12-D1

优先级队列

锦标赛排序：锦标赛树

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(logn)

insert() //O(logn)

❖ 树根总是全局冠军：若约定小者为胜，则类似于小顶堆

❖ 内部结点各对应于一场比赛的胜者——重复存储

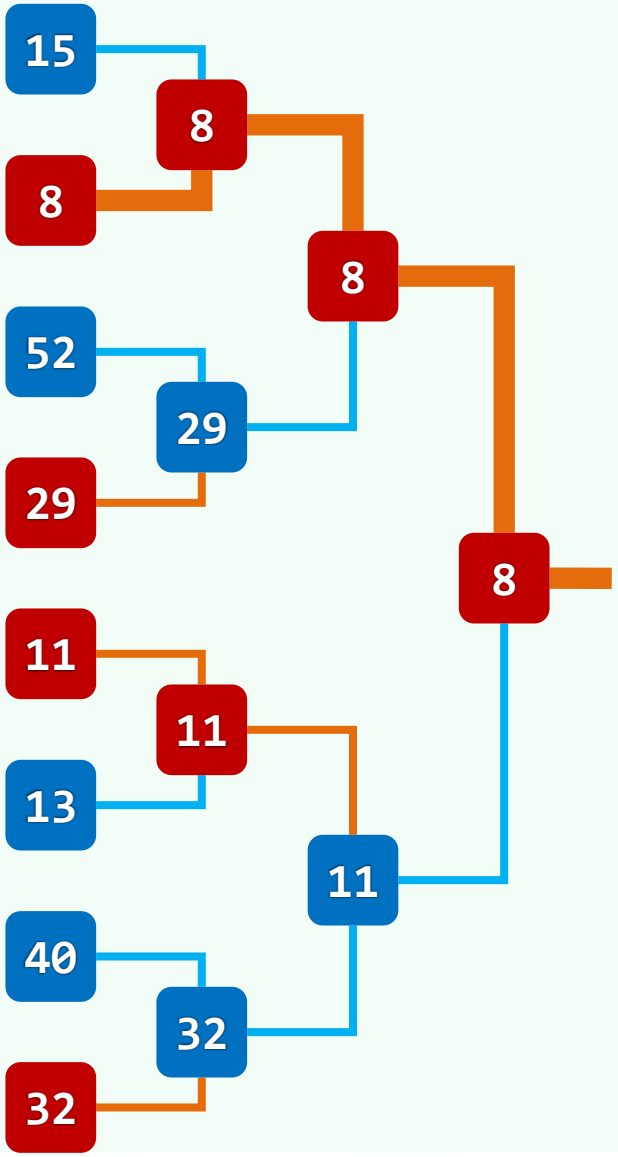
用以排序

- Tournamentsort()

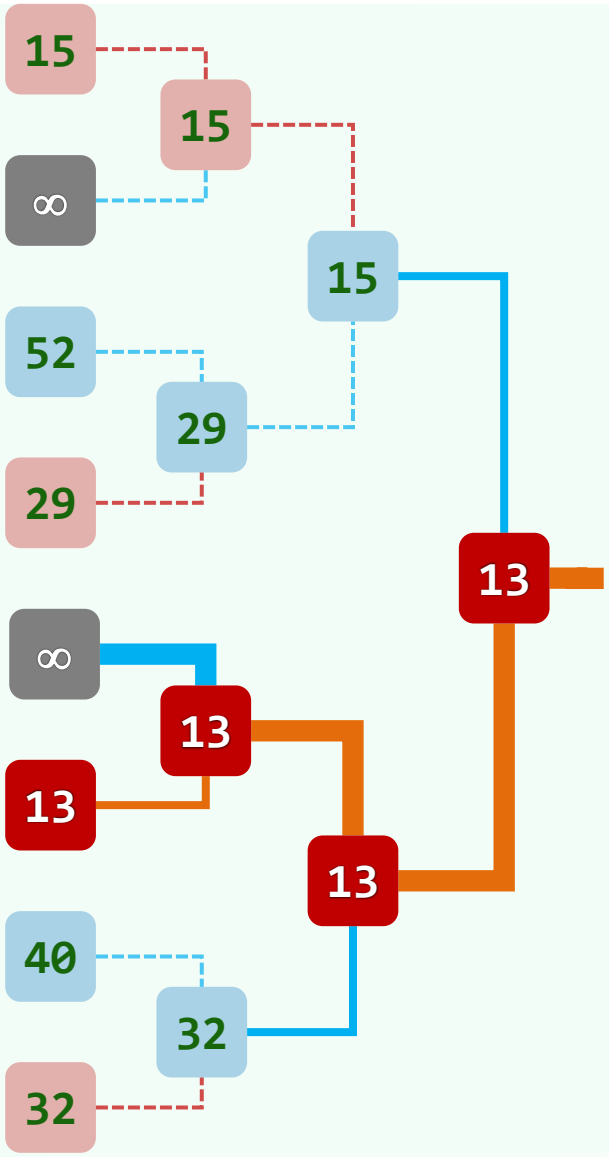
  CREATE a tournament tree for the input list

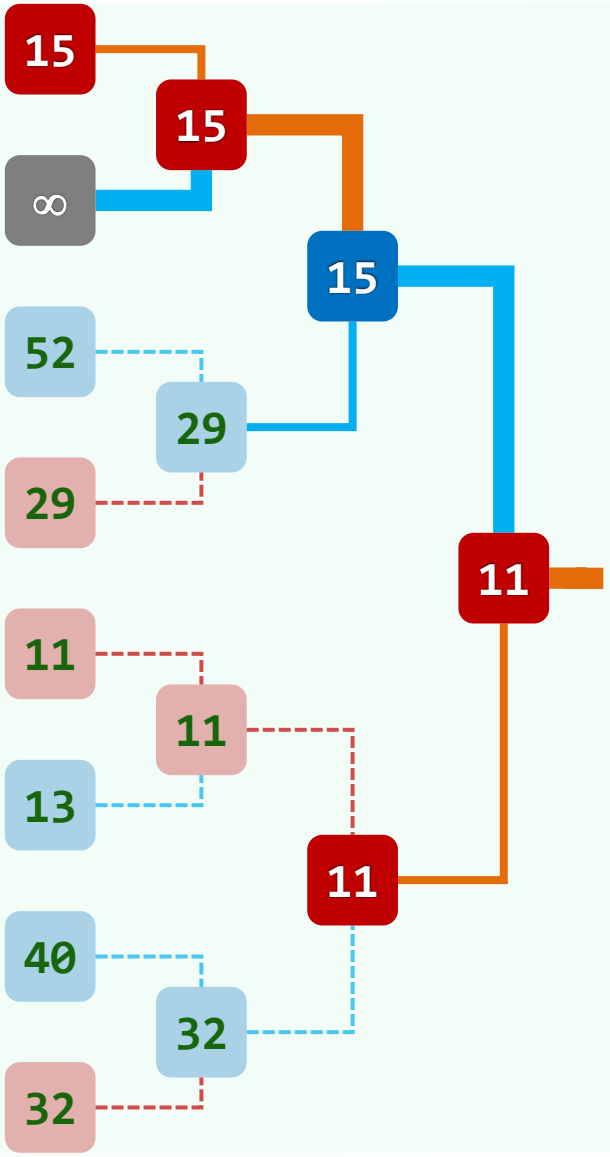
  while there are active leaves

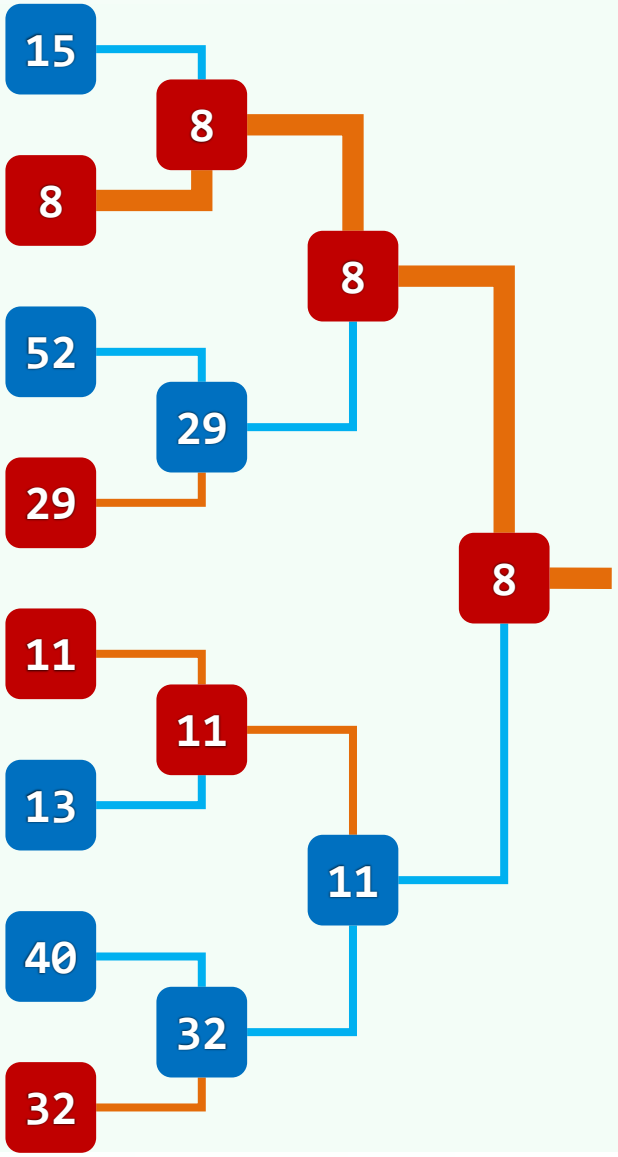
  - REMOVE the root
  - RETRACE the root down to its leaf
  - DEACTIVATE the leaf
  - REPLAY along the path back to the root



实例


15
15
∞
15
52
29
29
13
∞
13
13
13
40
32
32


15
15
∞
15
52
29
29
11
11
11
13
11
40
32
32

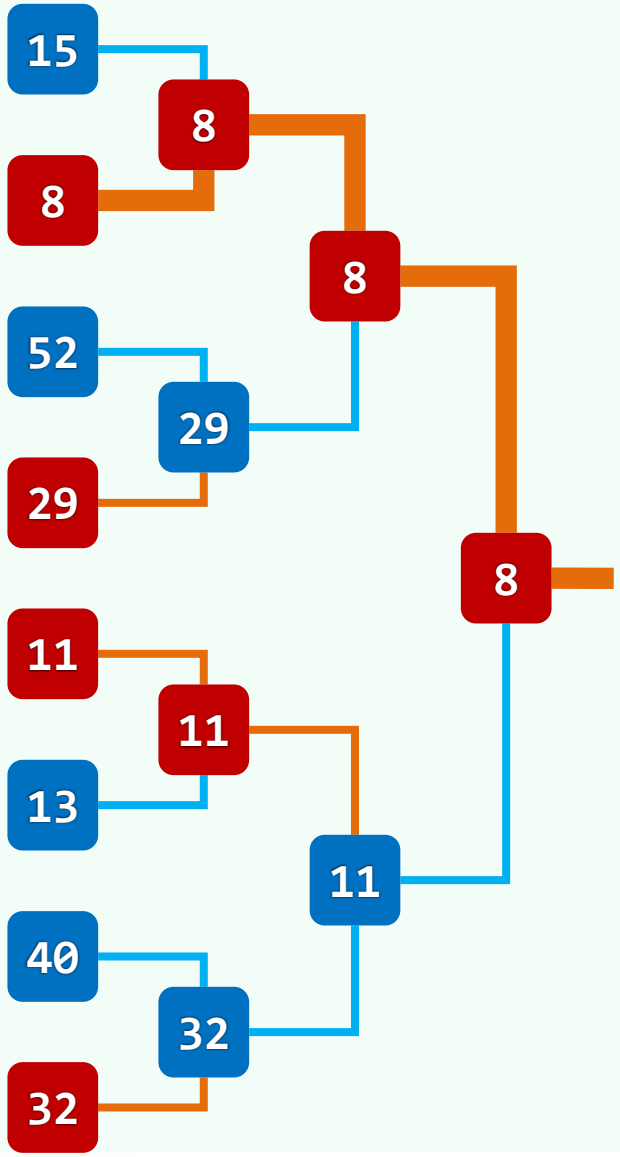

15
8
8
8
52
29
29
8
11
11
13
11
40
32
32

效率

- 空间： O(节点数) = O(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(logn)
- 时间： n轮 × O(logn) = O(nlogn)，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 << n
  - 初始化： O(n) //n-1次比较
  - 迭代k步： O(k*logn) //每步logn次比较

  与小顶堆旗鼓相当？

- 渐进意义上，的确如此

  但就常系数而言，区别不小...

- Floyd算法、delMax()中的percolateDown()

  在每一层需做2次比较，累计略少于2n次