

12-D1

优先级队列

锦标赛树：胜者树

老妖道：“怎么叫做分瓣梅花计？”

小妖道：“如今把洞中大小群妖，点将起来，千中选百，百中选十，十中只选三个...”

邓俊辉

deng@tsinghua.edu.cn

锦标赛树/Tournament Tree

❖ 完全二叉树

- 叶节点：待排序元素（选手）
- 内部节点：孩子中的胜者

❖ `create()` $O(n)$

`remove()` $O(\log n)$

`insert()` $O(\log n)$

❖ **树根总是全局冠军**：类似于堆顶

❖ 内部结点各对应于一场比赛的胜者——重复存储

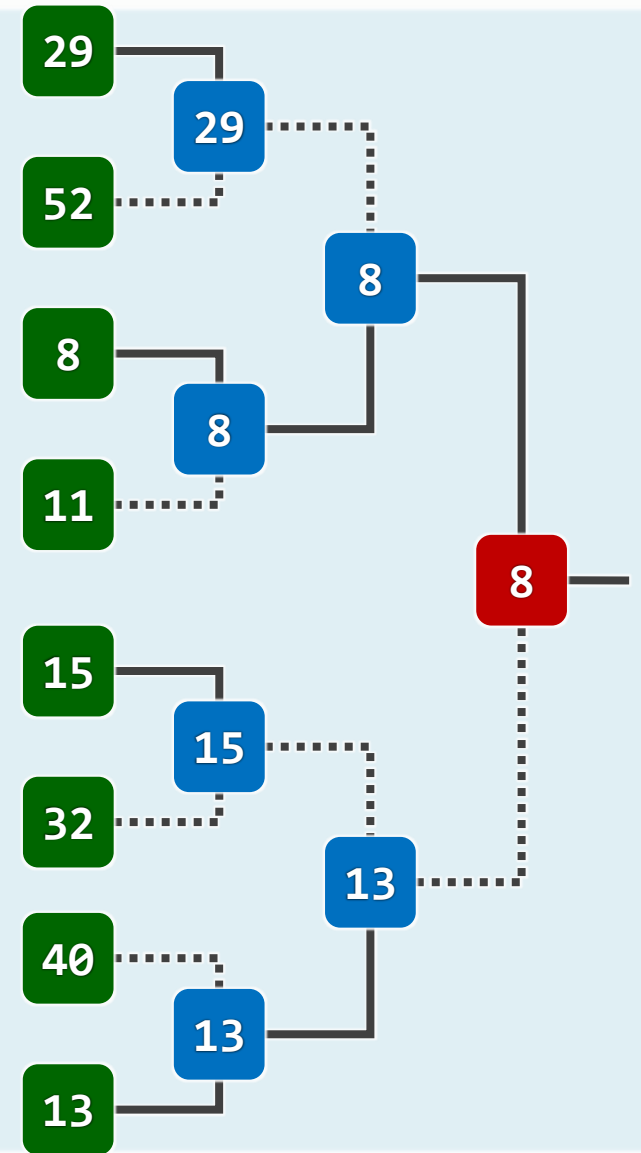


Tournamentsort()

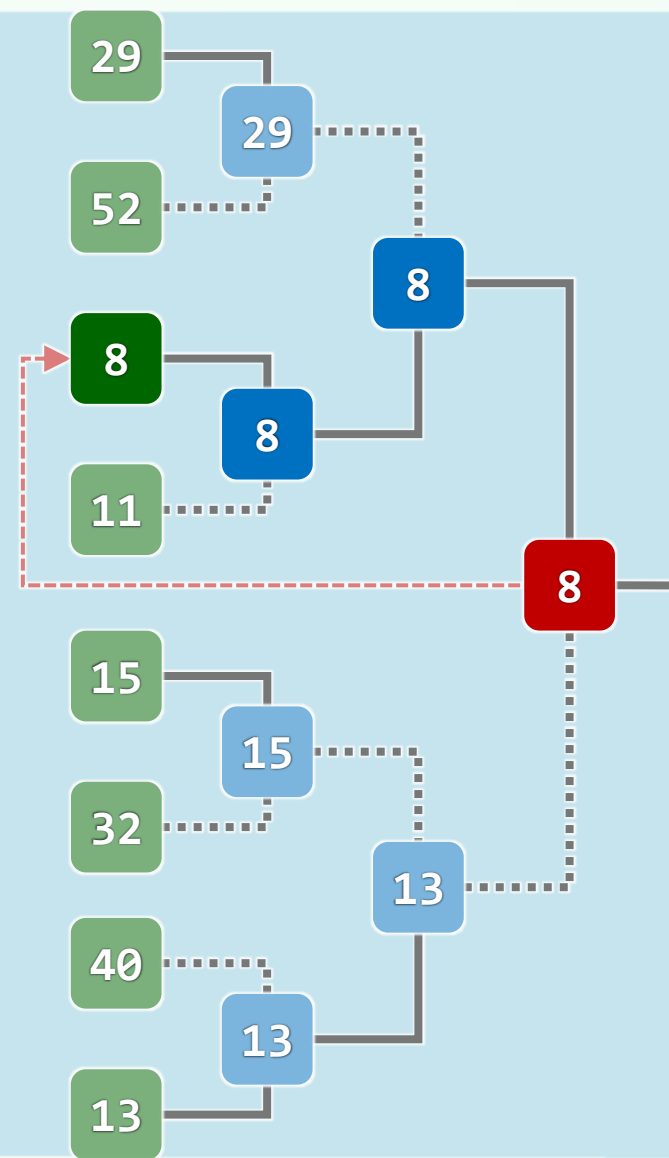
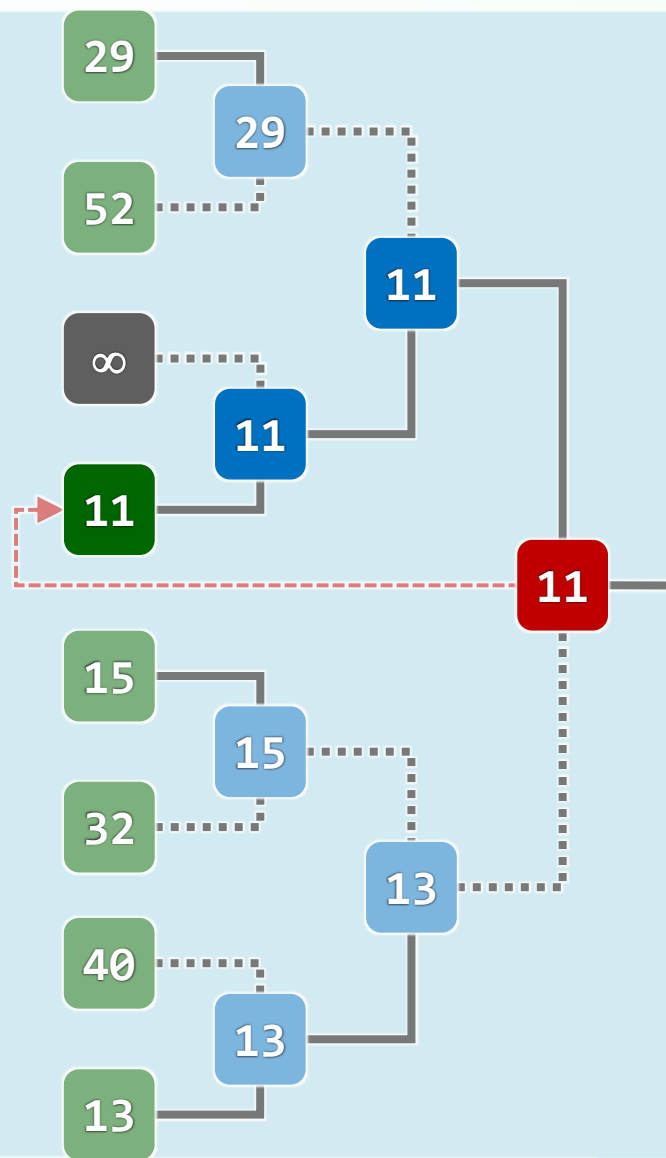
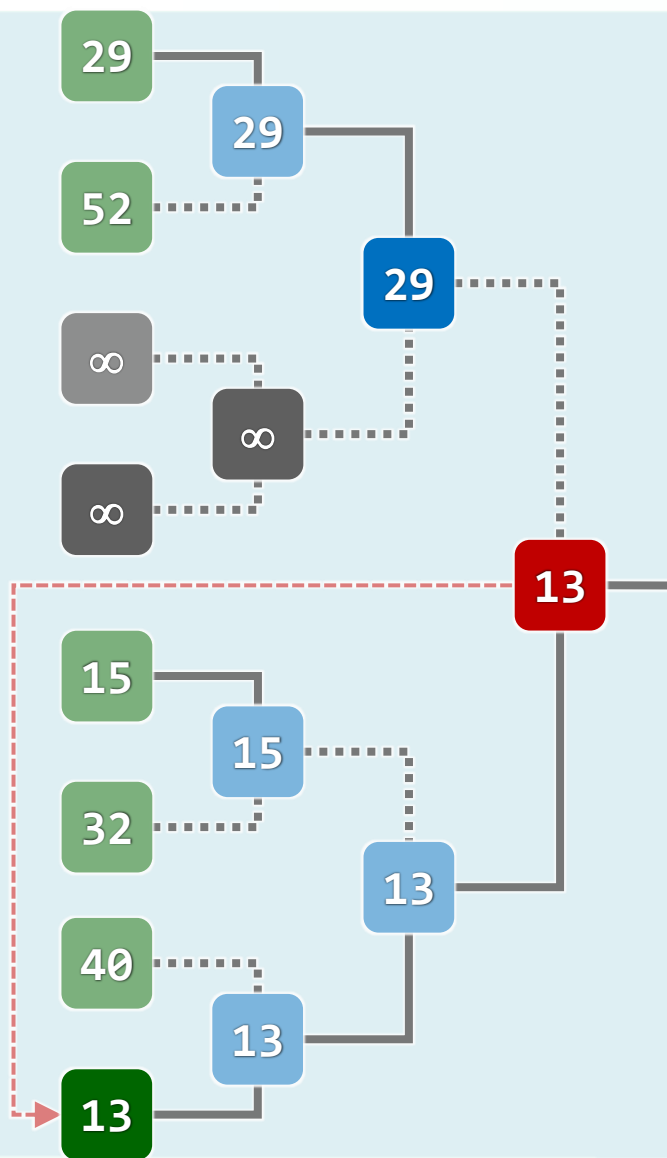
CREATE a tournament tree for the input list

while there are active leaves

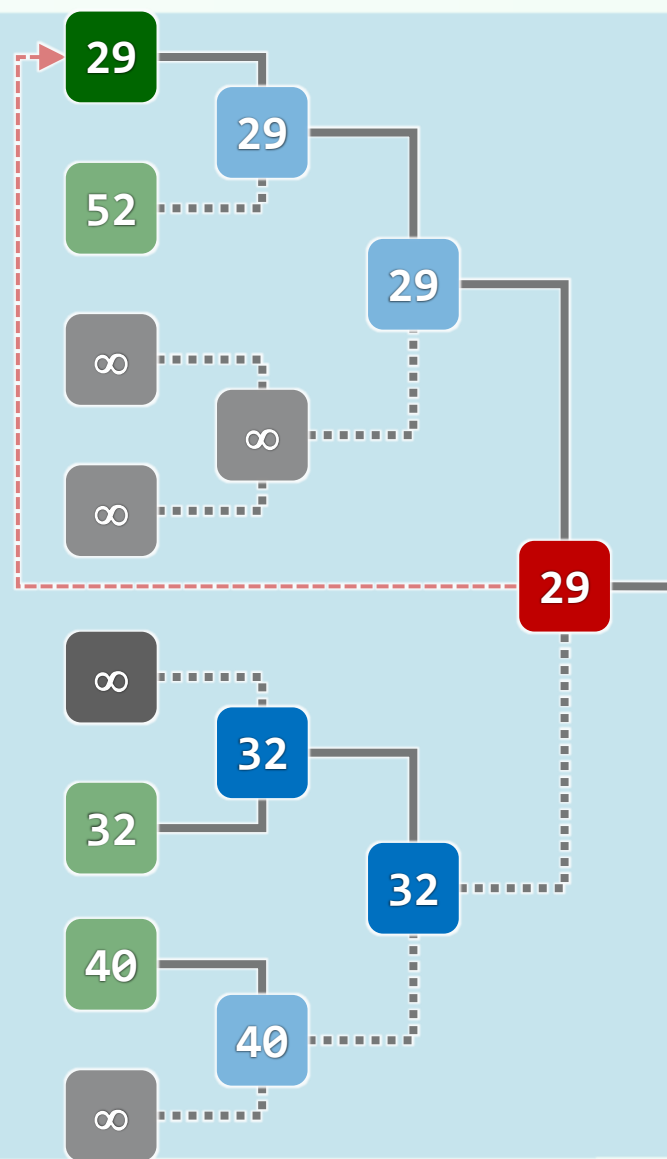
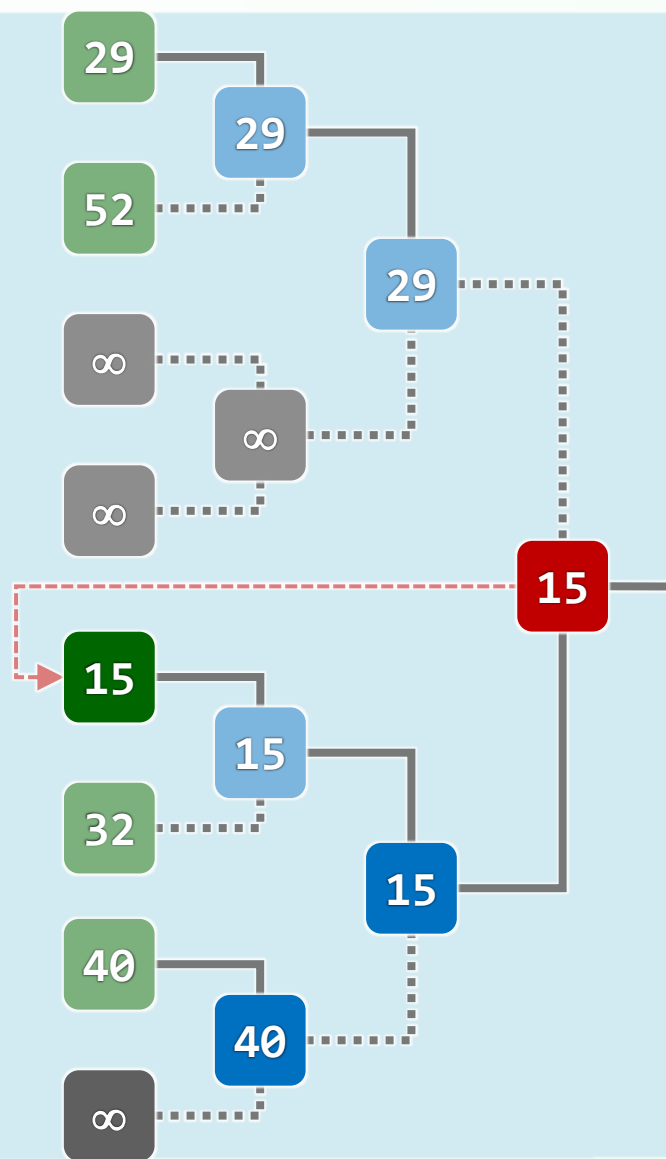
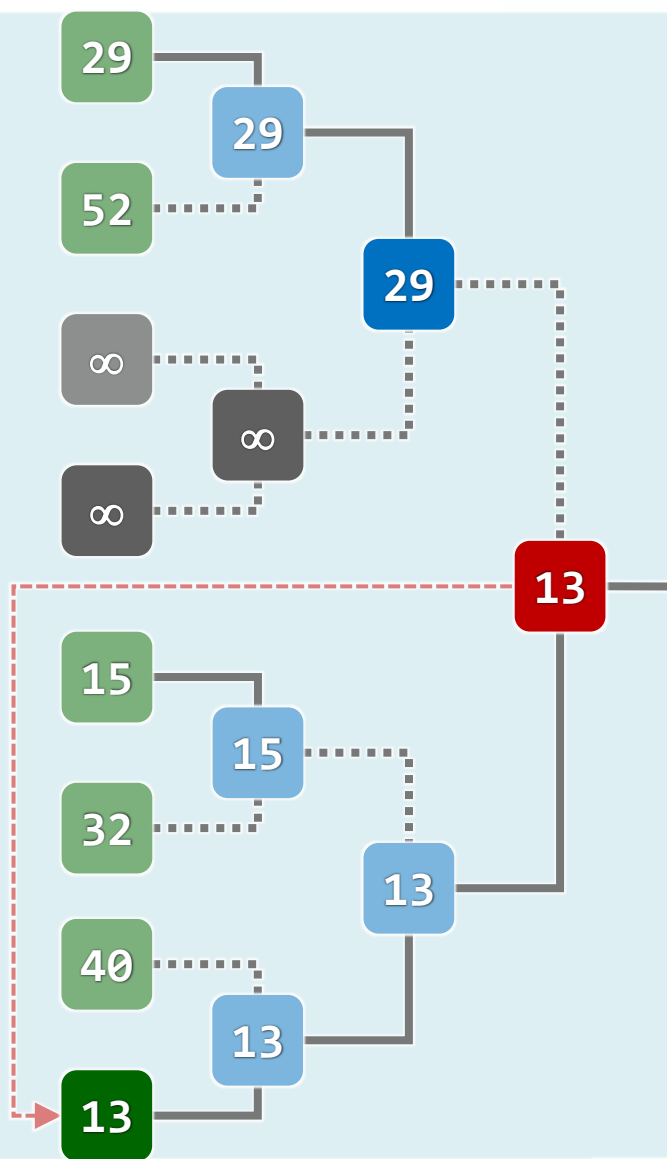
- **REMOVE** the root
- **RETRACE** the root down to its leaf
- **DEACTIVATE** the leaf
- **REPLAY** along the path back to the root



实例 (1/2)



实例 (2/2)



效率

❖ 空间: $O(\text{节点数}) = O(\text{叶节点数}) = O(n)$

❖ 构造: 仅需 $O(n)$ 时间

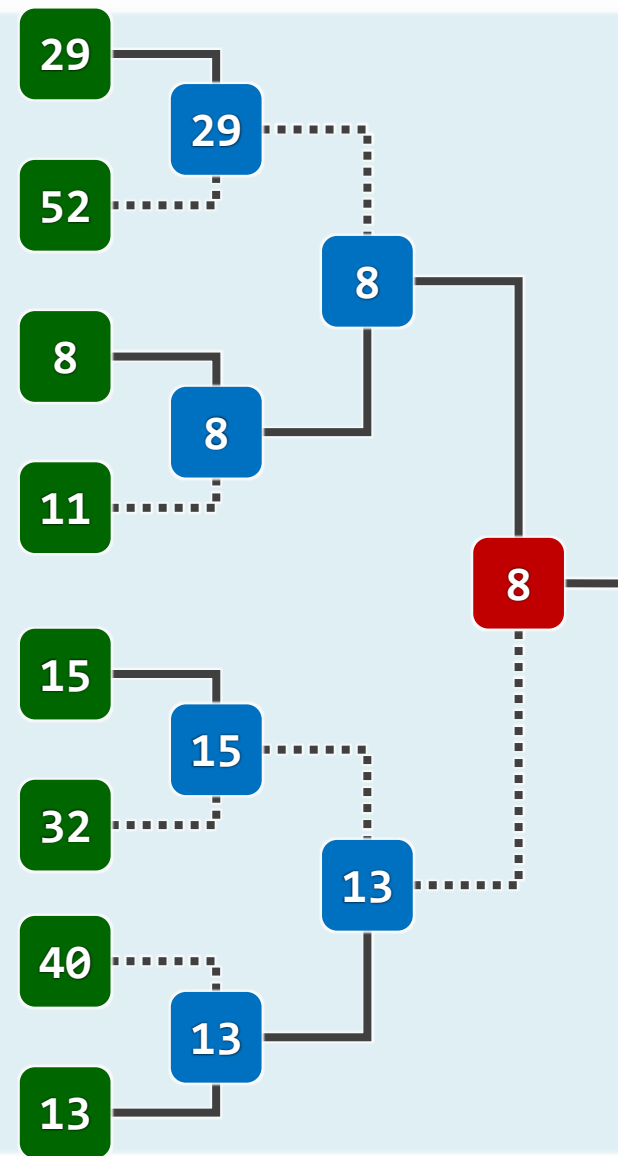
❖ 更新: 每次都须全体重赛`replay`?

唯上一优胜者的**祖先**, 才有必要!

❖ 为此 只需从其所在叶节点出发, 逐层上溯直到树根

如此 为确定各轮优胜者, 总共所需时间仅 $O(\log n)$

❖ 时间: $n \text{ 轮} \times O(\log n) = O(n \log n)$, 达到下界



选取

❖ 借助锦标赛树，从 n 个元素中找出最小的 k 个， $k \ll n$

- 初始化: $O(n)$ // $n-1$ 次比较
- 迭代 k 步: $O(k \cdot \log n)$ // 每步 $\log n$ 次比较

与小顶堆旗鼓相当？

❖ 渐近意义上，的确如此

但就常数而言，区别不小...

❖ Floyd算法、delMax()中的percolateDown()

在每一层需做2次比较，累计大致 $2 \cdot \log n$ 次

