12-F1

优先级队列

左式堆：沿藤合并

God's right hand is gentle

But terrible is his left hand

邓俊辉

deng@tsinghua.edu.cn

堆合并

- H = merge(A, B): 将堆A和B合二为一

  //不妨设|A| = n ≥ m = |B|

- 方法一: A.insert(B.delMax())

  $O(m * (\log m + \log(n + m)))$

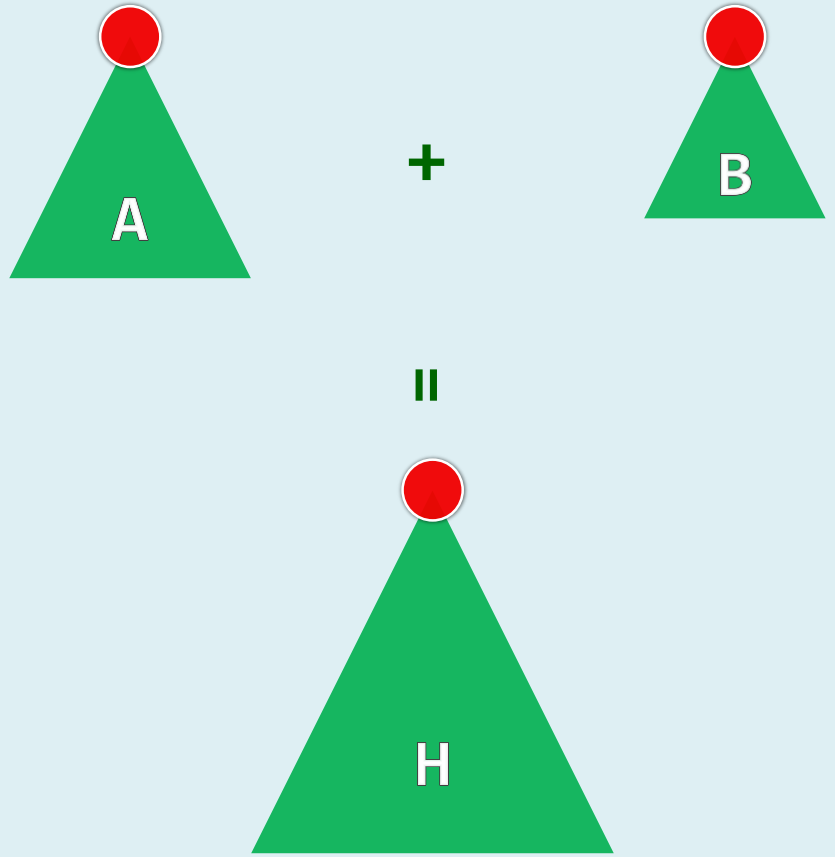
  $= O(m * \log(n + m))$

- 方法二: union(A, B).heapify(n + m)

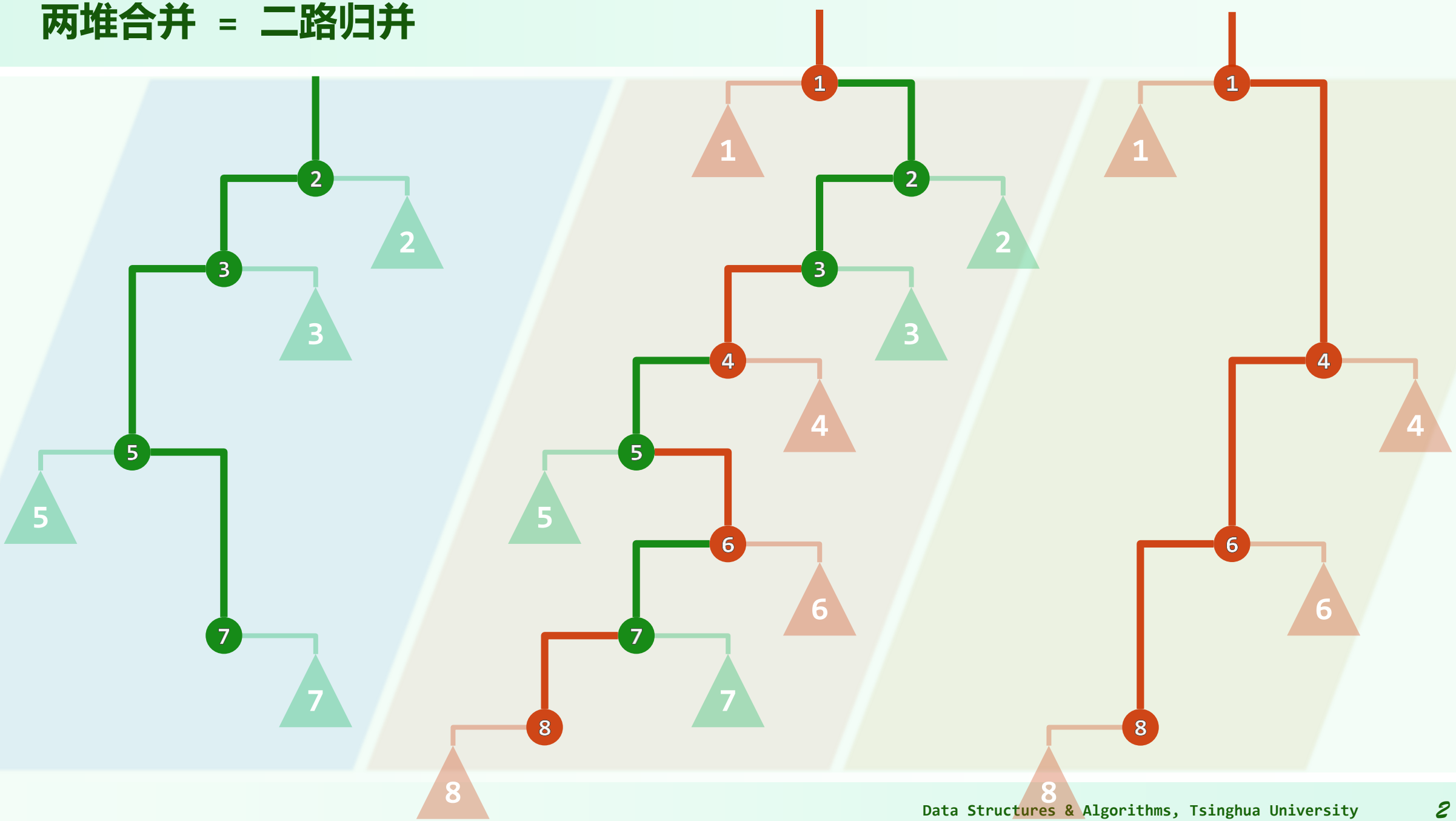
  $O(m + n)$

- 有没有更好的办法? 比如...

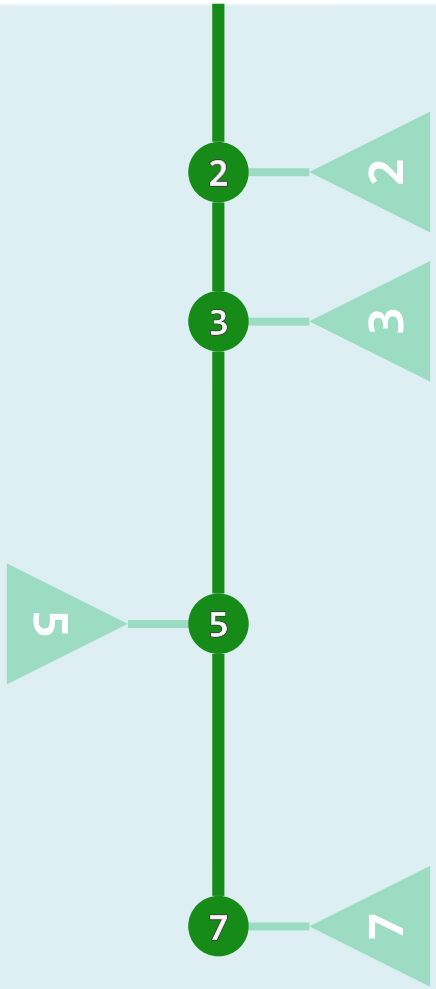
- 可否奢望在...$O(\log n)$...时间内实现merge()?

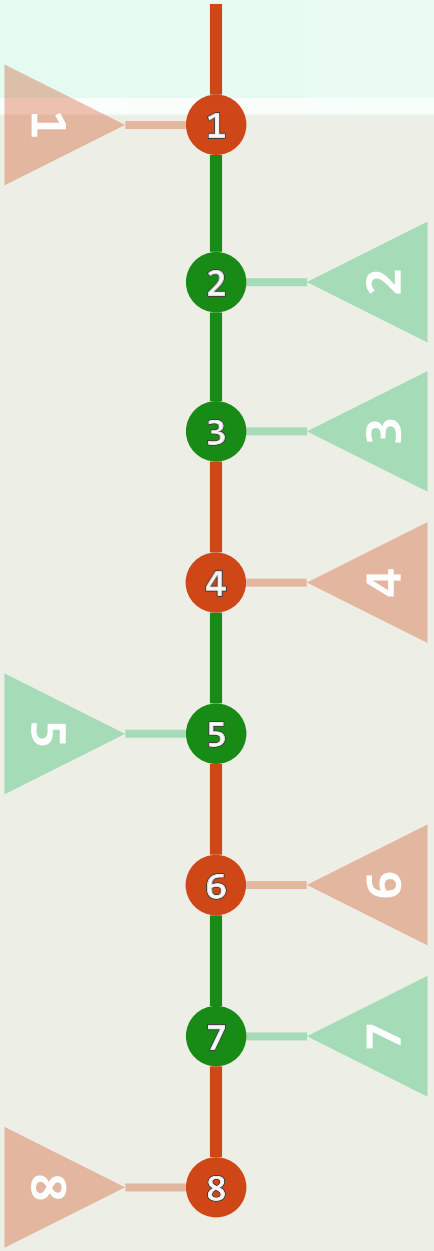


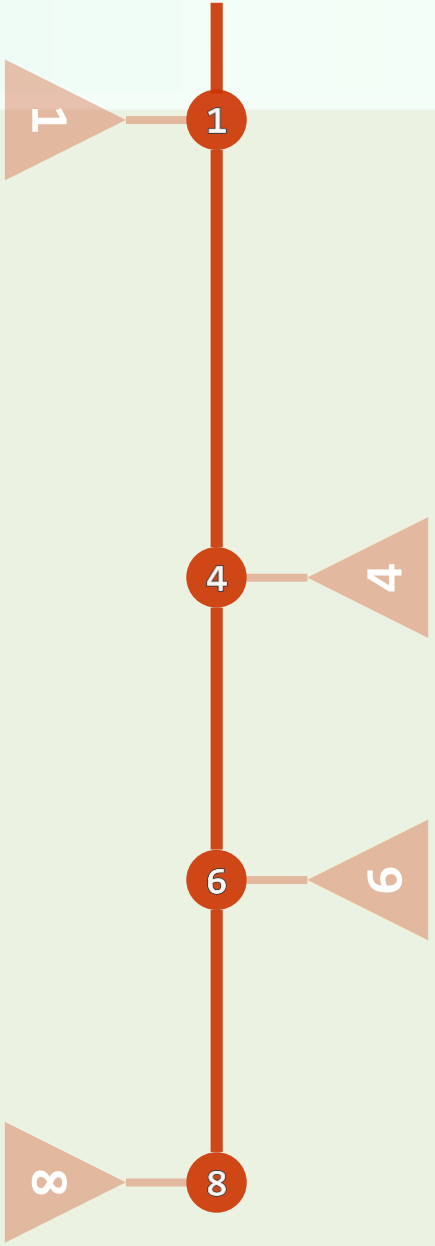
两堆合并 = 二路归并



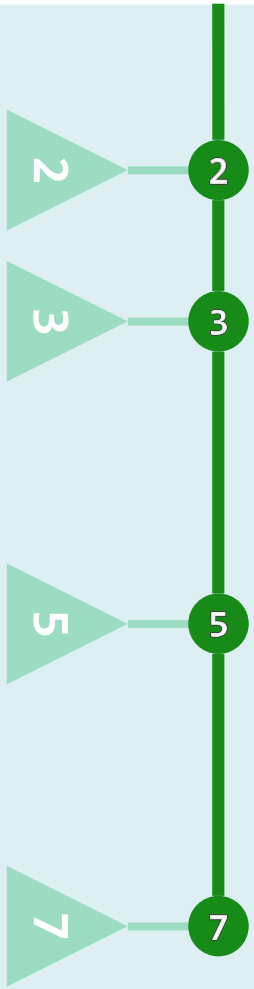
两堆合并 = 二路归并

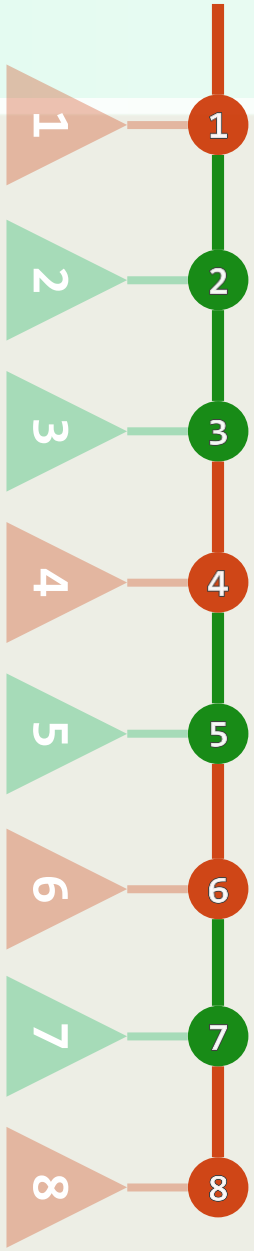


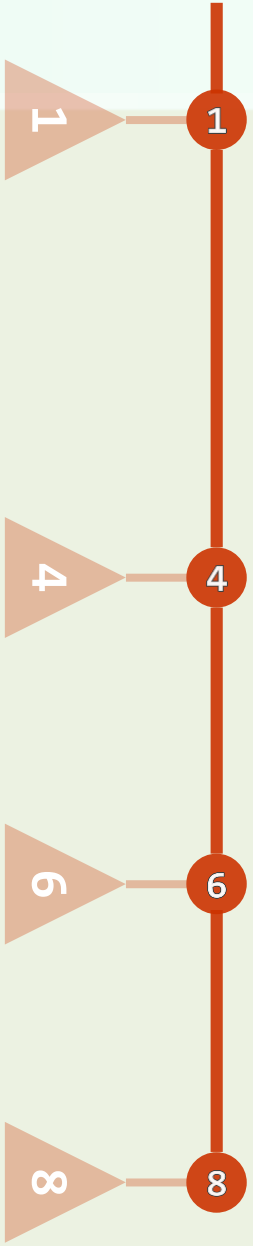




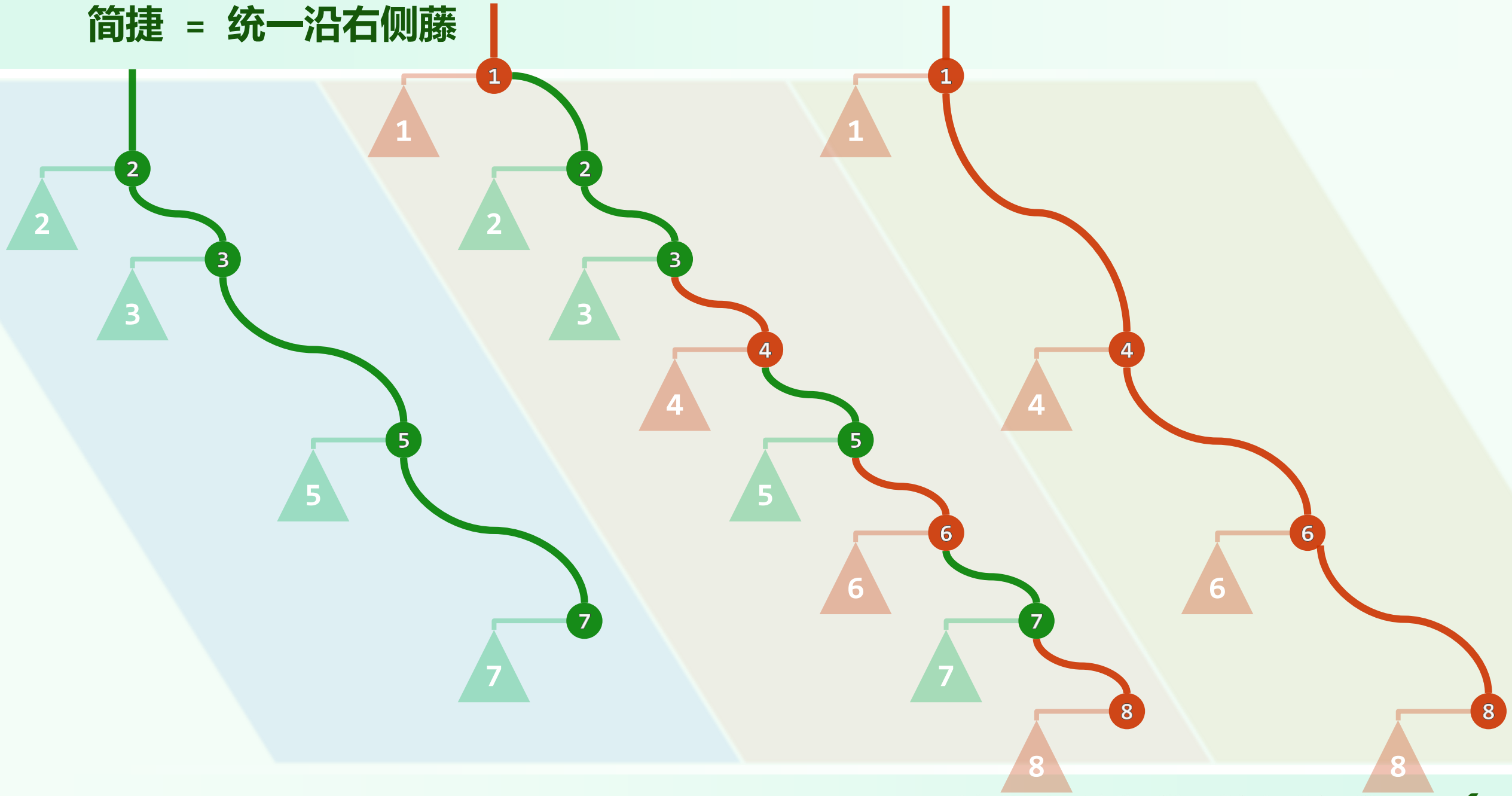
简捷 = 统一沿右侧藤







简捷 = 统一沿右侧藤



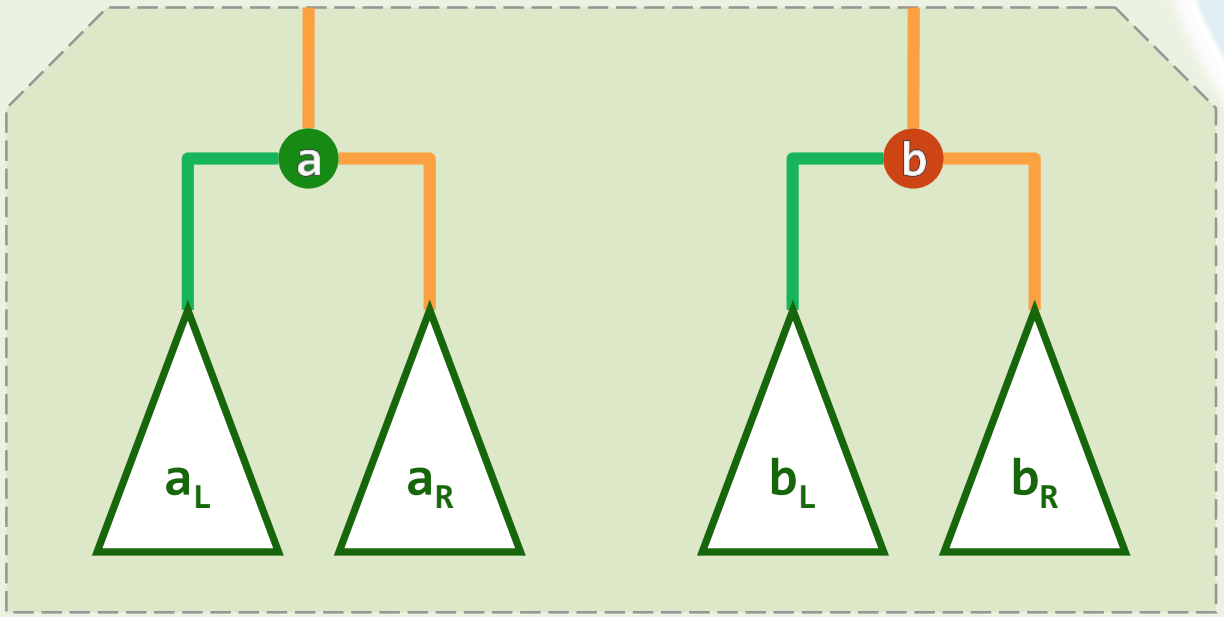
LeftHeap

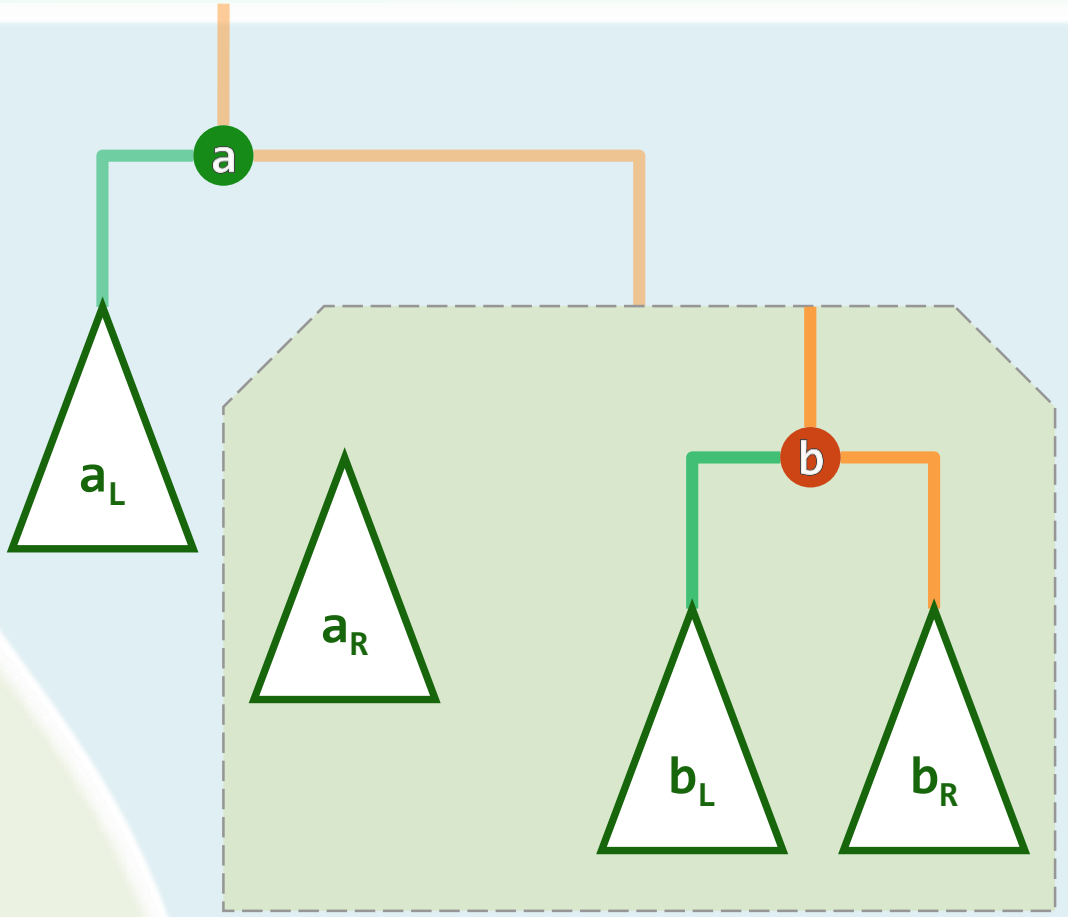
```
template <typename T> //基于二叉树，以左式堆形式实现的优先级队列
class PQ_LeftHeap : public PQ<T>, public BinTree<T> {
public:  T getMax() { return _root->data; }
         void insert(T); T delMax(); //均基于统一的合并操作实现...
         PQ_LeftHeap( PQ_LeftHeap & A, PQ_LeftHeap & B ) {
            _root = merge(A._root, B._root); _size = A._size + B._size;
            A._root = B._root = NULL; A._size = B._size = 0;
         }
};

template <typename T> BinNodePosi<T> merge(BinNodePosi<T>, BinNodePosi<T>);
```

递归实现

所需时间 ∝ 右侧藤总长

$\cancel{\mathcal{O}(n)}$ $\mathcal{O}(\log n)$





如何...控制藤长以...持续合并?