10.优先级队列

（c）锦标赛排序

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

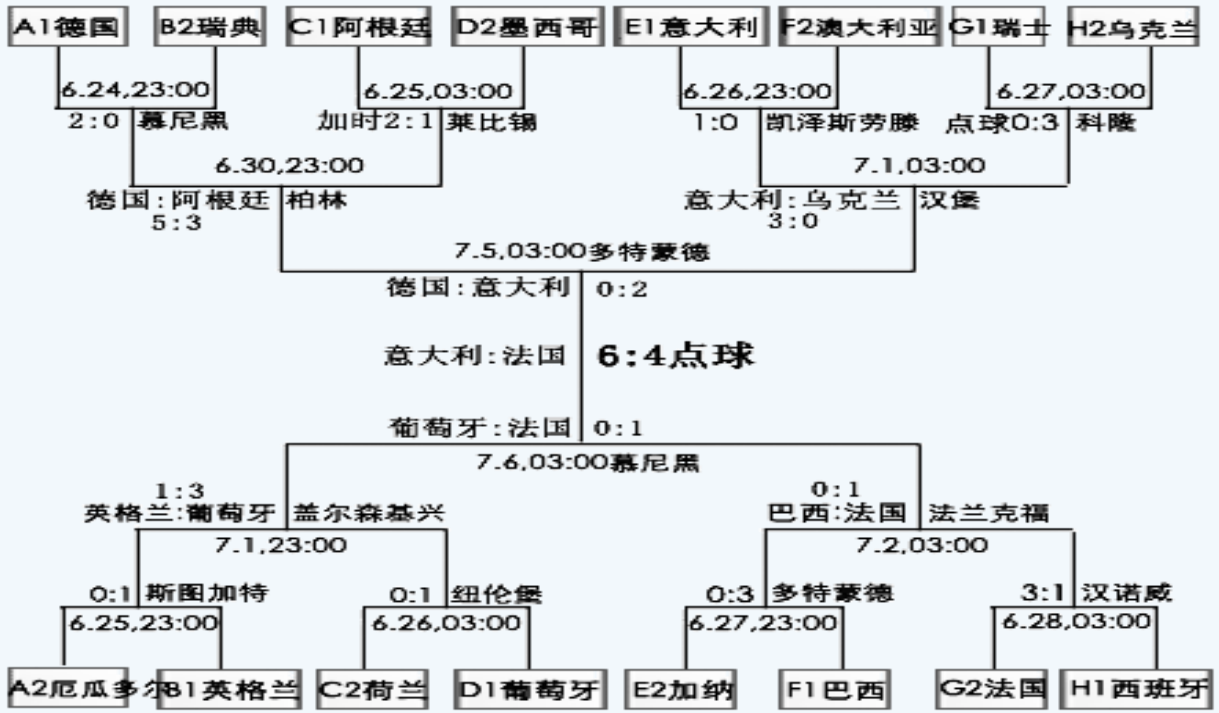
锦标赛树

- 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

  胜负判定原则：小者为胜



- create() //O(n)

  remove() //O(logn)

  insert() //O(logn)

- 树根总是全局优胜（最小）者——类似于最小堆

  树根到对应优胜者所在叶节点的路径，存在且唯一

  该路径上所有节点，关键码都相同

排序：算法

❖ Tournamentsort()

CREATE a tournament tree for the input list

while there are active leaves

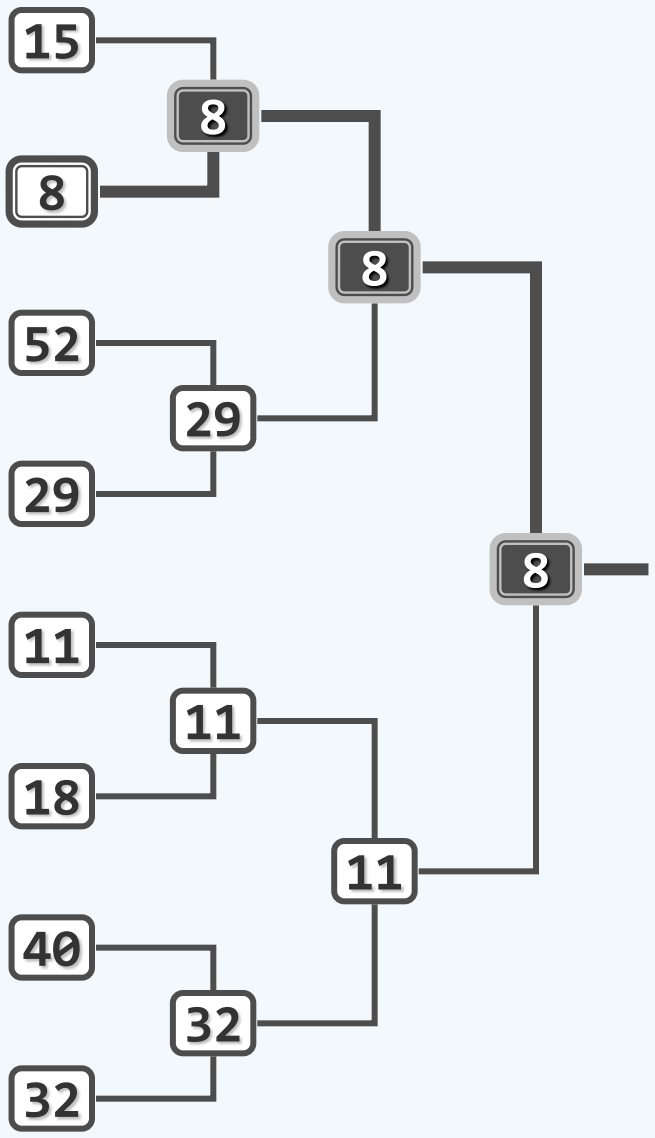
REMOVE the root

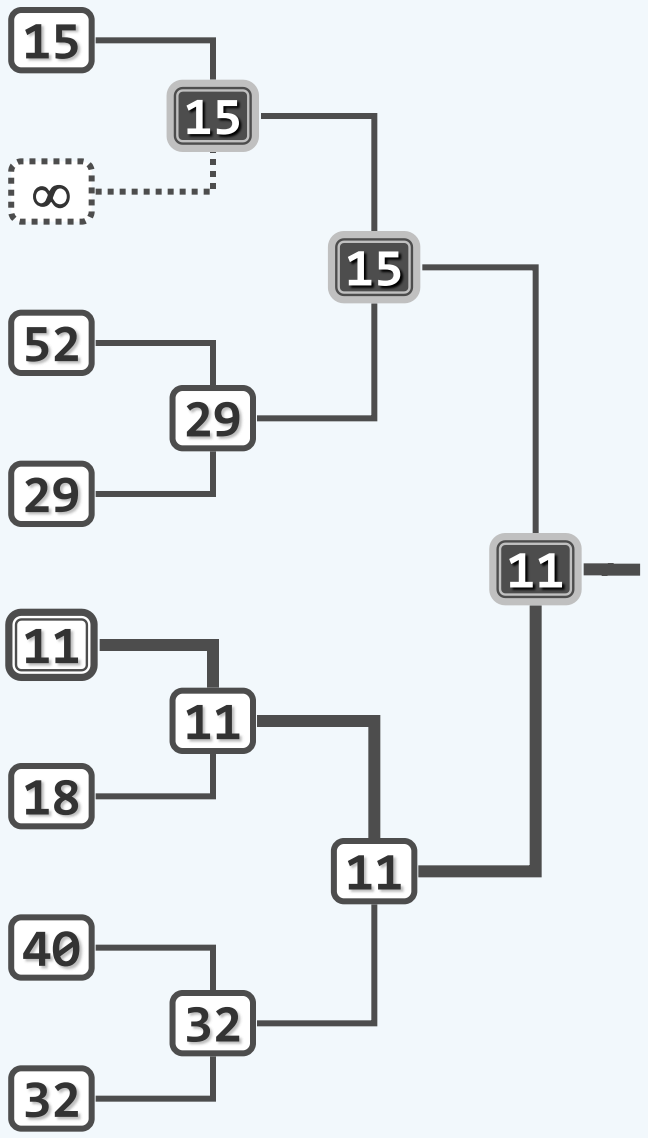
RETRACE the root down to its leaf

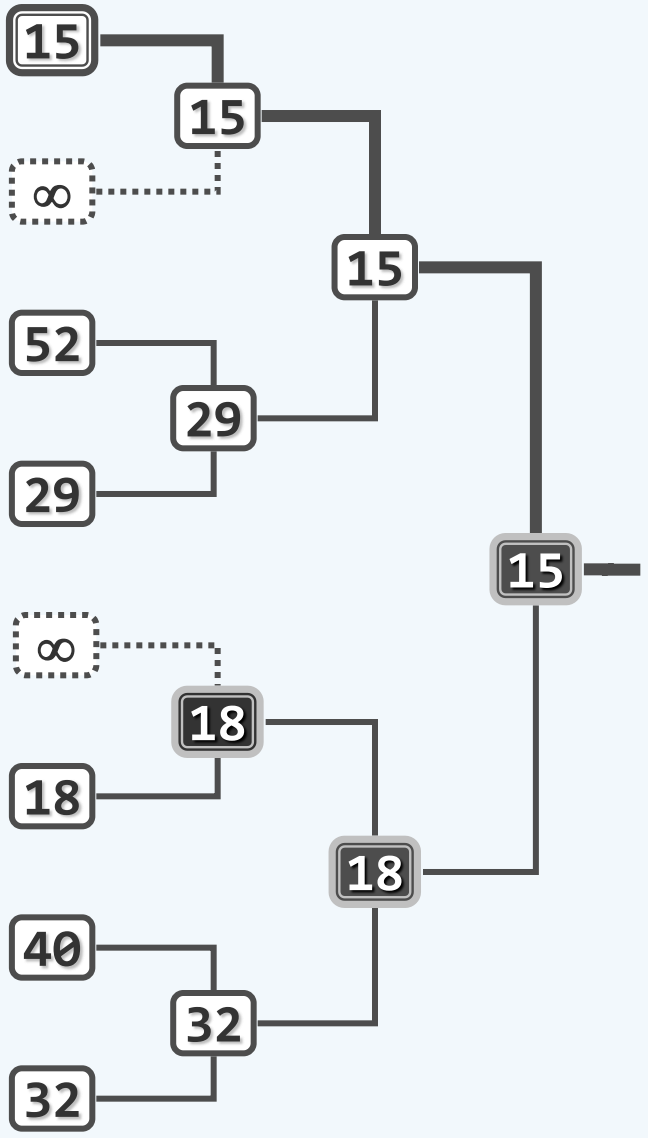
DEACTIVATE the leaf

REPLAY along the path back to the root

排序：实例


15
8
8
8
52
29
29
8
11
11
18
11
40
32
32


15
15
∞
15
52
29
29
11
11
11
18
11
40
32
32


15
15
∞
15
52
29
29
15
∞
18
18
18
40
32
32

排序：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$O(\log n)$
- 时间： n轮 × $O(\log n)$ = $O(n\log n)$，达到下界

  与`selectionSort()`的$O(n^2)$有天壤之别

选取

- 从n个元素中找出最小的k个，k << n　//回忆第01章的max2()，即是k = 2 << n
- 借助锦标赛树　初始化：　O(n)　//n - 1次比较

  迭代k步：　O(k*logn)　//每步logn次比较

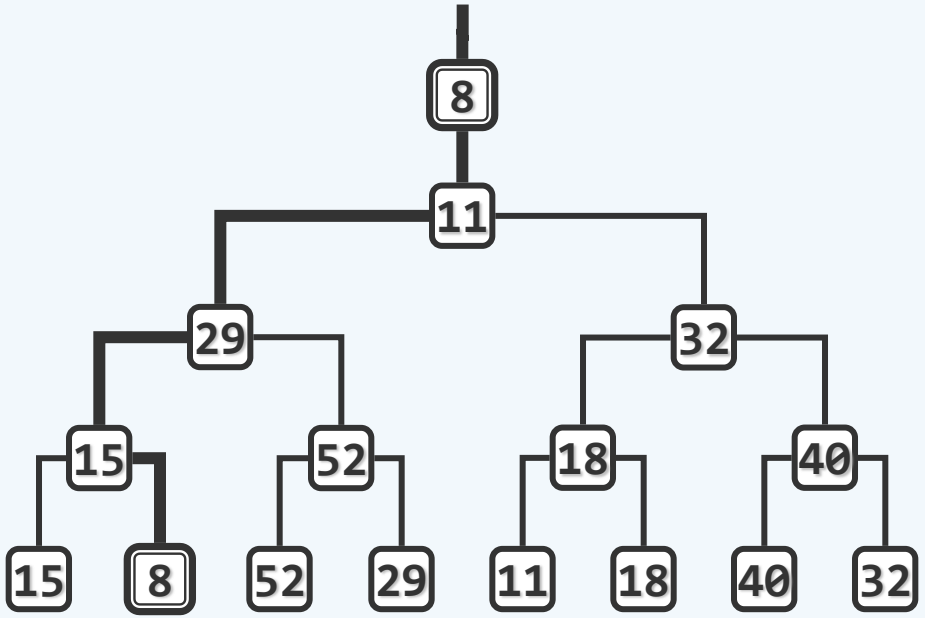
  与小顶堆旗鼓相当？
- 就渐进意义而言，的确如此

  但就常系数而言，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

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？
- 锦标赛排序的效率，可进一步提高...
- 败者树loser tree



逐层重赛过程中，沿途不必访问兄弟节点

1d-tree即是败者树的特例（只不过是大者优胜）