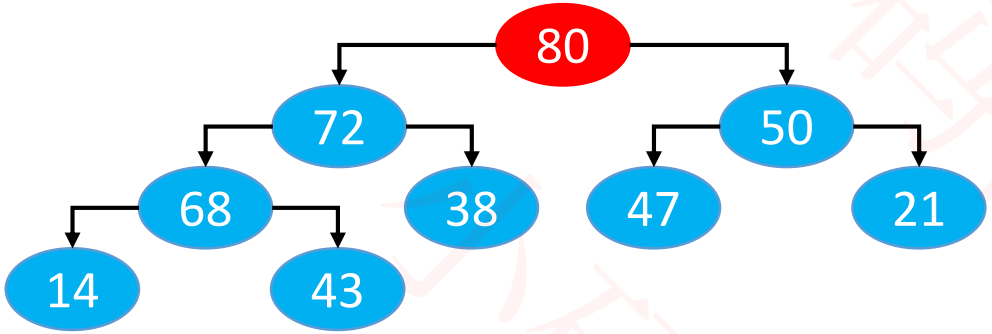
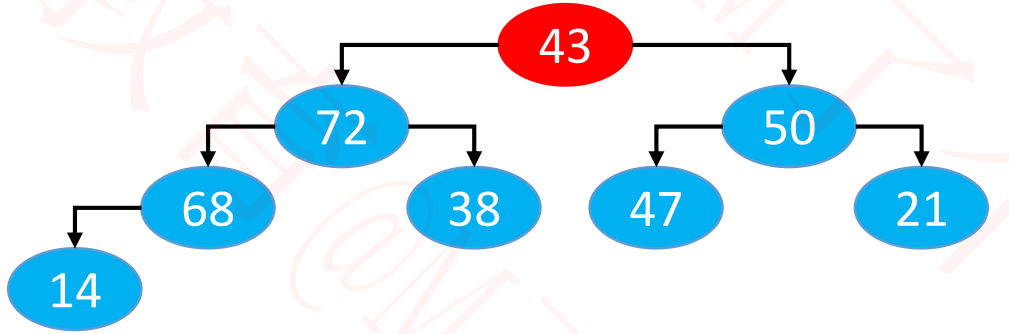
小码哥教育 SEEMYGO

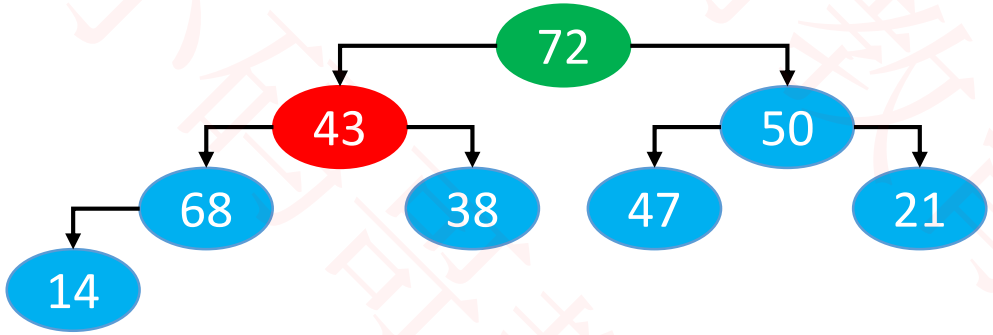
最大堆 – 删除



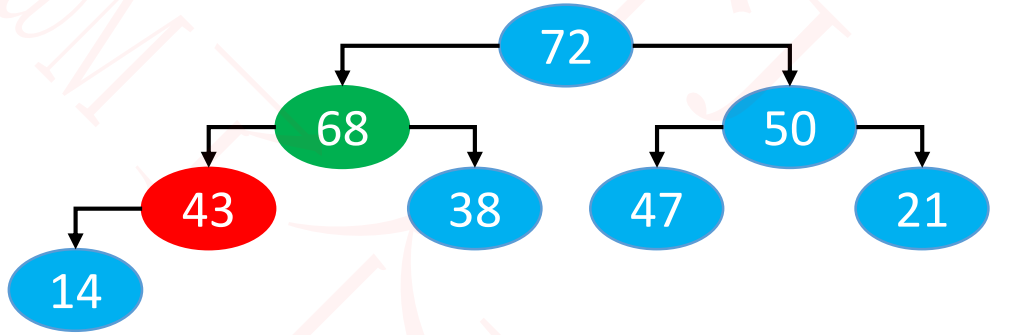
0	1	2	3	4	5	6	7	8	9
80	72	50	68	38	47	21	14	43	



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

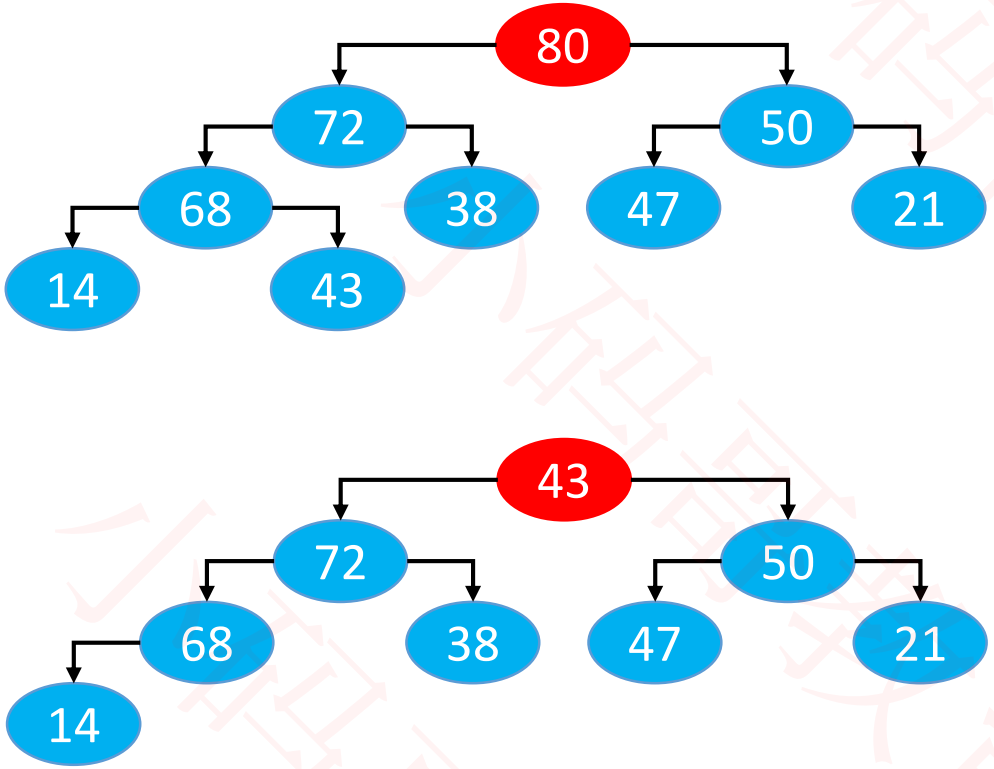


0	1	2	3	4	5	6	7	8	9
72	43	50	68	38	47	21	14		



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

最大堆 – 删除 – 总结



1. 用最后一个节点覆盖根节点
2. 删除最后一个节点
3. 循环执行以下操作（图中的 43 简称为 node）

- 如果 node < 最大的子节点
  - ✓ 与最大的子节点交换位置
- 如果 node ≥ 最大的子节点， 或者 node 没有子节点
  - ✓ 退出循环

- 这个过程，叫做下滤（Sift Down），时间复杂度：O(logn)

- 同样的，交换位置的操作可以像添加那样进行优化

最大堆 - 删除

```java
public E remove() {
    emptyCheck();
    E first = elements[0];
    int lastIndex = --size;
    elements[0] = elements[lastIndex];
    elements[lastIndex] = null;
    siftDown(0);
    return first;
}
```

```java
private void siftDown(int index) {
    E element = elements[index];

    int half = size >> 1;
    while (index < half) { // index必须是非叶子节点
        // 默认是左边跟父节点比
        int childIndex = (index << 1) + 1;
        E child = elements[childIndex];

        int rightIndex = childIndex + 1;
        // 右子节点比左子节点大
        if (rightIndex < size &&
                compare(elements[rightIndex], child) > 0) {
            child = elements[childIndex = rightIndex];
        }

        // 大于等于子节点
        if (compare(element, child) >= 0) break;

        elements[index] = child;
        index = childIndex;
    }
    elements[index] = element;
}
```