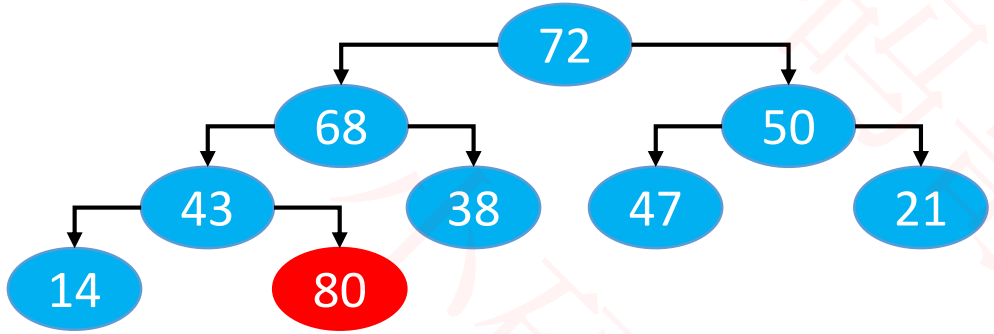
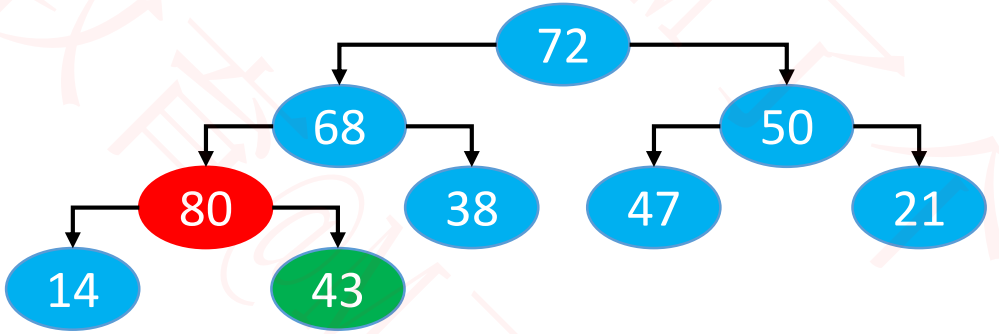
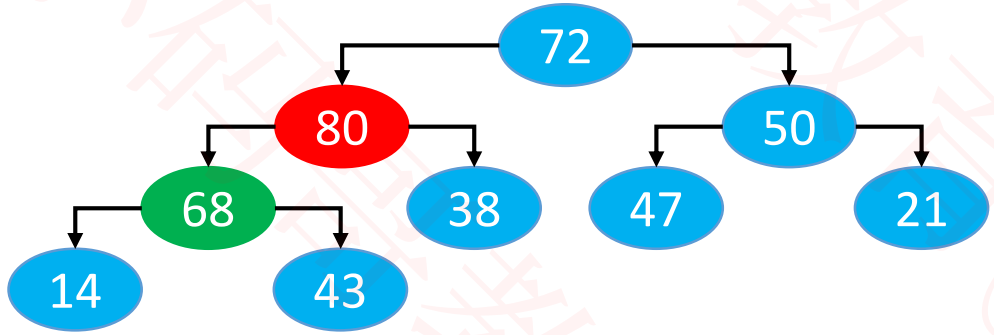
最大堆 – 添加



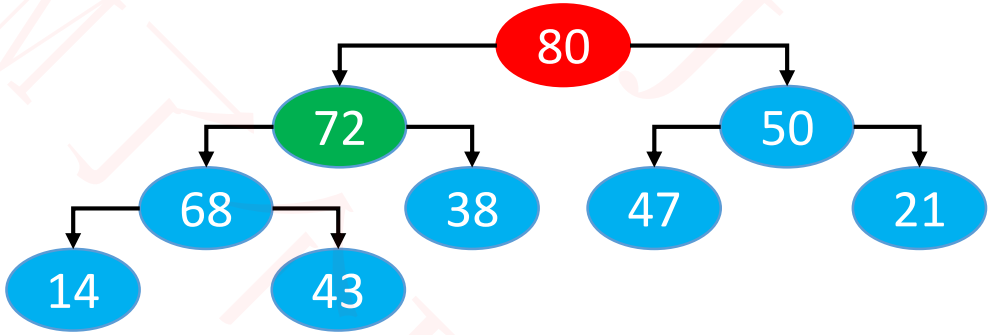
0	1	2	3	4	5	6	7	8	9
72	68	50	43	38	47	21	14	80	



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

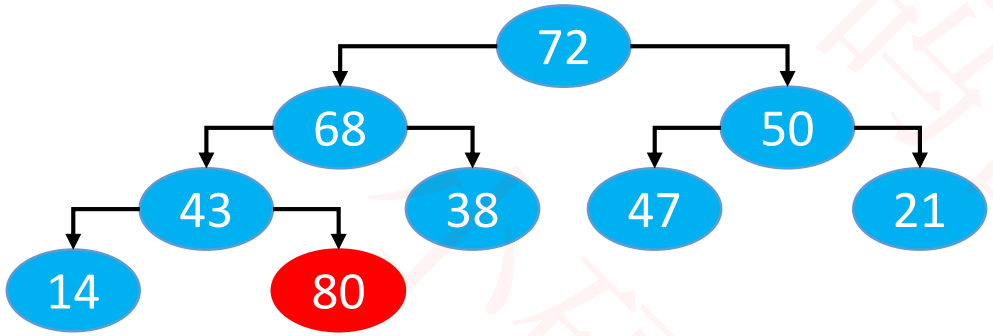


0	1	2	3	4	5	6	7	8	9
72	80	50	68	38	47	21	14	43	



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

最大堆 – 添加 – 总结



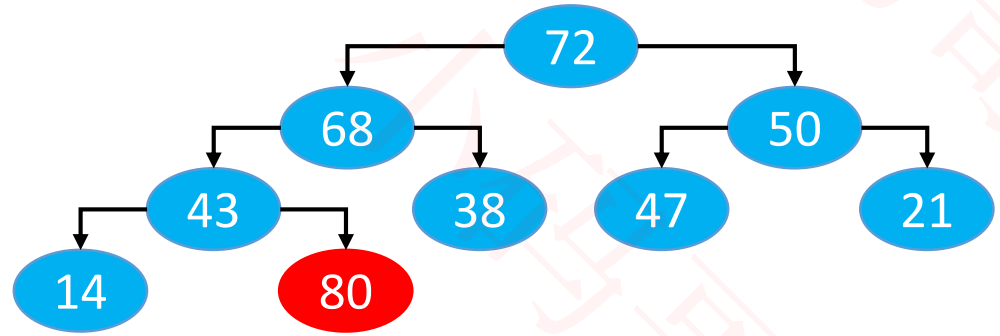

```
public void add(E element) {
    elementNotNullCheck(element);
    ensureCapacity(size + 1);
    elements[size++] = element;
    siftUp(size - 1);
}
```

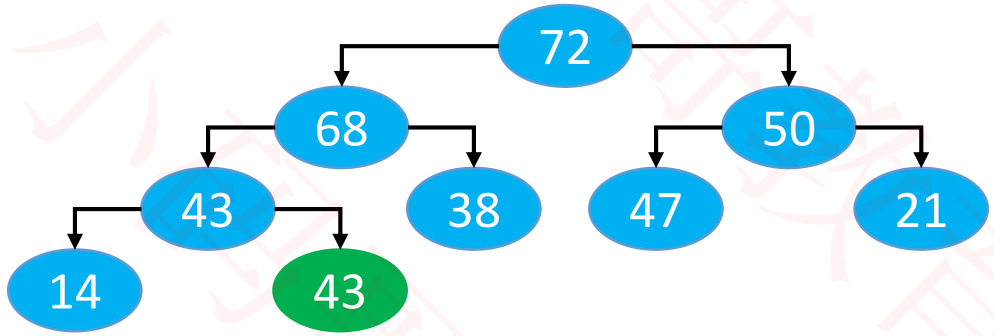
- 循环执行以下操作（图中的 80 简称为 node）
  - 如果 node > 父节点
    - ✓ 与父节点交换位置
  - 如果 node ≤ 父节点，或者 node 没有父节点
    - ✓ 退出循环
- 这个过程，叫做上滤（Sift Up）
  - 时间复杂度：O(logn)

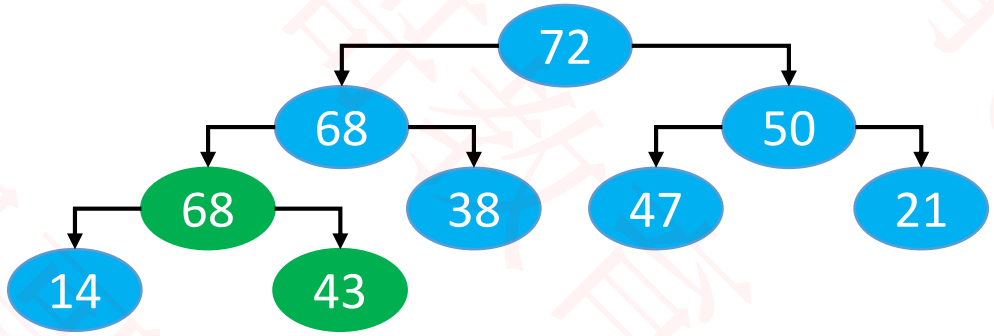
```
private void siftUp(int index) {
    E element = elements[index];
    while (index > 0) {
        int parentIndex = (index - 1) >> 1;
        E parent = elements[parentIndex];
        // 小于父节点
        if (compare(parent, element) >= 0) break;
        // 将父元素安排到index位置
        elements[index] = parent;
        index = parentIndex;
    }
    elements[index] = element;
}
```

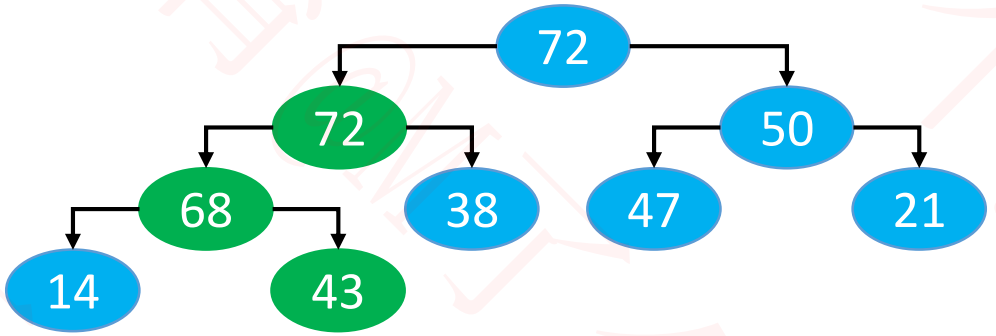
最大堆 – 添加 – 交换位置的优化

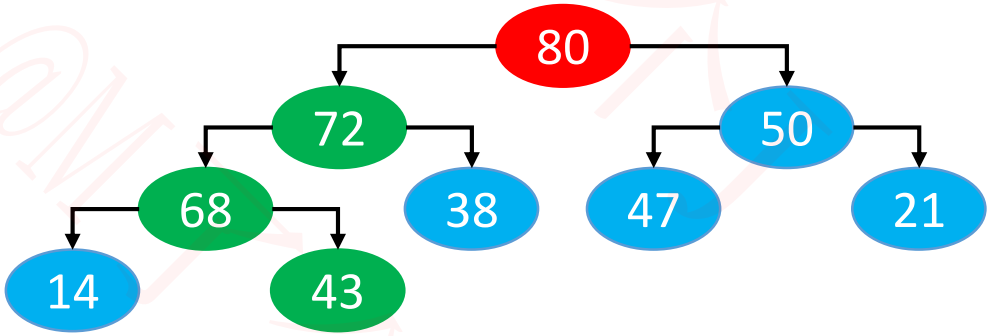
- 一般交换位置需要3行代码，可以进一步优化
  - 将新添加节点备份，确定最终位置才摆放上去











- 仅从交换位置的代码角度看
  - 可以由大概的 3 * O(logn) 优化到 1 * O(logn) + 1