

10. 优先级队列

(d) 锦标赛排序

老妖道：“怎么叫做分辨梅花计？”

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

胜负判定原则 : **小者为胜**

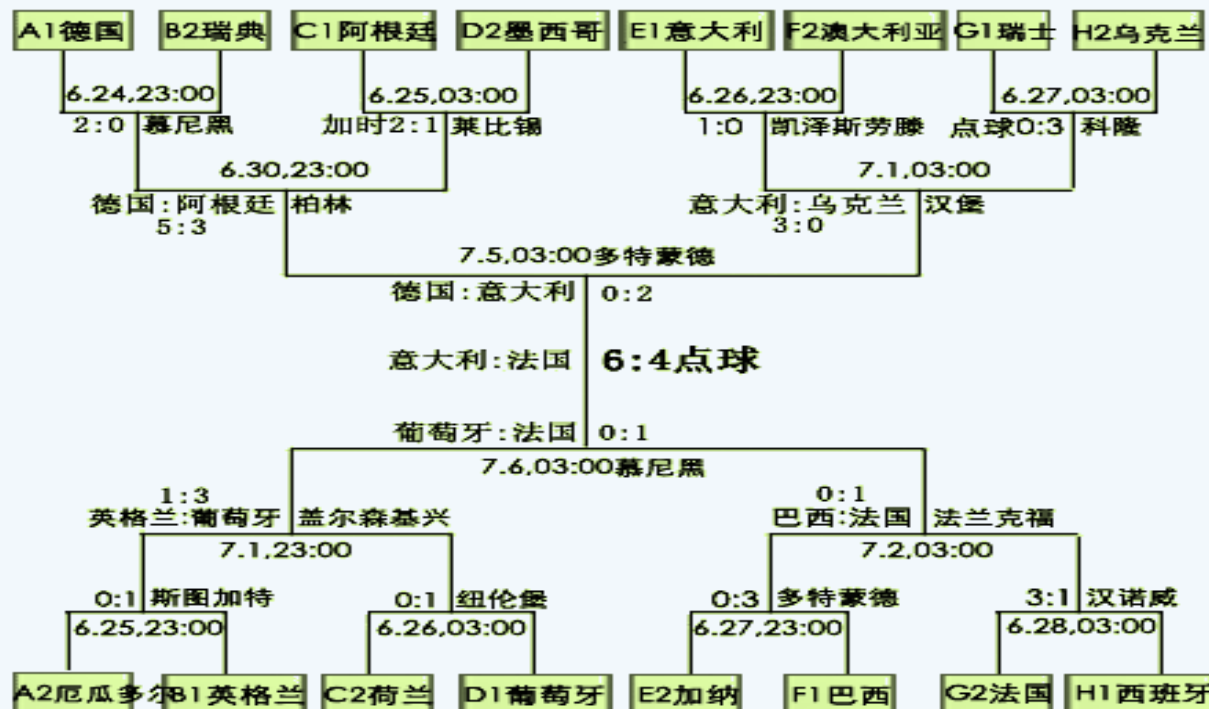
❖ **create()** $O(n)$

remove() $O(\log n)$

insert() $O(\log n)$

❖ **树根** 总是全局 **冠军** —— 类似于 **最小堆**

有重复 : 在任一子树中, 从 **根** 通往 **优胜者** 的沿途, 所有节点都是 **优胜者**



❖ Tournamentsort()

CREATE a tournament tree for the input list

while there are active leaves

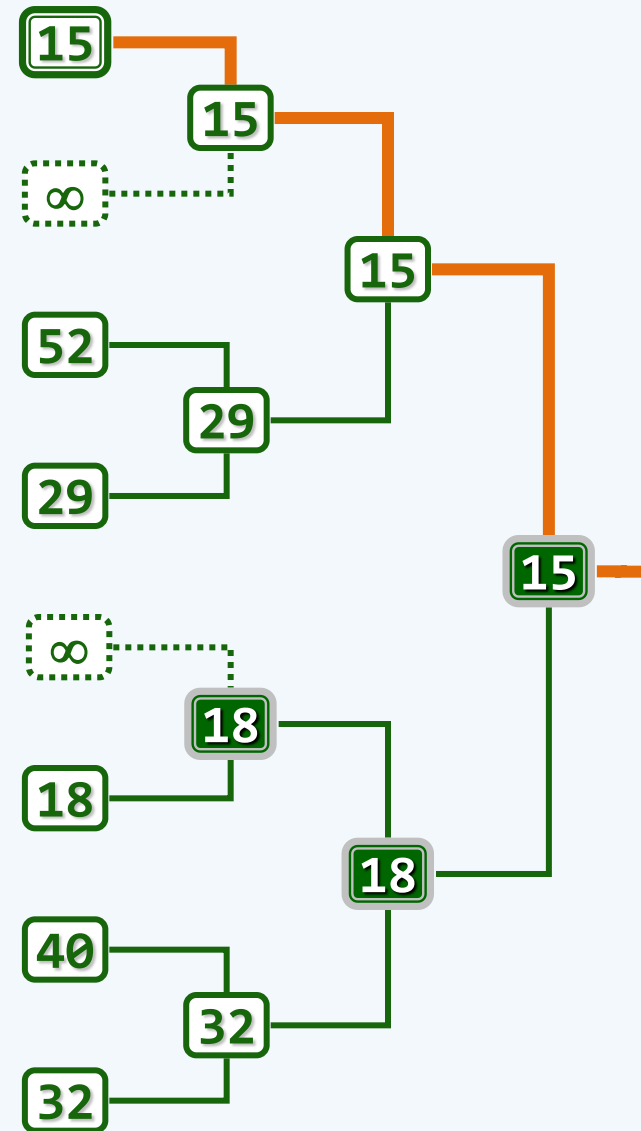
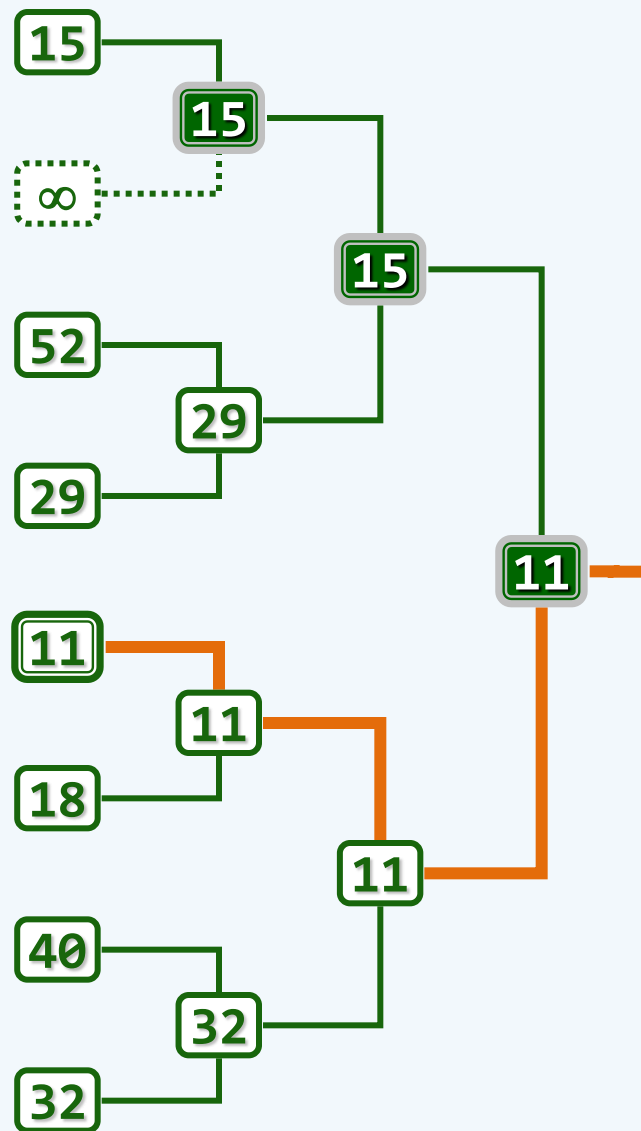
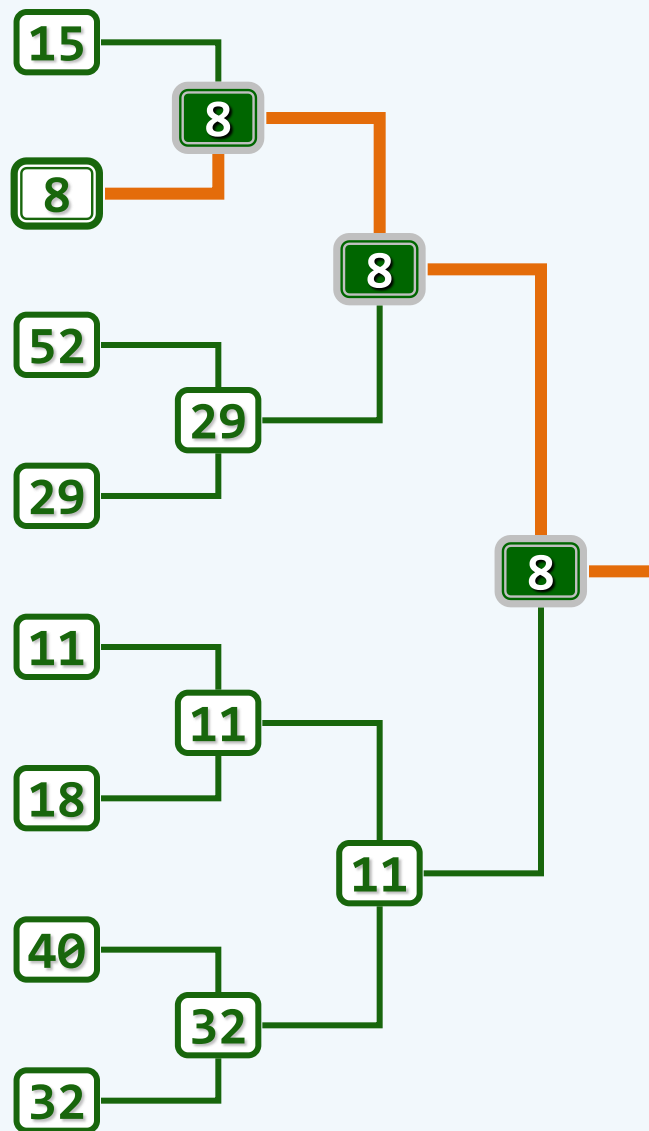
REMOVE the root

RETRACE the root down to its leaf

DEACTIVATE the leaf

REPLAY along the path back to the root

实例



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，才有必要参加！

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

选取

❖ 从 n 个元素中找出最小的 k 个, $k \ll n$ // 回忆第01章的 `max2()`, 即是 $k = 2 \ll n$

❖ 借助锦标赛树 初始化: $O(n)$ // $n - 1$ 次比较

迭代 k 步: $O(k \log n)$ // 每步 $\log n$ 次比较

与小顶堆旗鼓相当?

❖ 渐进意义上, 的确如此

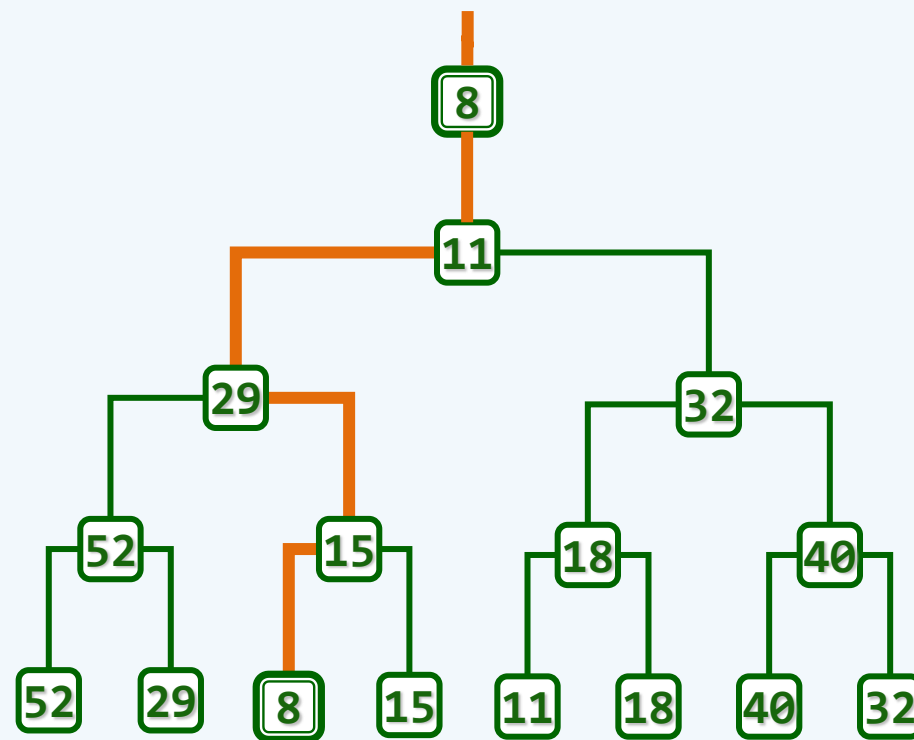
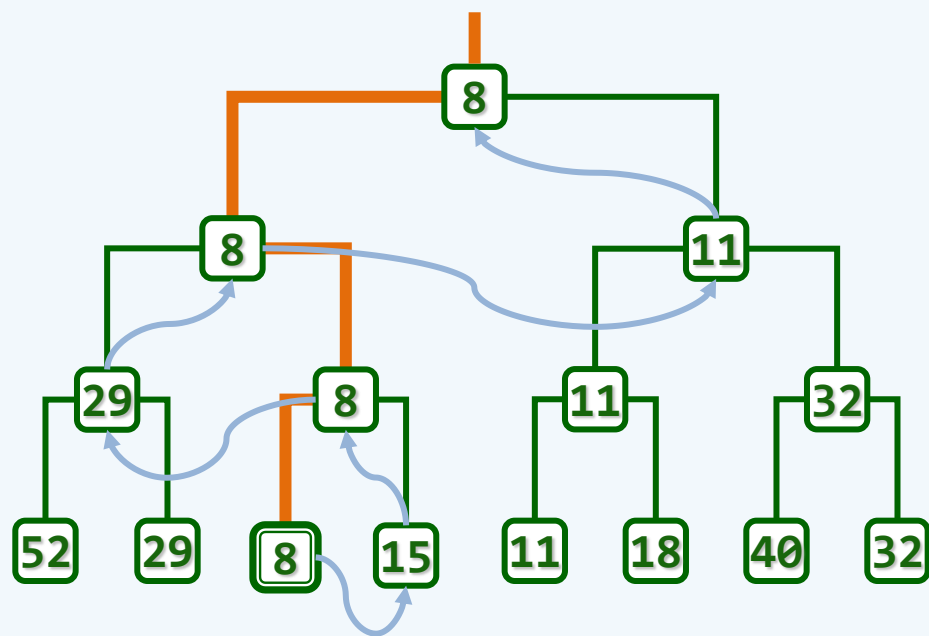
但就常数而言, 区别不小...

❖ Floyd 算法中的 `percolateDown()`, 在每一层需做 2 次比较, 累计略少于 $2n$ 次

`delMax()` 中的 `percolateDown()`, 亦是如此

败者树

- ❖ 重赛过程中，除了沿途各节点，还需间隔地访问兄弟
- ❖ loser tree：内部节点，记录对应比赛的败者
增设根的“父节点”，记录冠军



课后

- ❖ 利用锦标赛树，可否实现**稳定**的排序？
- ❖ 利用锦标赛树，如何实现**多路归并**？