思考

- 设计一种数据结构，用来存放整数，要求提供 3 个接口
  - 添加元素
  - 获取最大值
  - 删除最大值

0	1	2	3	4	5	6
31	66	17	15	28	20	59

0	1	2	3	4	5	6
15	17	20	28	31	59	66

	获取最大值	删除最大值	添加元素	
动态数组\双向链表	O(n)	O(n)	O(1)	
有序动态数组\双向链表	O(1)	O(1)	O(n)	全排序有点浪费
BBST	O(logn)	O(logn)	O(logn)	杀鸡用了牛刀

- 有没有更优的数据结构?
  - 堆
  - ✓ 获取最大值：O(1)、删除最大值：O(logn)、添加元素：O(logn)

小码哥教育 SEEMYGO

Top K问题

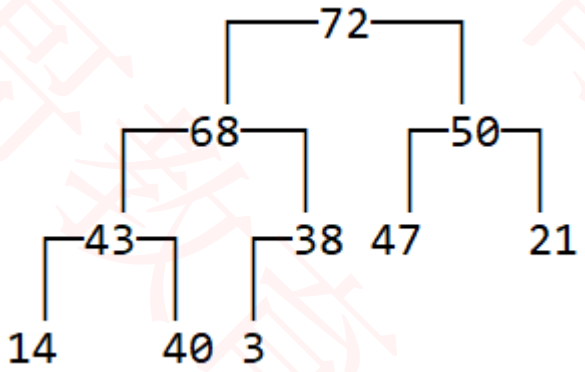
- 什么是 Top K 问题
  - 从海量数据中找出前 K 个数据

- 比如
  - 从 100 万个整数中找出最大的 100 个整数

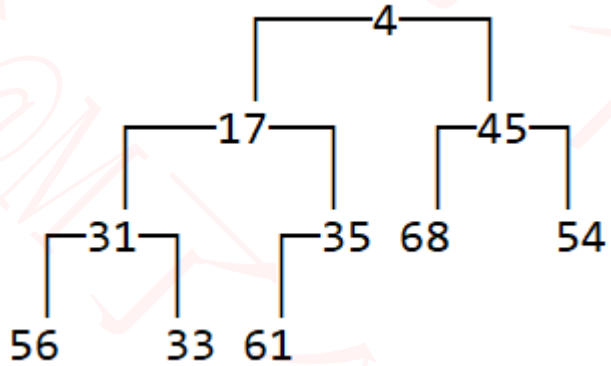
- Top K 问题的解法之一：可以用数据结构“堆”来解决

堆（Heap）

- 堆（Heap）也是一种树状的数据结构（不要跟内存模型中的“堆空间”混淆），常见的堆实现有
  - 二叉堆（Binary Heap，完全二叉堆）
  - 多叉堆（D-heap、D-ary Heap）
  - 索引堆（Index Heap）
  - 二项堆（Binomial Heap）
  - 斐波那契堆（Fibonacci Heap）
  - 左倾堆（Leftist Heap，左式堆）
  - 斜堆（Skew Heap）



最大堆_二叉堆



最小堆_二叉堆

- 堆的一个重要性质：任意节点的值总是 ≥（≤）子节点的值
  - 如果任意节点的值总是 ≥ 子节点的值，称为：最大堆、大根堆、大顶堆
  - 如果任意节点的值总是 ≤ 子节点的值，称为：最小堆、小根堆、小顶堆

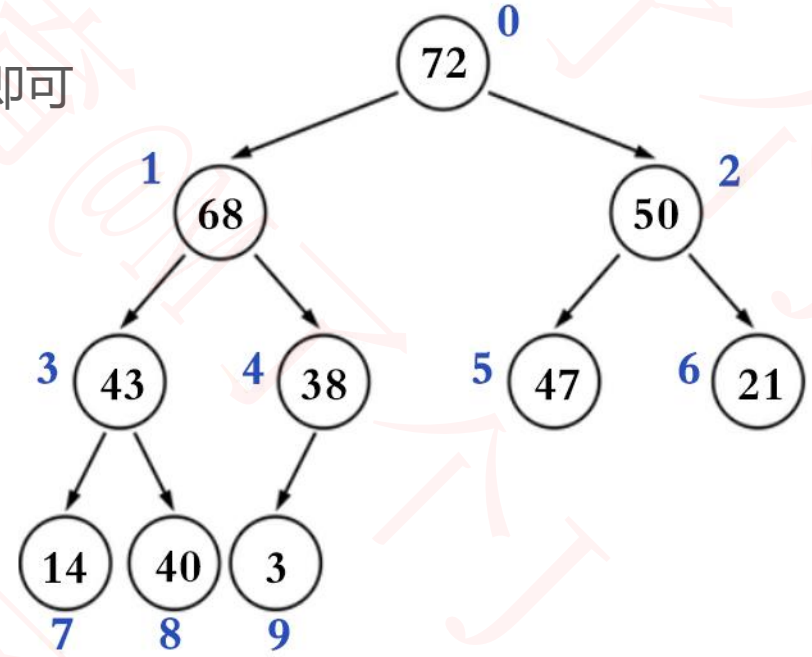
- 由此可见，堆中的元素必须具备可比较性（跟二叉搜索树一样）

堆的基本接口设计

- `int size();  // 元素的数量`
- `boolean isEmpty();  // 是否为空`
- `void clear();  // 清空`
- `void add(E element); // 添加元素`
- `E get();  // 获得堆顶元素`
- `E remove(); // 删除堆顶元素`
- `E replace(E element); // 删除堆顶元素的同时插入一个新元素`

二叉堆（Binary Heap）

- 二叉堆的逻辑结构就是一棵完全二叉树，所以也叫完全二叉堆
- 鉴于完全二叉树的一些特性，二叉堆的底层（物理结构）一般用数组实现即可
- 索引 i 的规律（ n 是元素数量）
  - 如果 i = 0 ， 它是根节点
  - 如果 i > 0 ， 它的父节点的索引为 floor((i – 1) / 2)
  - 如果 2i + 1 ≤ n – 1，它的左子节点的索引为 2i + 1
  - 如果 2i + 1 > n – 1 ， 它无左子节点
  - 如果 2i + 2 ≤ n – 1 ， 它的右子节点的索引为 2i + 2
  - 如果 2i + 2 > n – 1 ， 它无右子节点



0	1	2	3	4	5	6	7	8	9
72	68	50	43	38	47	21	14	40	3

小码哥教育 SEEMYGO

获取最大值

```
public E get() {
    emptyCheck();
    return elements[0];
}
```

```
private void emptyCheck() {
    if (size == 0) {
        throw new IndexOutOfBoundsException("Heap is empty");
    }
}
```