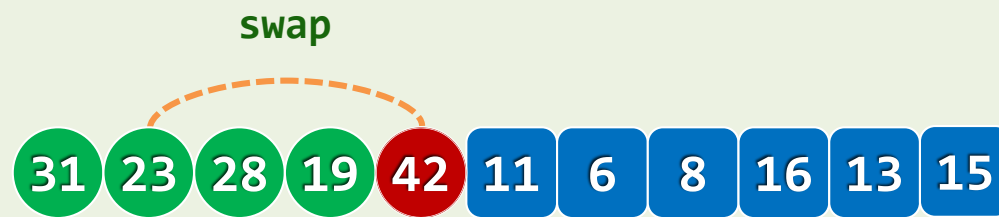
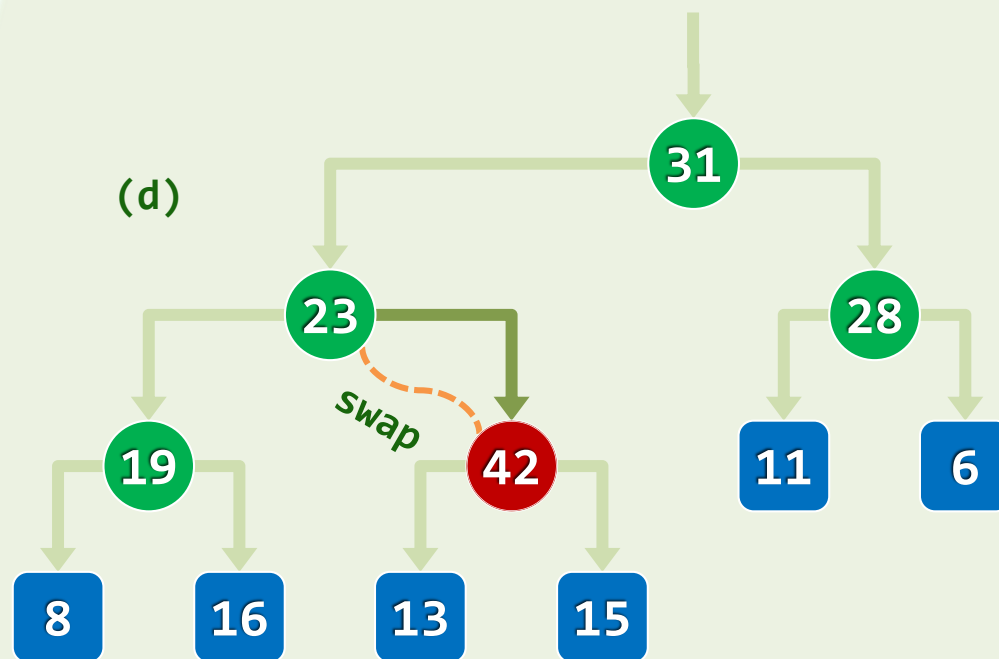
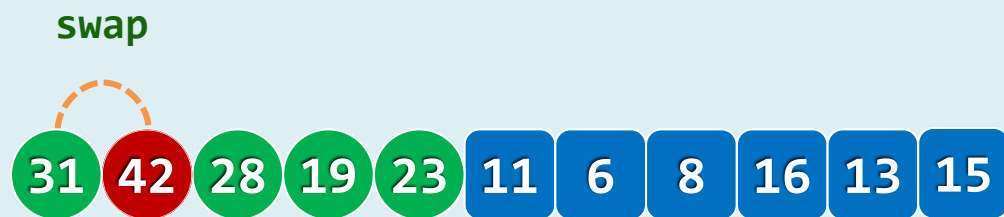
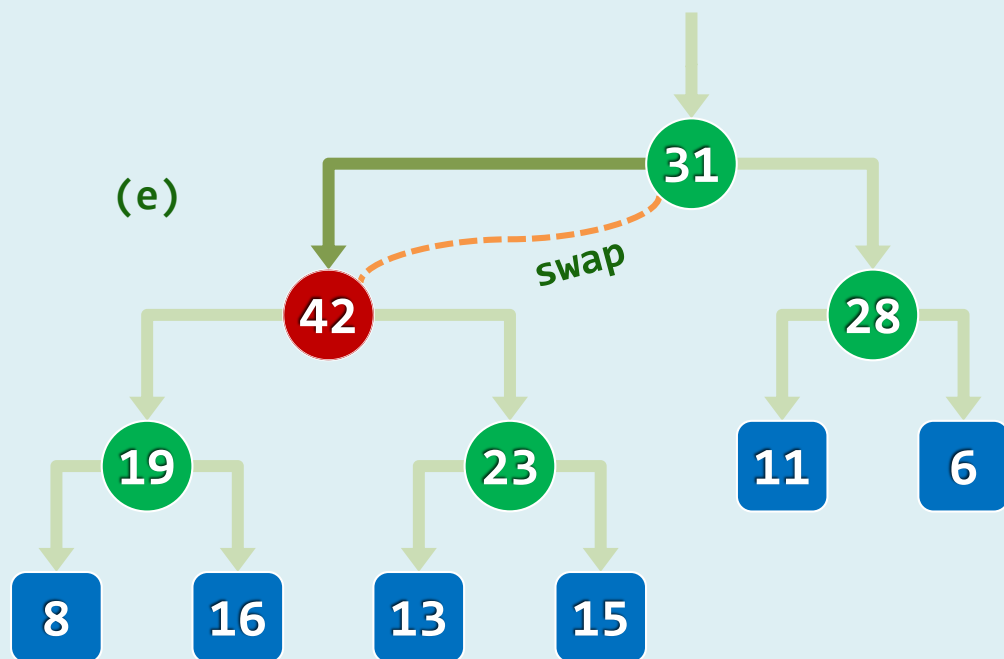
实例 (2/5)



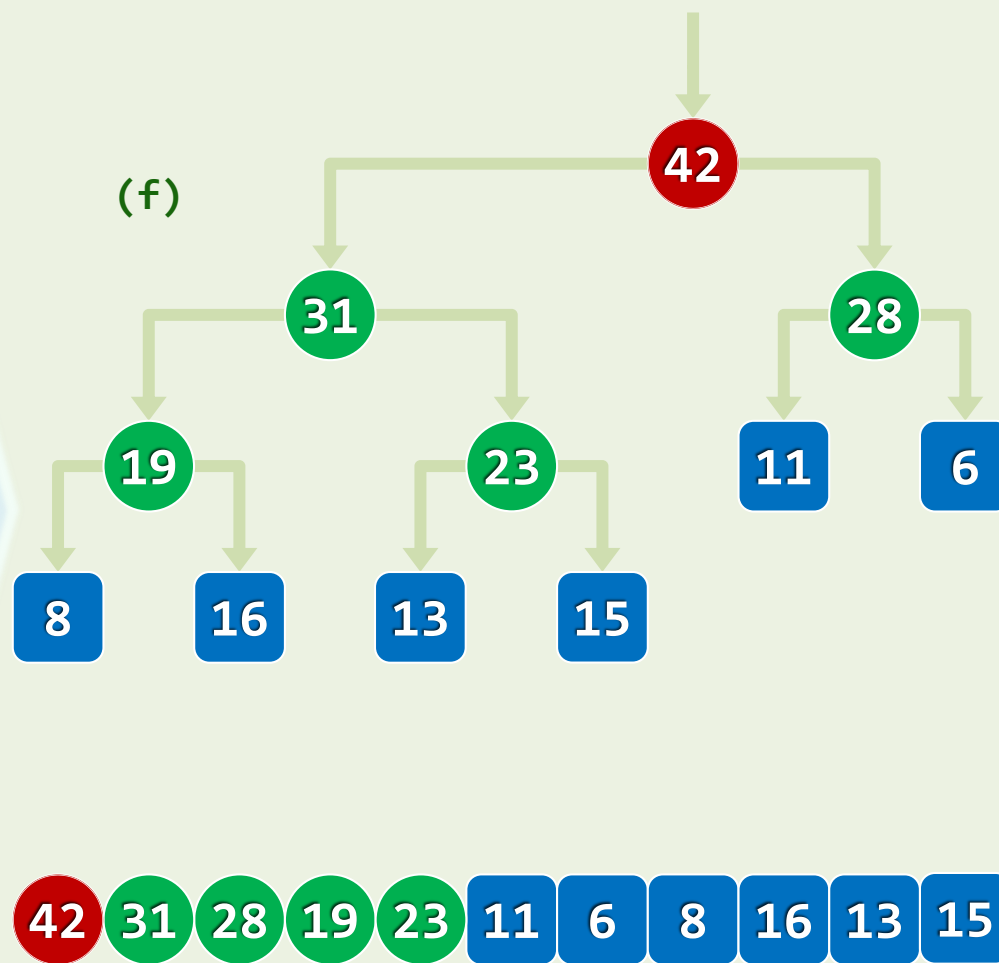
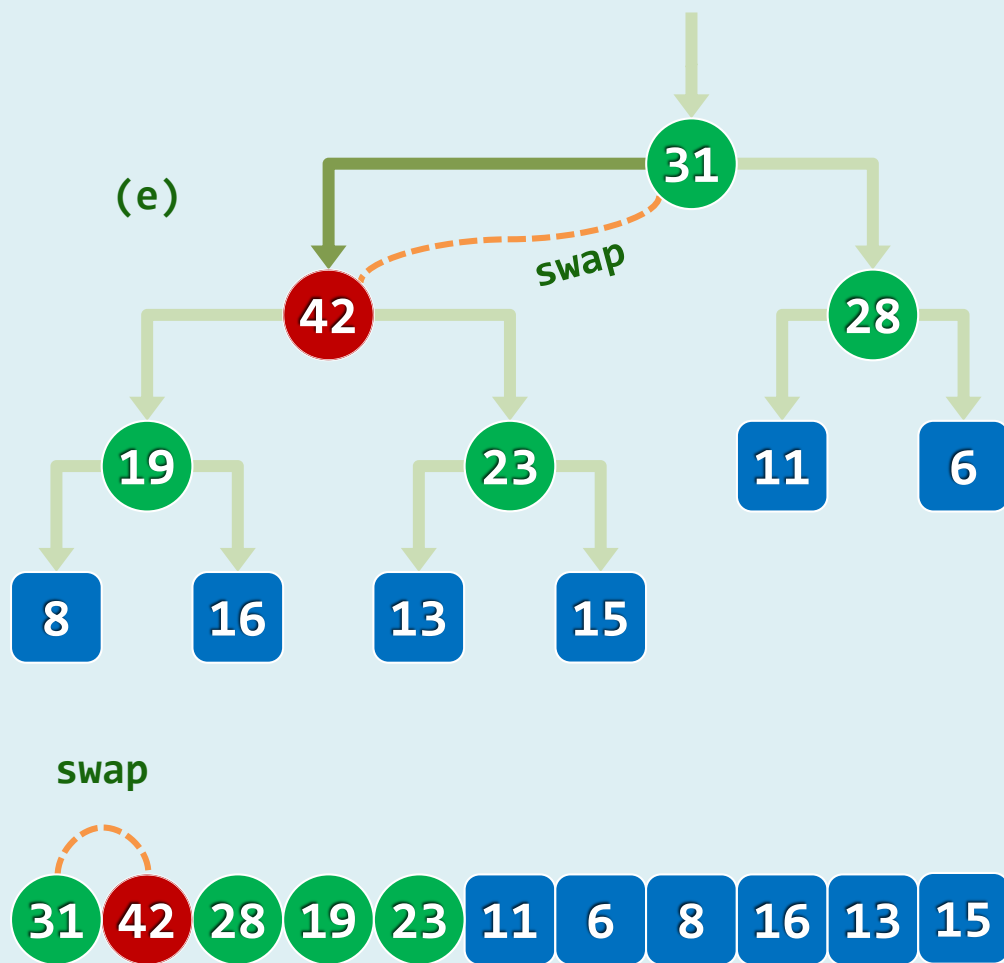
实例 (3/5)



实例 (4/5)



实例 (5/5)



实现

```
❖ template <typename T> void PQ_ComplHeap<T>::insert( T e ) //插入
    { Vector<T>::insert( e ); percolateUp( _elem, _size - 1 ); } //此insert()非彼

❖ template <typename T> Rank percolateUp( T* A, Rank i ) { //0 <= i < _size
    while ( 0 < i ) { //在抵达堆顶之前，反复地
        Rank j = Parent( i ); //考查[i]之父亲[j]
        if ( lt( A[i], A[j] ) ) break; //一旦父子顺序，上滤旋即完成；否则
        swap( A[i], A[j] ); i = j; //父子换位，并继续考查上一层
    } //while
    return i; //返回上滤最终抵达的位置
}
```


效率

- ❖ e在上滤过程中，只可能与祖先们交换
 - ❖ 完全树必平衡，e的祖先不超过 $O(\log n)$ 个
 - ❖ 故知插入操作可在 $O(\log n)$ 时间内完成
 - ❖ 然而就数学期望而言
- 实际效率往往远远更高...

