12-B3

优先级队列

完全二叉堆：删除

I have scaled the peak and

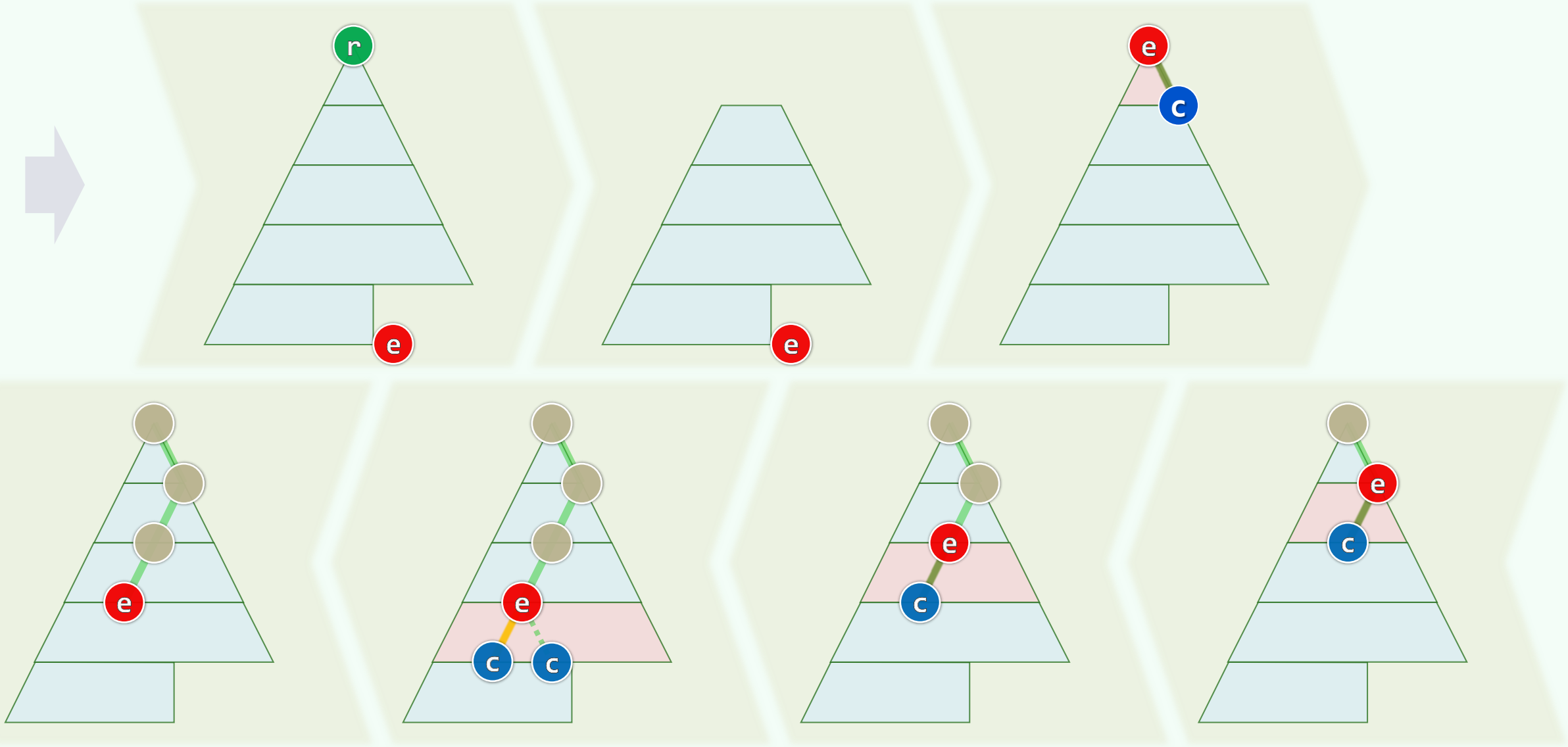
found no shelter in

fame's bleak and barren height.

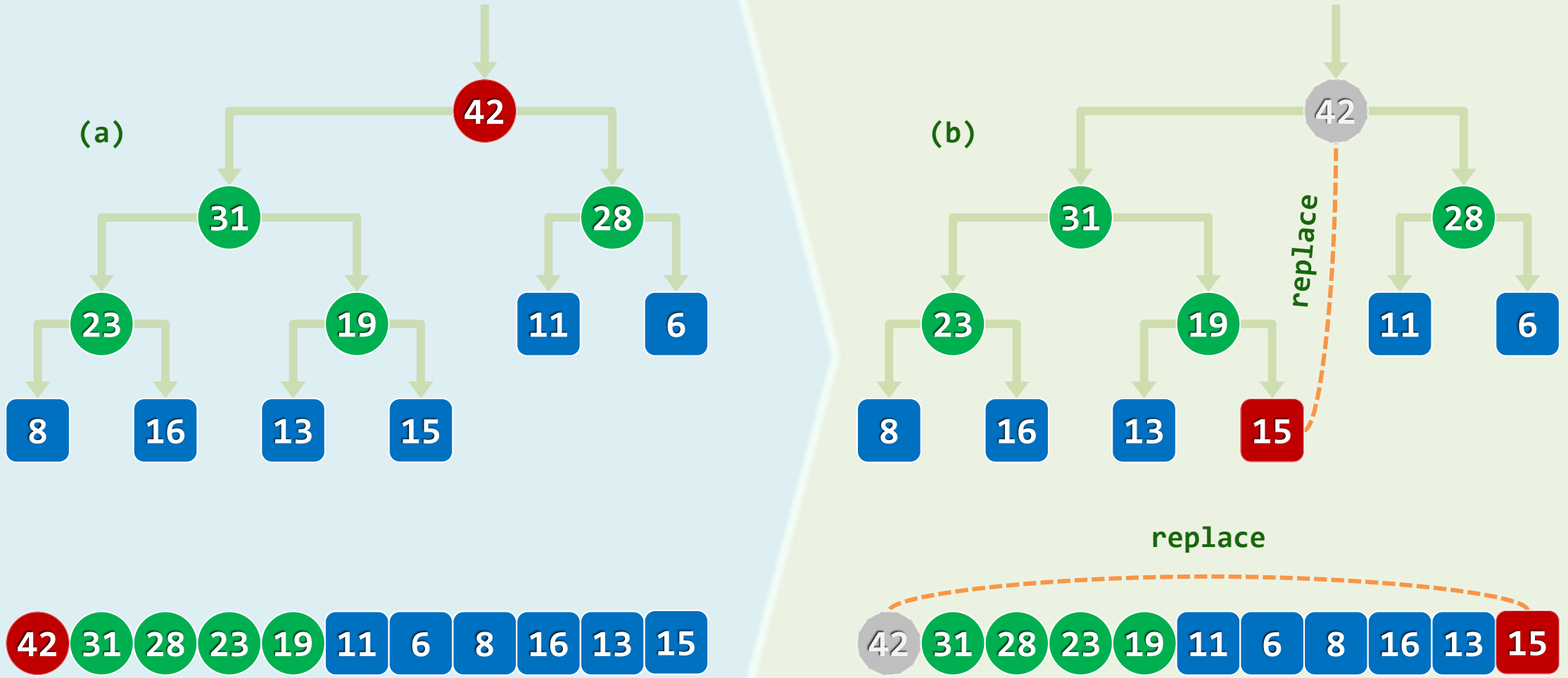
邓俊辉

deng@tsinghua.edu.cn

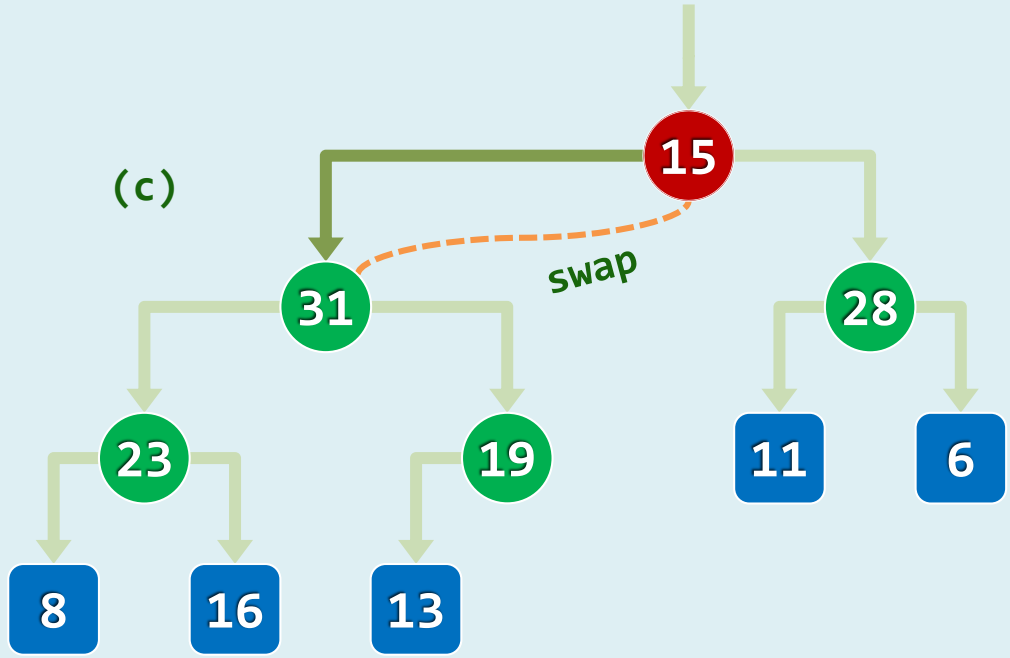
算法：割肉补疮 + 逐层下滤



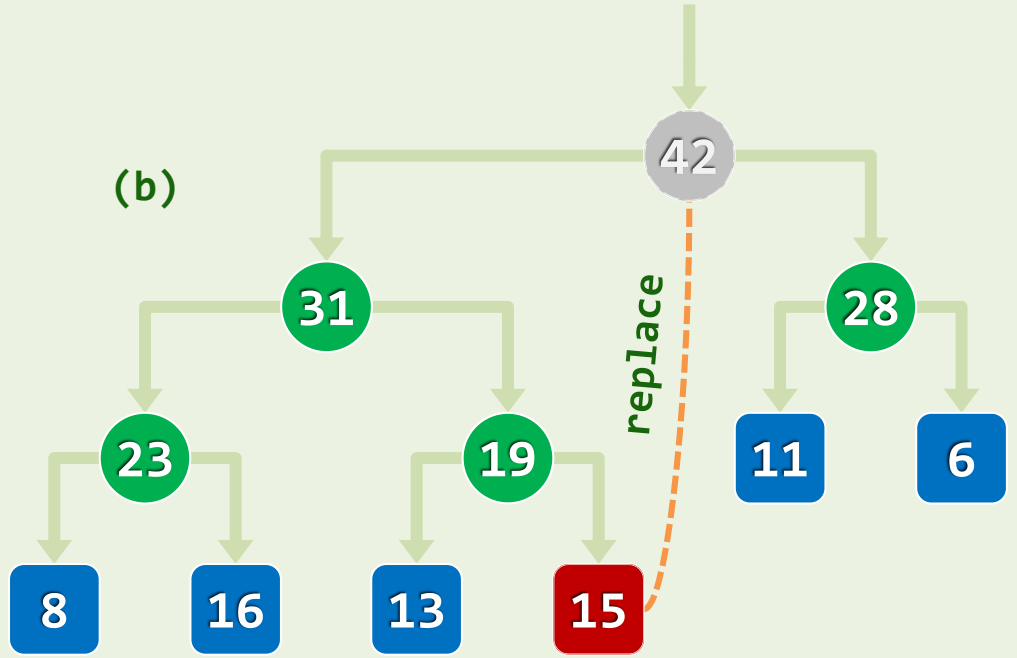
实例（1/5）


(a)
42
31
28
23
19
11
6
8
16
13
15
42 31 28 23 19 11 6 8 16 13 15
(b)
42
31
28
23
19
11
6
8
16
13
15
replace
replace
42 31 28 23 19 11 6 8 16 13 15

实例（2/5）

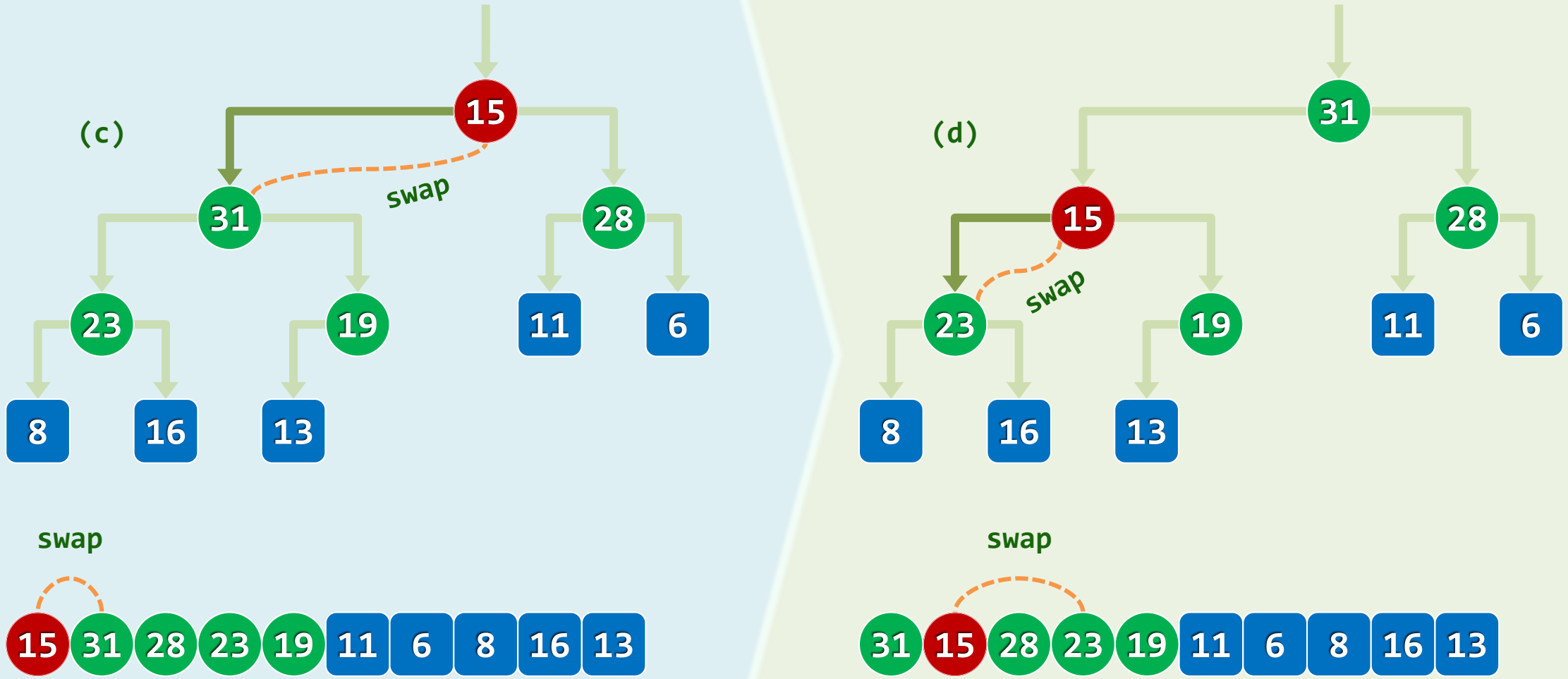

(c)
15
swap
31
28
23
19
11
6
8
16
13


swap
15
31
28
23
19
11
6
8
16
13

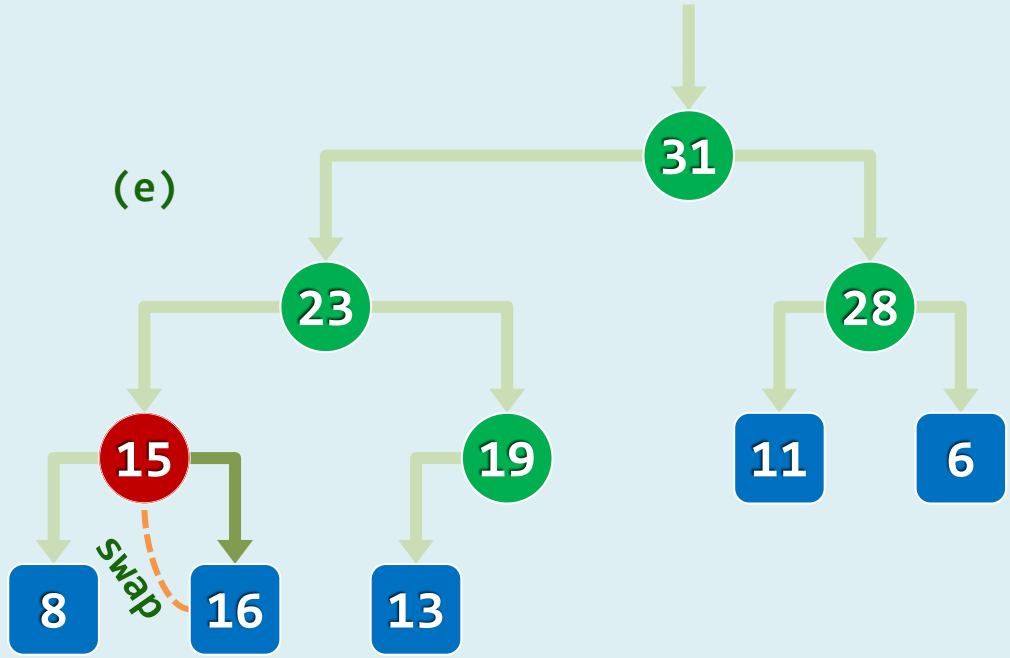

(b)
42
31
28
23
19
replace
11
6
8
16
13
15


replace
42
31
28
23
19
11
6
8
16
13
15

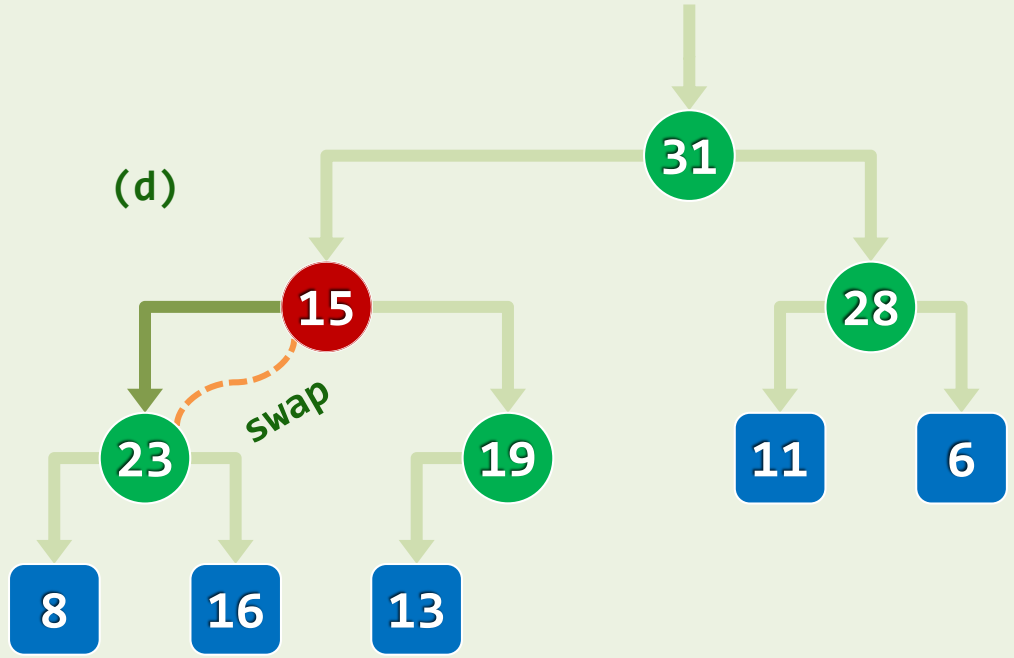
实例（3/5）

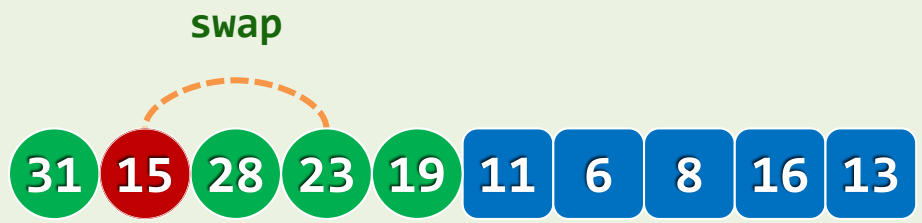

(c)
15
31
28
23
19
11
6
8
16
13
swap
swap
15
31
28
23
19
11
6
8
16
13
(d)
31
15
28
23
19
11
6
8
16
13
swap
swap
31
15
28
23
19
11
6
8
16
13

实例（4/5）

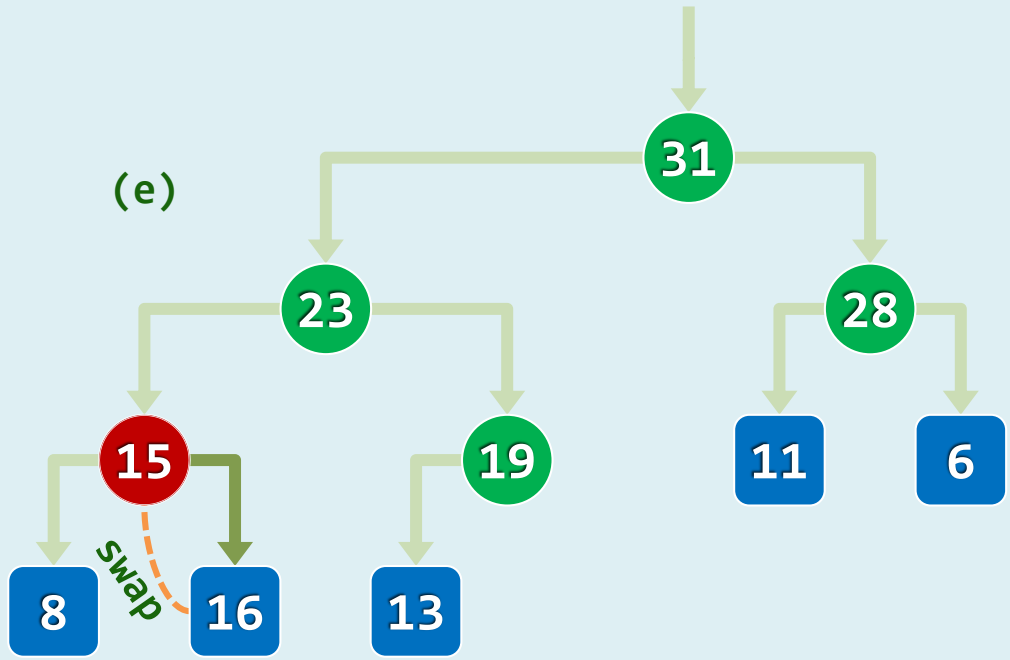

(e)
31
23
28
15
19
11
6
8
swap
16
13


swap
31
23
28
15
19
11
6
8
16
13

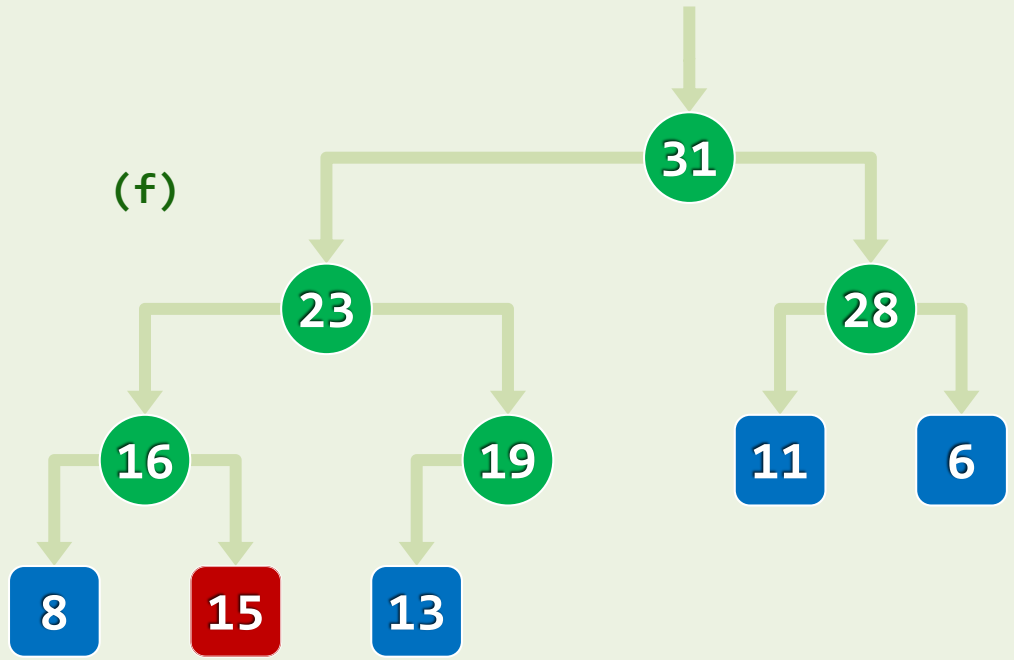

(d)
31
15
28
swap
23
19
11
6
8
16
13


swap
31
15
28
23
19
11
6
8
16
13

实例（5/5）


(e)
31
23
28
15
19
11
6
8
swap
16
13


swap
31
23
28
15
19
11
6
8
16
13


(f)
31
23
28
16
19
11
6
8
15
13


31
23
28
16
19
11
6
8
15
13

实现

```
❖ template <typename T> T PQ_ComplHeap<T>::delMax() { //删除
    T maxElem = _elem[0]; _elem[0] = _elem[ --_size ]; //摘除堆顶，代之以末词条
    percolateDown( _elem, _size, 0 ); //对新堆顶实施下滤
    return maxElem; //返回此前备份的最大词条
  }
❖ template <typename T> Rank percolateDown( T* A, Rank n, Rank i ) { //0 <= i < n
    Rank j; //i及其（至多两个）孩子中，堪为父者
    while ( i != ( j = ProperParent( A, n, i ) ) ) //只要i非j，则
       { swap( A[i], A[j] ); i = j; } //换位，并继续考察i
    return i; //返回下滤抵达的位置（亦i亦j）
  }
```

效率

- e在每一高度至多交换一次

  累计交换不超过$O(\log n)$次

- 通过下滤，可在$O(\log n)$时间内
  - 删除堆顶节点，并
  - 整体重新调整为堆

- 数学期望呢?