第十七讲

Binary Heap

薛浩

2023 年 6 月 1 日

www.stickmind.com

今日话题

话题 1：编程基础 初学编程的新手，一般应该熟练使用函数和库处理字符串相关的编程任务。

话题 2：抽象数据类型的使用 在尝试实现抽象数据类型之前，应该先熟练使用这些工具解决问题。

话题 3：递归和算法分析 递归是一种强有力的思想，一旦掌握就可以解决很多看起来非常难的问题。

话题 4：类和内存管理 使用 C++ 实现数据抽象之前，应先学习 C++ 的内存机制。

话题 5：常见数据结构 在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

话题 5：常见数据结构

在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

- 链表
- 动态数组
- 二叉堆
- 二叉搜索树

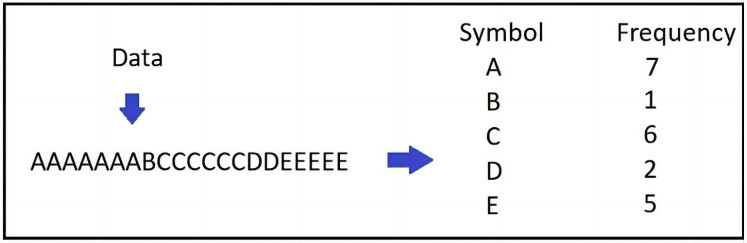


Figure 1: 数据结构和算法

如何用二叉堆实现优先队列?

目录

优先队列 Priority Queue

优先队列 Priority Queue

优先队列（priority queue）是区别于队列（queue）的另一类抽象数据类型。优先队列中的每个元素都有各自的优先级，优先级最高的元素最先出队；优先级相同的元素按照其在优先队列中的顺序依次出队。

优先队列的应用非常广泛，操作系统的实现通常都有优先队列的身影，比如 freeRTOS 等实时操作系统的任务调度系统。

优先队列的实现

优先队列至少需要支持下述操作：

- 插入带优先级的元素 enqueue(value, priority)
- 取出具有最高优先级的元素 dequeue()
- 查看最高优先级的元素 peek()

优先队列较为初级的实现是使用链表或动态数组直接存储元素，该实现入队时间复杂度可以达到 O(1)，但是出队时间复杂度却为 O(N)，查询时间复杂度也为 O(N)。

出于性能的考虑，优先队列常用二叉堆来实现，入队时间复杂度为 O(log N)，出队时间复杂度为 O(log N)，查询时间复杂度为 O(1)。

二叉堆 Binary Heap

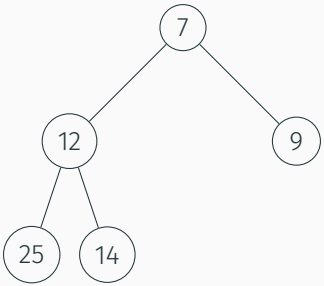
二叉堆 Binary Heap

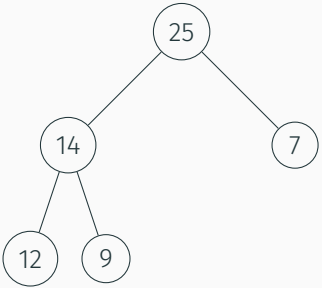
二叉堆（Binary Heap），也称作堆（heap），是一种树状的结构，父节点比任一子节点拥有较高的优先级。二叉堆有如下两个特性：

Binary	二叉表示每个父节点最多只能拥有两个子节点
Completed	除了底部节点，每个父节点都必须拥有两个子节点；底部节点必须从左到右依次填充

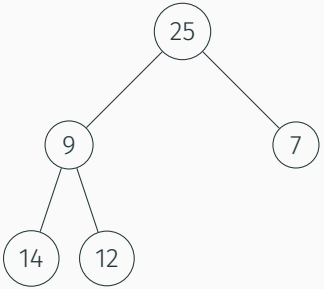
用较小的值表示高优先级的堆，称为 Min-Heap；用较大的值表示高优先级的堆，称为 Max-Heap。

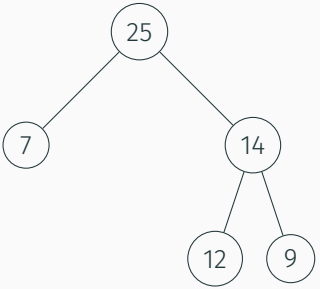
Min-Heap vs Max-Heap





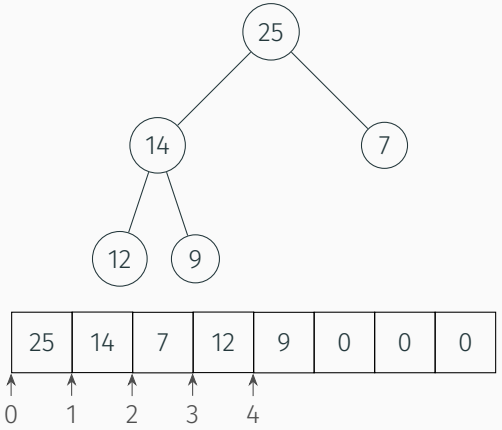
练习：Max-Heap





二叉堆的实现

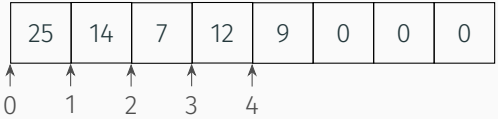
根据二叉堆的特性，可以很容易地将树状的结构映射到一个一维数组中。



二叉堆的实现

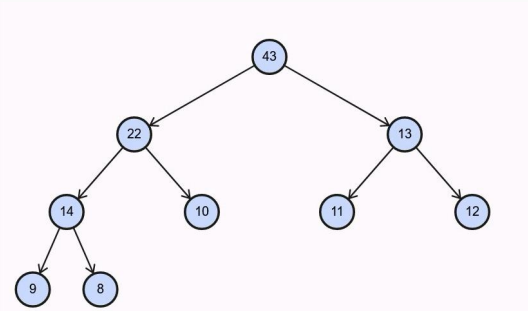
利用数组实现的二叉堆有个重要的特性：

- 任意索引 i 的父节点，其左子节点索引为 $2 * i + 1$，右子节点索引为 $2 * i + 2$
- 任意索引 i 的子节点，其父节点的索引为 $(i - 1)/2$



二叉堆操作过程

Max-Heap

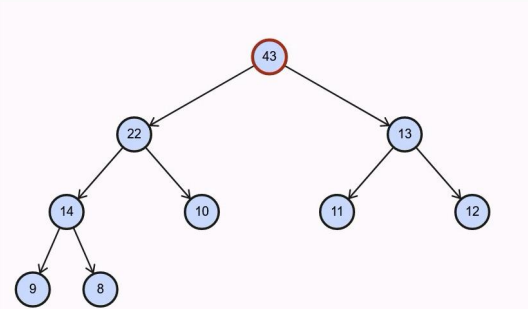


43	22	13	14	10	11	12	9	8	0
0	1	2	3	4	5	6	7	8	

二叉堆 peek 过程

- 查看最高优先级的元素 peek()
  返回最高优先级的元素（Max-Heap 返回最大值），该操作不改变堆的状态。
- 插入带优先级的元素 enqueue(value, priority)
- 取出具有最高优先级的元素 dequeue()

peek

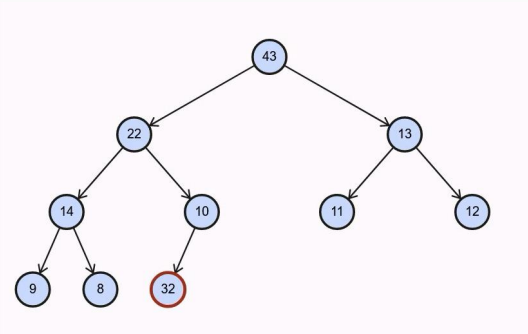


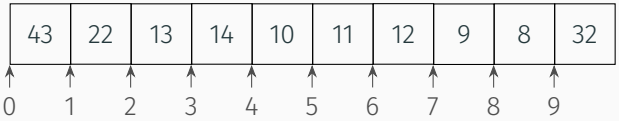
43	22	13	14	10	11	12	9	8	0
0	1	2	3	4	5	6	7	8	

二叉堆 enqueue 过程

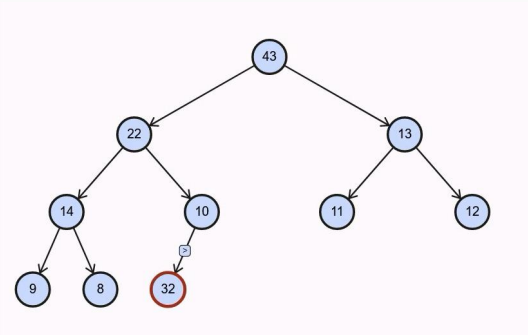
- 查看最高优先级的元素 peek()
  返回最高优先级的元素（Max-Heap 返回最大值），该操作不改变堆的状态。
- 插入带优先级的元素 enqueue(value, priority)
  新元素插入到数组后依然要保证堆的属性，此时需要 bubbleUp 操作。
- 取出具有最高优先级的元素 dequeue()

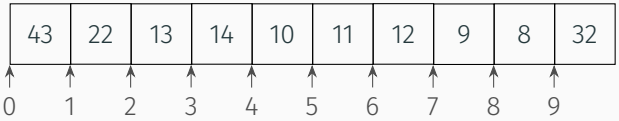
enqueue(1/5)



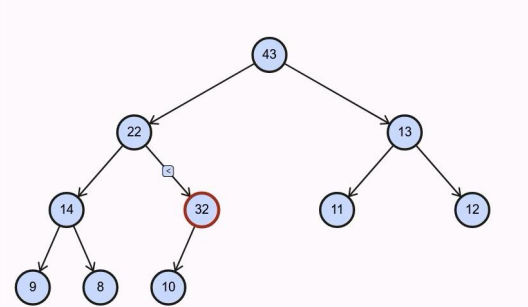


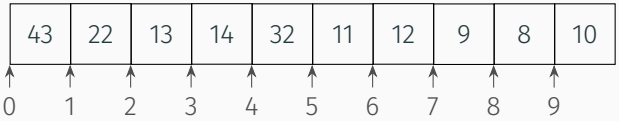
enqueue(2/5)



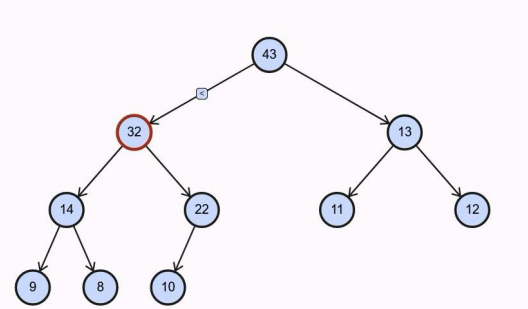


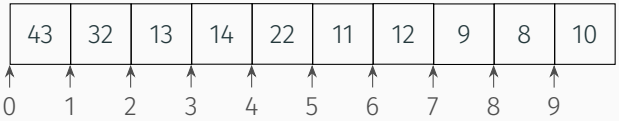
enqueue(3/5)



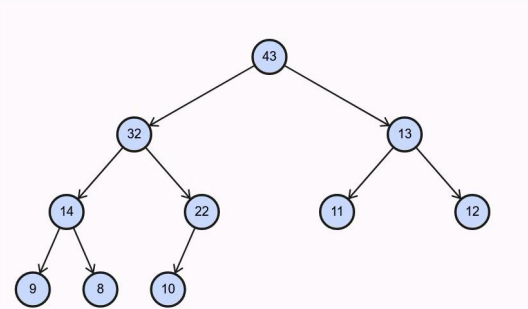


enqueue(4/5)





enqueue(5/5)



43	32	13	14	22	11	12	9	8	10
0	1	2	3	4	5	6	7	8	9

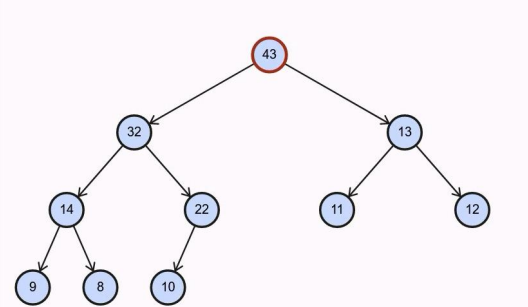
二叉堆 enqueue 过程

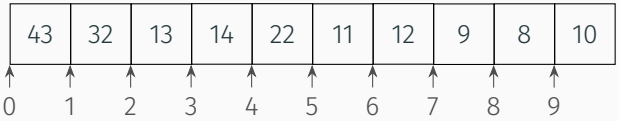
- 在数组第一个空置位置插入新元素，*此时堆的属性可能被破坏*
- 使用 bubbleUp 重复比较新元素和其父元素的关系
  如果优先级小于等于父元素，则 bubbleUp 完成;
  如果优先级大于父元素，则交换两个元素

二叉堆操作过程

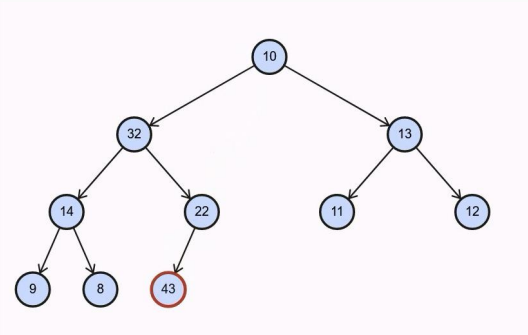
- 查看最高优先级的元素 peek()
  返回最高优先级的元素（Max-Heap 返回最大值），该操作不改变堆的状态。
- 插入带优先级的元素 enqueue(value, priority)
  新元素插入到数组后依然要保证堆的属性，此时需要 bubbleUp 操作。
- 取出具有最高优先级的元素 dequeue()
  取出 root 元素后，堆的属性也被改变，此时需要进行 bubbleDown 操作。

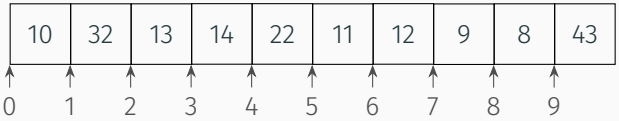
dequeue(1/6)



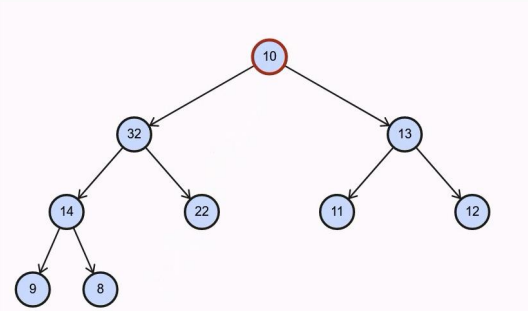


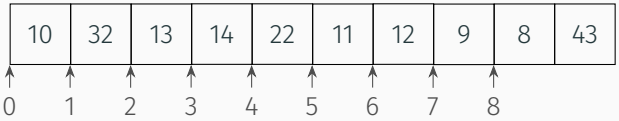
dequeue(2/6)



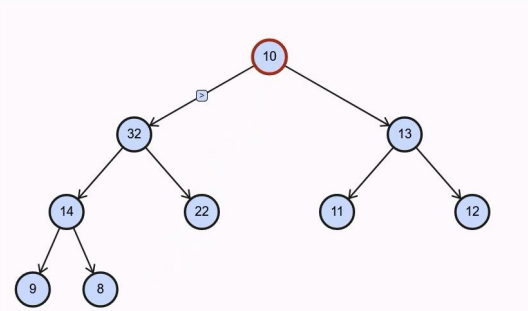


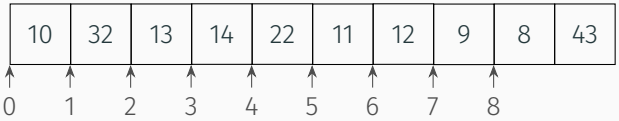
dequeue(3/6)



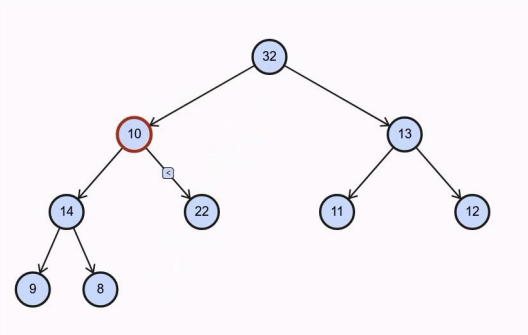


dequeue(4/6)



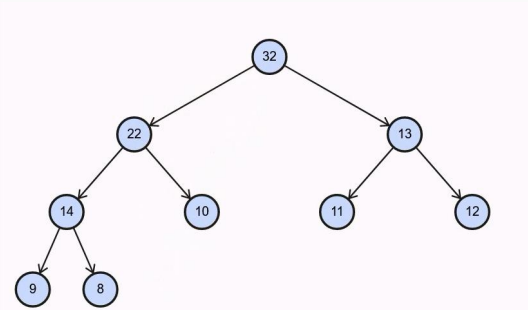


dequeue(5/6)



32	10	13	14	22	11	12	9	8	43
0	1	2	3	4	5	6	7	8	

dequeue(6/6)

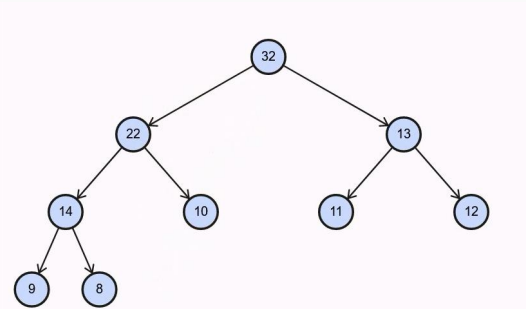


32	22	13	14	10	11	12	9	8	43
0	1	2	3	4	5	6	7	8	

二叉堆 dequeue 过程

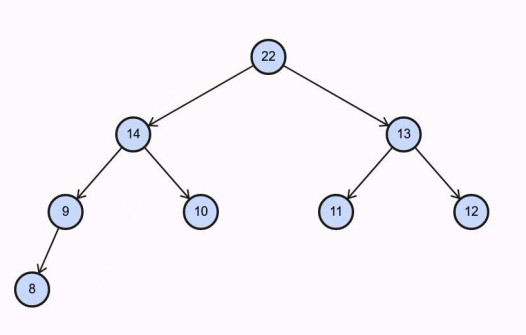
- 交换首尾两个元素来移除根元素（优先级最高），*此时堆的属性可能被破坏*
- 使用 bubbleDown 重复比较新的根元素（即原堆尾元素）和其子元素的关系
  如果仅一个子元素优先级更高，则交换父元素和该子元素;
  如果两个子元素优先级都高，则交换父元素和优先级较高的子元素

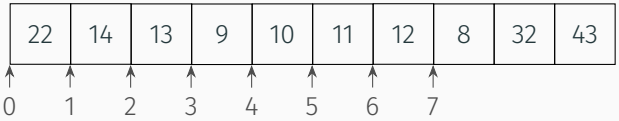
练习：dequeue



32	22	13	14	10	11	12	9	8	43
0	1	2	3	4	5	6	7	8	

练习：dequeue





堆排序 Heap Sort

堆排序 Heap Sort

优先队列的另一个应用是实现堆排序。创建一个优先队列 pq 依次插入元素，利用 pq 出队的性质就可以实现有序数组的功能。

但这样的实现有两个缺点：

- 需要额外的队列进行存储，空间复杂度为 O(N)
- 符合堆属性的底层的数组，并不能保证自身的排序状态

堆排序 Heap Sort

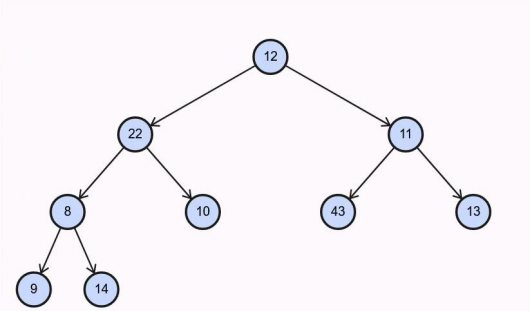
一个更好的策略是：

build 先将数组调整为符合优先队列的状态

该过程只需要从非子元素的第一个子堆 size()/2-1 开始，依次往前 bubbleDown：

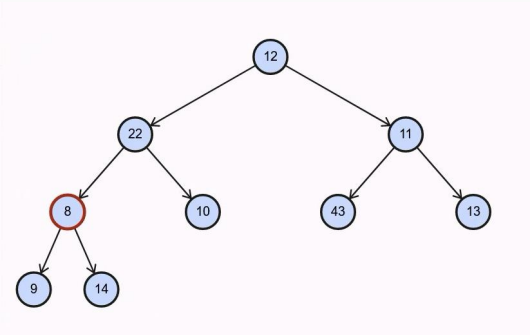
```
for (int i = size() / 2 - 1; i >= 0; i--) {
    bubbleDown(arr, n, i); // heap size is n
}
```

build(1/12)



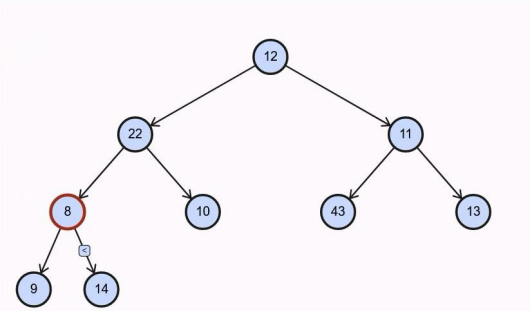
12	22	11	8	10	43	13	9	14	0
0	1	2	3	4	5	6	7	8	

build(2/12)



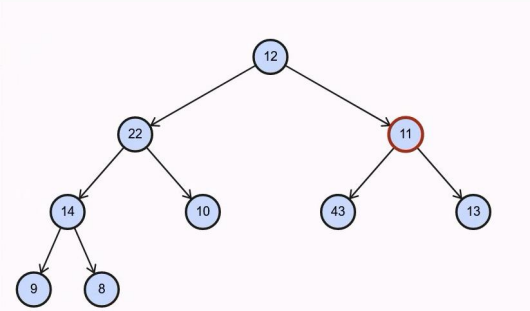
12	22	11	8	10	43	13	9	14	0
0	1	2	3	4	5	6	7	8	

build(3/12)



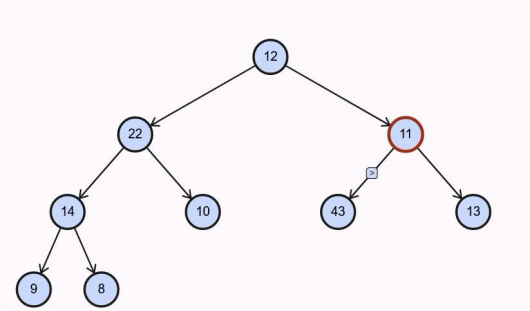
12	22	11	8	10	43	13	9	14	0
0	1	2	3	4	5	6	7	8	

build(4/12)



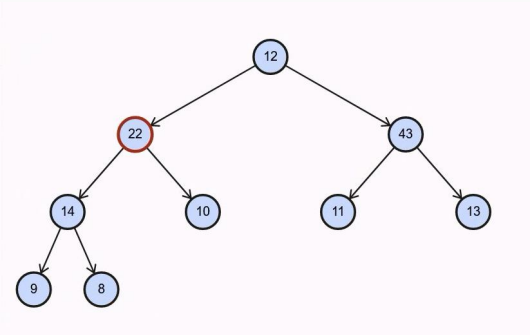
12	22	11	14	10	43	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(5/12)



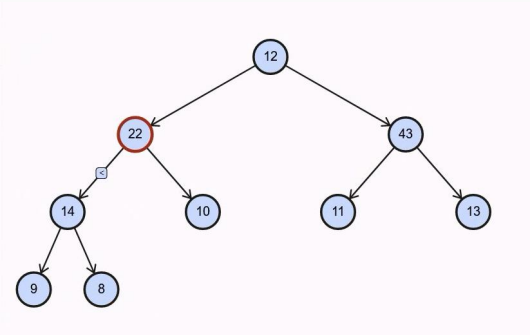
12	22	11	14	10	43	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(6/12)



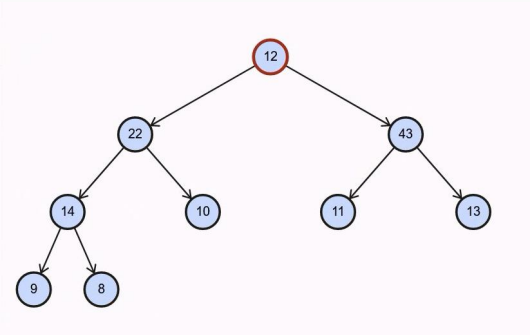
12	22	43	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(7/12)



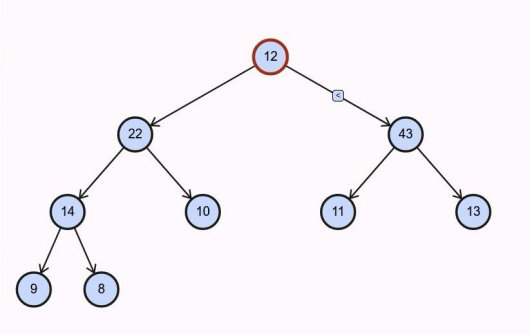
12	22	43	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(8/12)



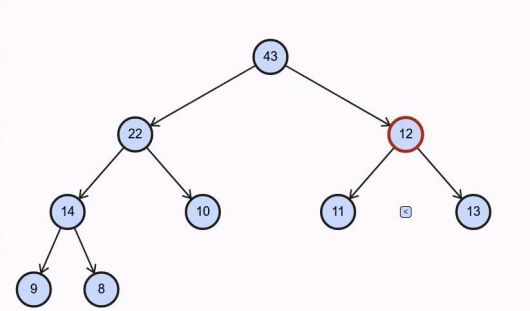
12	22	43	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(9/12)



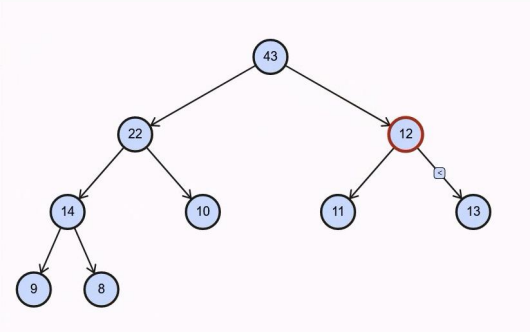
12	22	43	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(10/12)



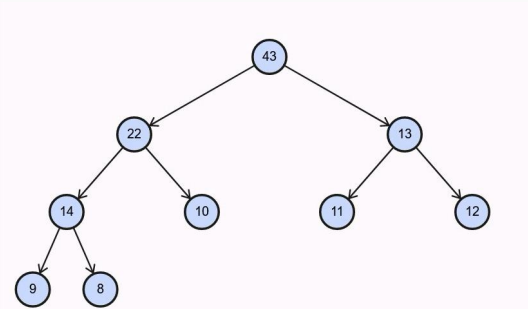
43	22	12	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

build(11/12)



43	22	12	14	10	11	13	9	8	0
0	1	2	3	4	5	6	7	8	

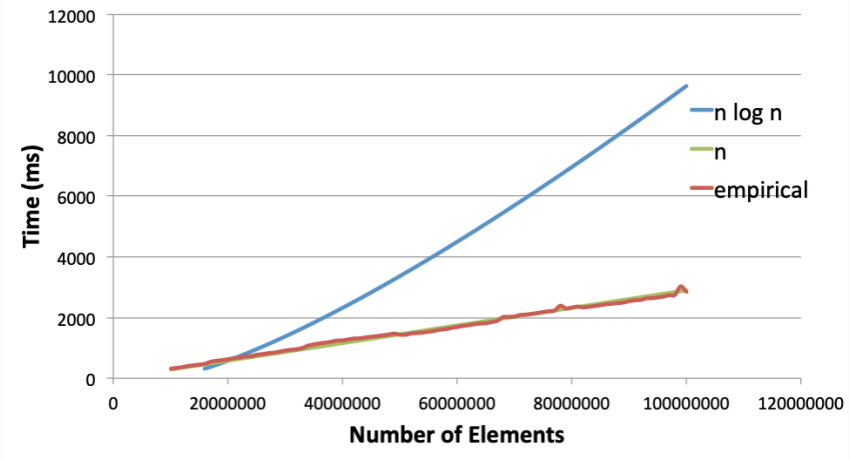
build(12/12)



43	22	13	14	10	11	12	9	8	0
0	1	2	3	4	5	6	7	8	

build 时间复杂度

该过程的时间复杂度为 O(N)，参考 https://stackoverflow.com/a/18742428。



堆排序 Heap Sort

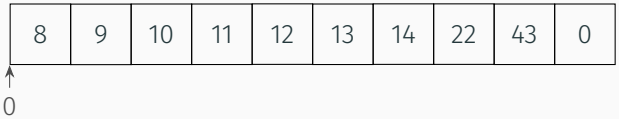
一个更好的策略是：

build	先将数组调整为符合优先队列的状态
swap	交换根元素和尾元素
remove	逻辑上减少一个堆元素
bubbleDown	此时堆的属性可能被破坏，对根元素重新 bubbleDown

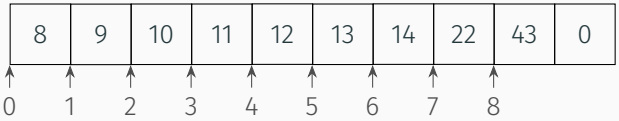
以上过程类似优先队列的 dequeue 操作，只是对数组元素来说，最终是一个排序的状态。

```
for (int i = n - 1; i >= 0; i--) {
  swap(arr[0], arr[i]);
  bubbleDown(arr, i, 0); // heap size must adjust to i
}
```

堆排序 Heap Sort



堆排序 Heap Sort



堆排序 Heap Sort

一个更好的策略是：

build	先将数组调整为符合优先队列的状态
swap	交换根元素和尾元素
remove	逻辑上减少一个堆元素
bubbleDown	此时堆的属性可能被破坏，对根元素重新 bubbleDown

综上，堆排序的时间复杂度为 O(N log N)，空间复杂度为 O(1)。

如何用二叉堆实现优先队列?