

## 算法从入门到“放弃” (10) - 堆排序



Zhou

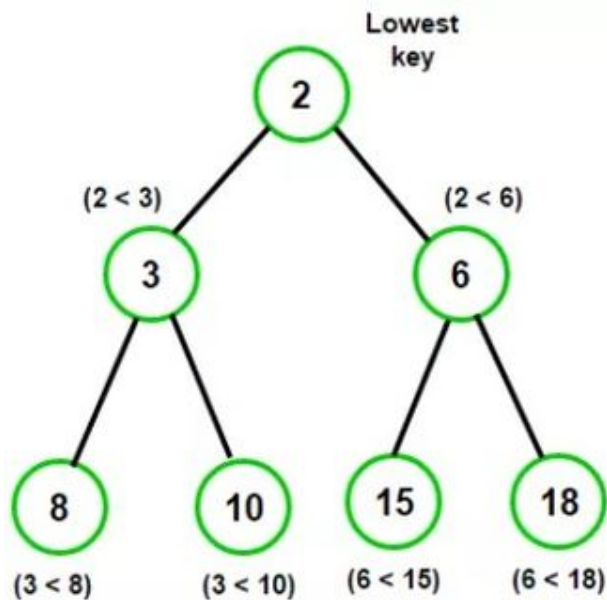
曾有两猪，现有一柴，爱好代码，兴趣攻城。

关注他

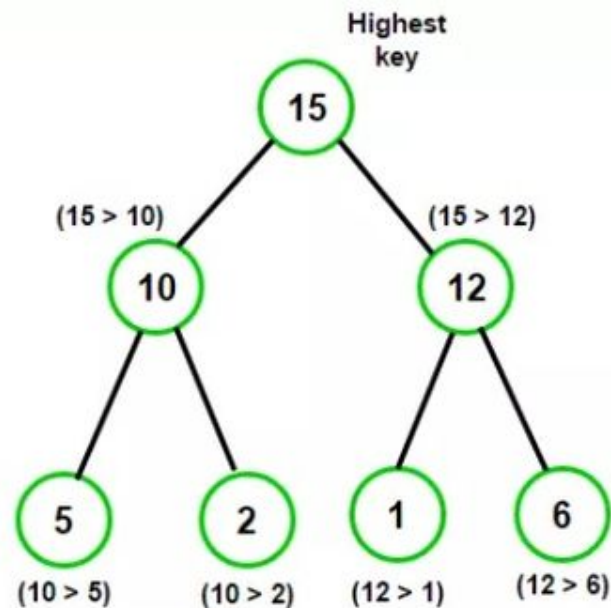
162 人赞同了该文章

### 本文提炼

堆排序算法，是一种不稳定的基于比较的排序算法。



**Min Heap**  
(Parent key is less than or equal to  $(\leq)$  the child key)



**Max Heap**  
(Parent key is greater than or equal to  $(\geq)$  the child key)

赞同 162

4 条评论

分享

喜欢

收藏

申请转载

...



## 什么是堆排序？

定义来自于[维基百科](#)，堆排序，Heap Sort.

在计算机科学中，堆排序是一种基于比较的排序算法。堆排序可以被认为是一种改进的选择排序：就像选择算法一样，它将输入分成已排序的和还未排序的区域，它通过提取未排序的区域内最大的元素并将其移动到已排序的区域来迭代缩小未排序的区域。堆排序相对选择排序改进的部分包括使用堆数据结构而不是线性时间的搜索来找到最大值。

虽然在大多数机器上的实际运行速度要比实现良好的快速排序慢一些，但是它的最坏时间复杂度是  $O(n * \log(n))$ 。堆排序是一种就地in-place 排序算法，而且它不是稳定排序。

堆排序Heapsort 是J. W. J. Williams在1964年发明的。这也是堆诞生的时候，Williams 将其本身作为一个有用的数据结构提出。同年，R. W. Floyd发表了一个改进的版本，可以对数组进行就地排序，继续他早期对[treesort](#) 算法的研究。

### 我自己的看法

堆排序，就是利用堆的特征来对目标输入数组进行排序；依据目的是从大到小还是从小到大，我们可以相应地用最大堆排序或则最小堆排序来取出堆顶的最大数或最小数，再将剩下的堆调整为最大堆或最小堆；整个过程持续迭代至只有一个数剩下。

赞同 162

分享

赞同 162

4 条评论

分享

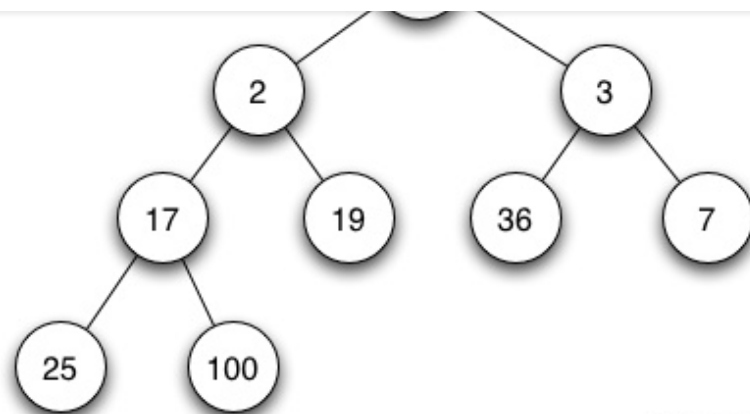
喜欢

收藏

申请转载

...





知乎 @Zhou

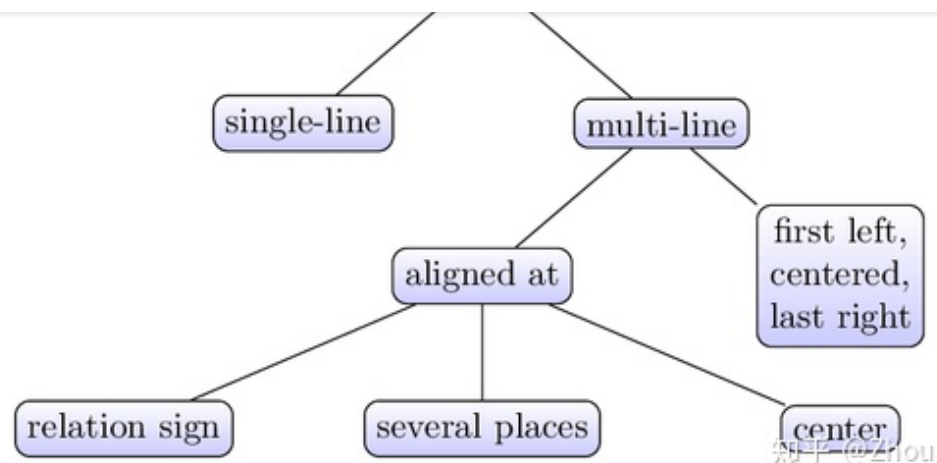
一个堆的展示

## 什么是堆？

这个问题本应该在我的数据结构系列里进行讲解，奈何算法系列先行一步讲到了堆。那我们先来简单聊聊，等到日后我更新数据系列到了堆再进行补充。

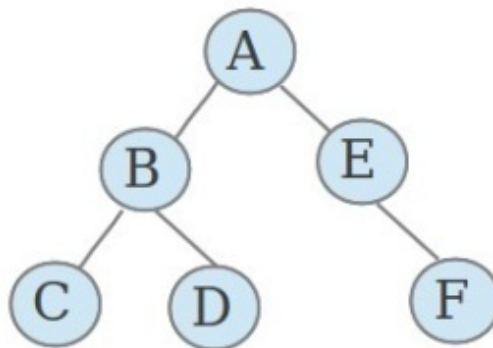
讲解堆之前，我们先了解下什么是树形结构，下面的树结构 tree structure 引用自[维基百科](#)。

树状结构或树形图是一种表示图形形式结构的层次结构的方法。它被命名为“树状结构”，因为其经典的表示类似于一棵树，尽管与实际的树相比，图表通常是颠倒的，顶部是“根”，底部是“叶子”。



这张图就用了树结构

而在树结构中，又有一个大名鼎鼎的二叉树binary tree 结构。什么是二叉树呢？简单来说，二叉树结构指的是该树每个节点最多有两个子树的树结构，左边的子树称为左子树，右边的称为右子树，次序不能颠倒；特征是第 $i$ 层至多有 $2^{(i-1)}$ 个结点；深度为 $k$ 的二叉树至多有 $2^k - 1$ 个结点。



知乎 @Zhou

赞同 162

4 条评论

分享

喜欢

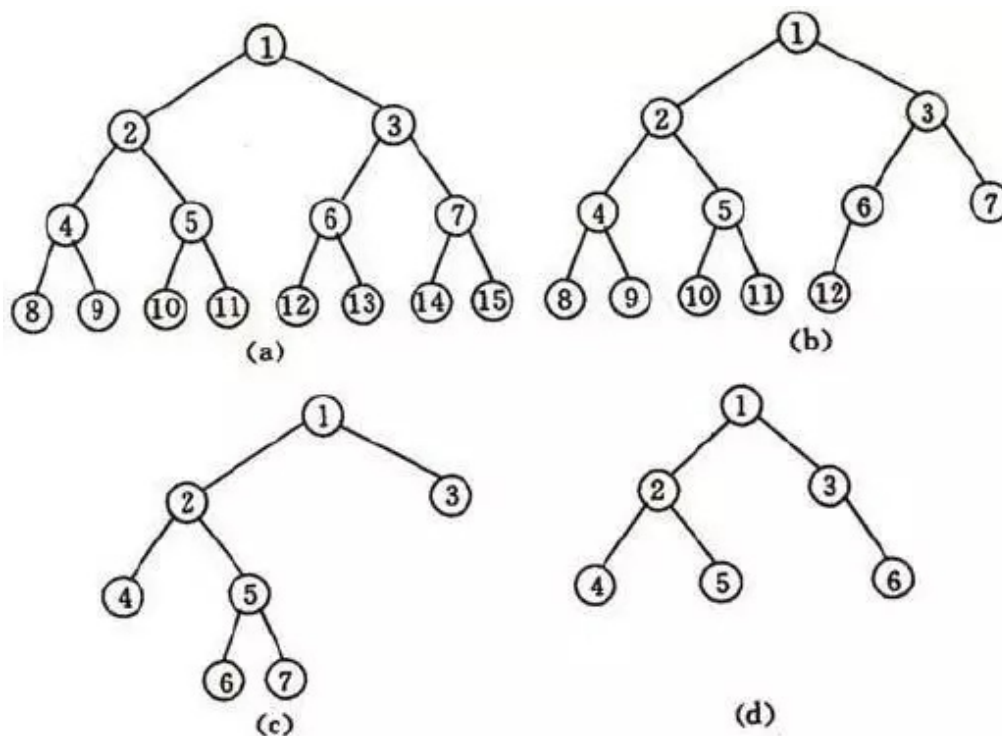
收藏

申请转载

...



有  $n$  个节点的二叉树，当且仅当其每一个节点都与深度为  $k$  的满二叉树中序号为 1 至  $n$  的节点对应；简单来说，就是除深度是  $k$  层之外，其他各层（1至 $k-1$ ）的节点数都达到最大个数，第 $k$ 层的所有节点都连续几个在最左边）。



特殊形态的二叉树

(a) 满二叉树；(b) 完全二叉树；(c) 和 (d) 非完全二叉树

图片来源：简书，侵权删

那么重点终于在基本介绍完树形结构后来了，什么是堆呢？

赞同 162

分享

赞同 162

4 条评论

分享

喜欢

收藏

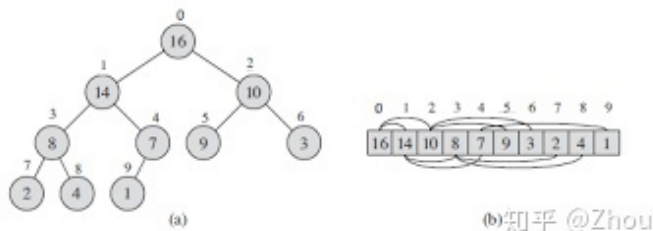
申请转载

...

通俗来说，堆可以被看作是一棵树的数组对象，堆中某个节点的值总是不大于或不小于其父节点的值。

堆的实现通常是通过构造二叉堆实现。而且因为二叉堆的应用很普遍，当不加限时，堆通常指的就是二叉堆。

堆（二叉堆）可以视为一棵完全的二叉树。完全二叉树的一个优秀的性质就是，除了最底层之外，每一层都是满的，这使得堆可以利用数组来表示（一般的二叉树通常用链表作为基本容器表示），每一个结点对应数组中的一个元素。



堆在数组中的表现，图片来源：网络，侵权删。

而二叉堆一般分为两种：最大堆和最小堆。

最大堆：最大堆中的最大元素在根结点（堆顶）；堆中每个父节点的元素值都大于等于其子结点（如果子节点存在）

最小堆：最小堆中的最小元素出现在根结点（堆顶）；堆中每个父节点的元素值都小于等于其子结点（如果子节点存在）

## 基本思想

**输入：**一系列的无序元素（比如说，数字）组成的输入数组A

**经过：**堆排序的过程可以具体分为三步，创建堆，调整堆，堆排序。

创建堆，以数组的形式将堆中所有的数据重新排序，使其成为最大堆/最小堆。

调整堆，调整过程需要保证堆序性质：在一个二叉堆中任意父节点大于其子节点。

堆排序，取出位于堆顶的第一个数据（最大堆则为最大数，最小堆则为最小数），放入输出数组B中，再将剩下的对作调整堆的迭代/重复运算直至输入数组 A中只剩下最后一个元素。

**输出：**输出数组B，里面包含的元素都是A 中的但是已经按照要求排好了顺序

## 评价算法好坏

堆排序是非稳定的排序算法

分类：排序算法

数据结构：数组

最优时间复杂度： $O(n \cdot \log(n))$ ，当keys 的值都不一样； $O(n)$ ，当keys 的值都一样

最坏时间复杂度： $O(n \cdot \log(n))$

赞同 162

分享

赞同 162

4 条评论

分享

喜欢

收藏

申请转载

...



## 实例分析

本文实例引用自[数据结构示例——堆排序过程](#)，侵权删。



赞同 162



分享

假设我们要对目标数组A {57, 40, 38, 11, 13, 34, 48, 75, 6, 19, 9, 7}进行堆排序。

首先第一步和第二步，创建堆，这里我们用最大堆；创建过程中，保证调整堆的特性。从最后一个分支的节点开始进行调整为最大堆。

▲ 赞同 162 ▼

💬 4 条评论

➦ 分享

♥ 喜欢

★ 收藏

📄 申请转载

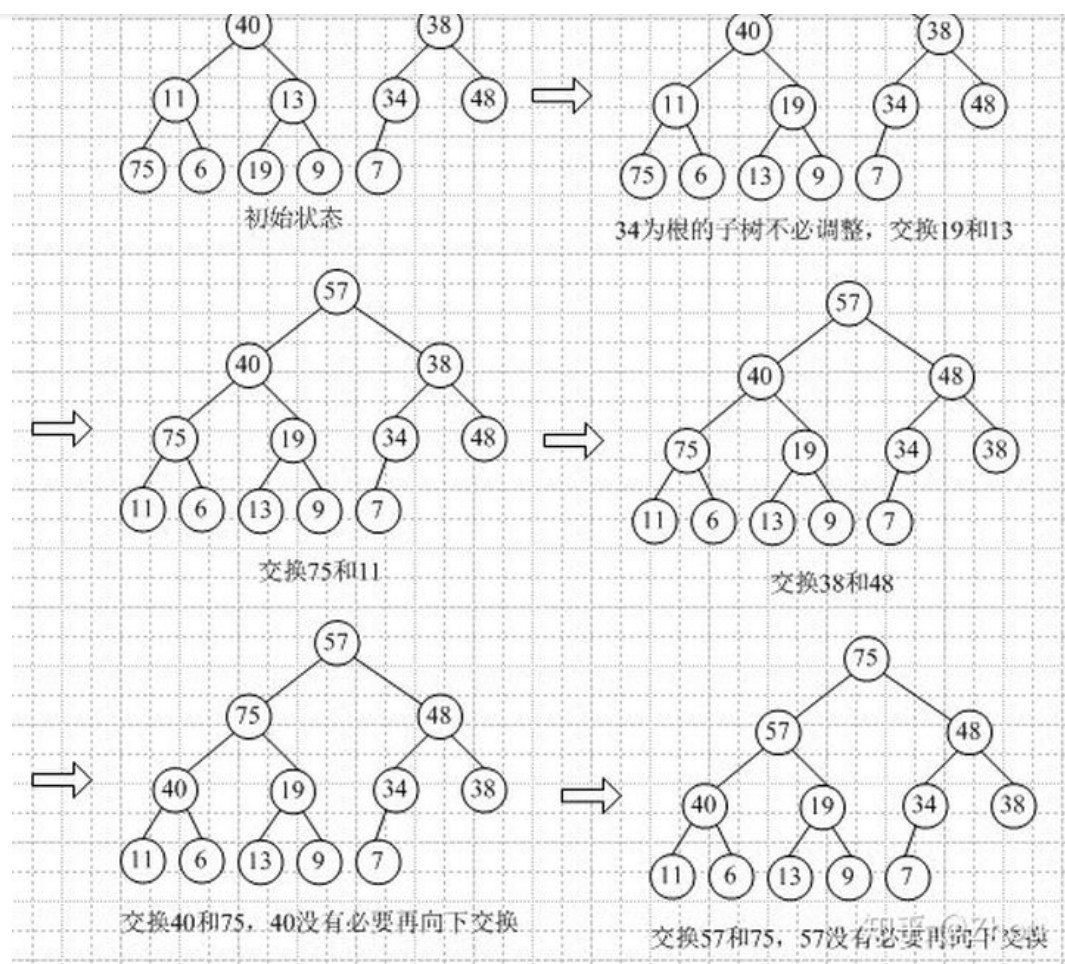
...





赞同 162

分享



从右往左，从下至上

现在得到的最大堆的存储结构如下：

	75	57	48	40	19	34	38	11	6	13	9	7
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]

赞同 162

4 条评论

分享

喜欢

收藏

申请转载

...





首发于  
AnotherWorld

写文章



赞同 162



分享

赞同 162



4 条评论

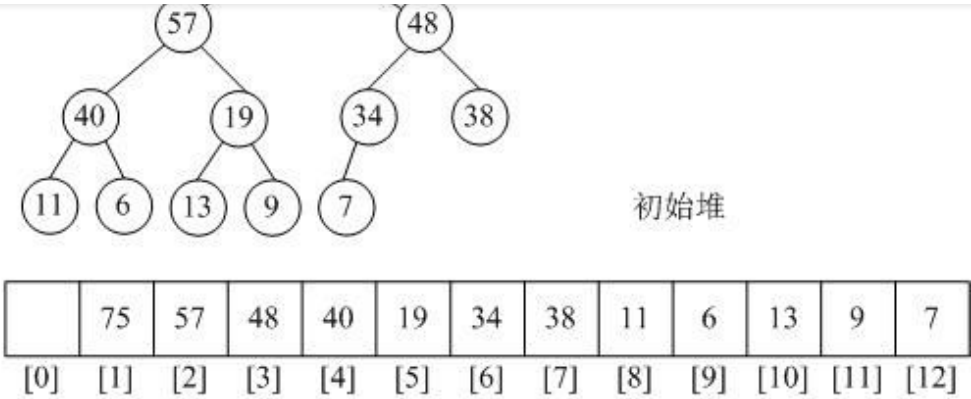
分享

喜欢

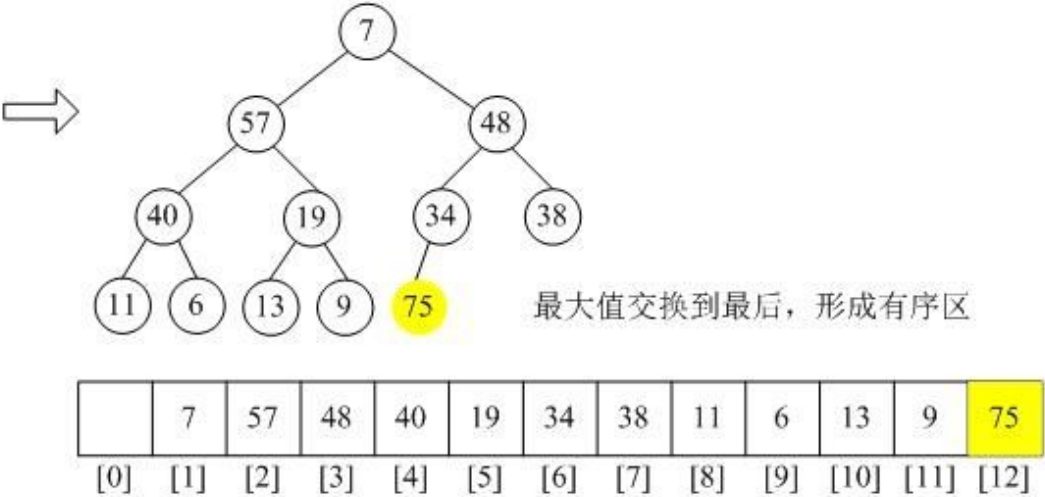
收藏

申请转载

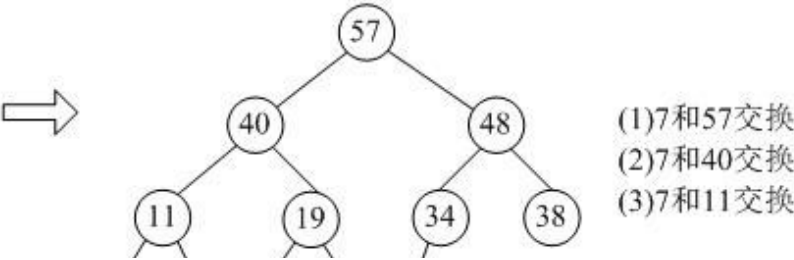




初始堆



最大值交换到最后，形成有序区



- (1) 7和57交换
- (2) 7和40交换
- (3) 7和11交换

赞同 162



分享

赞同 162

4 条评论

分享

喜欢

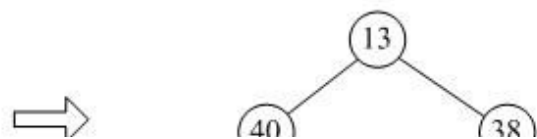
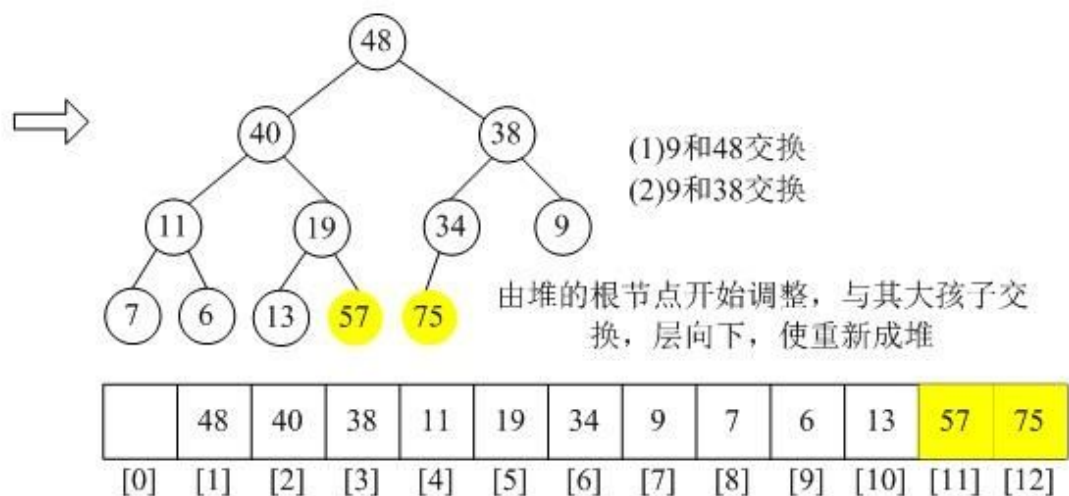
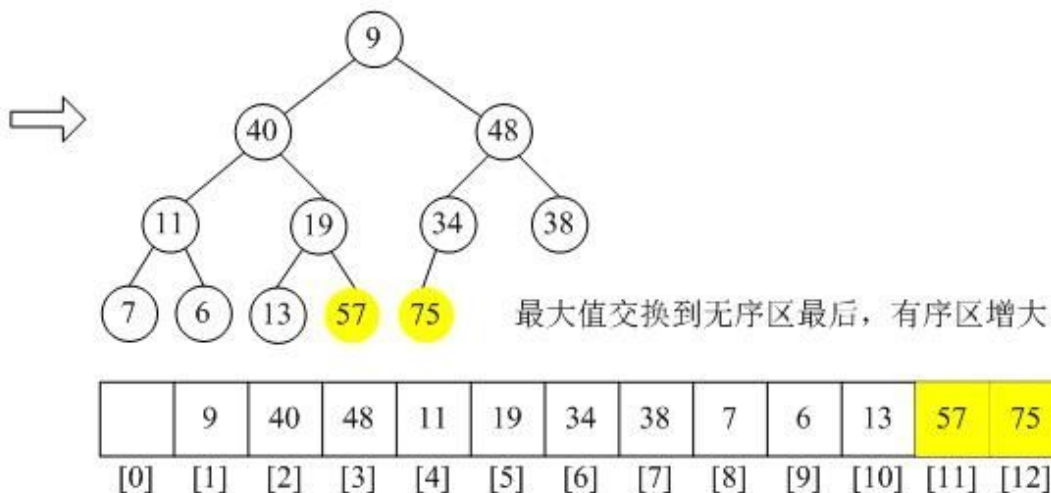
收藏

申请转载

...



[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12]



▲ 赞同 162 ▼

● 4 条评论

➦ 分享

❤ 喜欢

★ 收藏

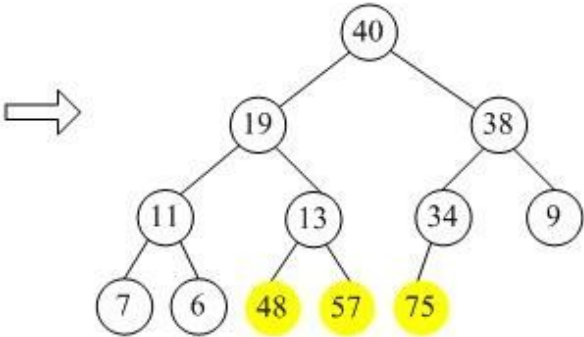
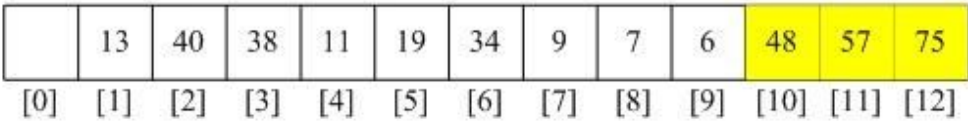
📄 申请转载

...

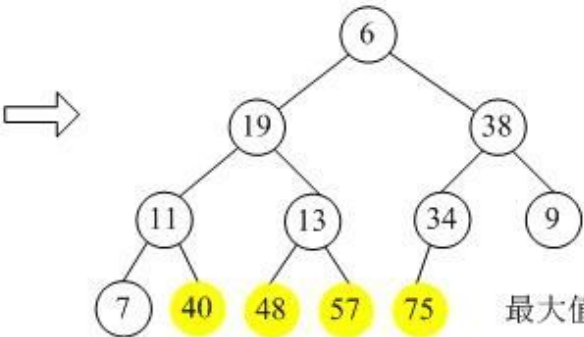
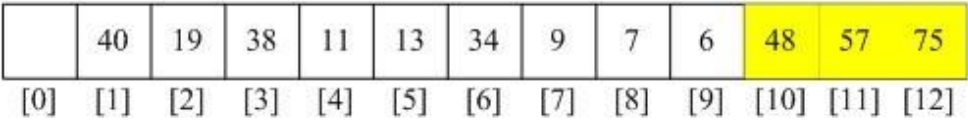


赞同 162

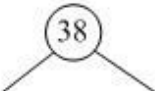
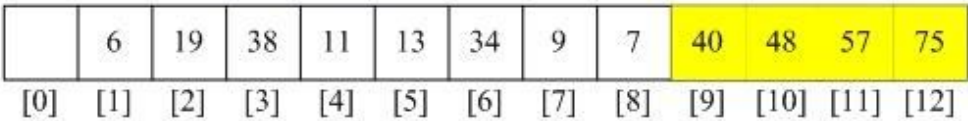
分享

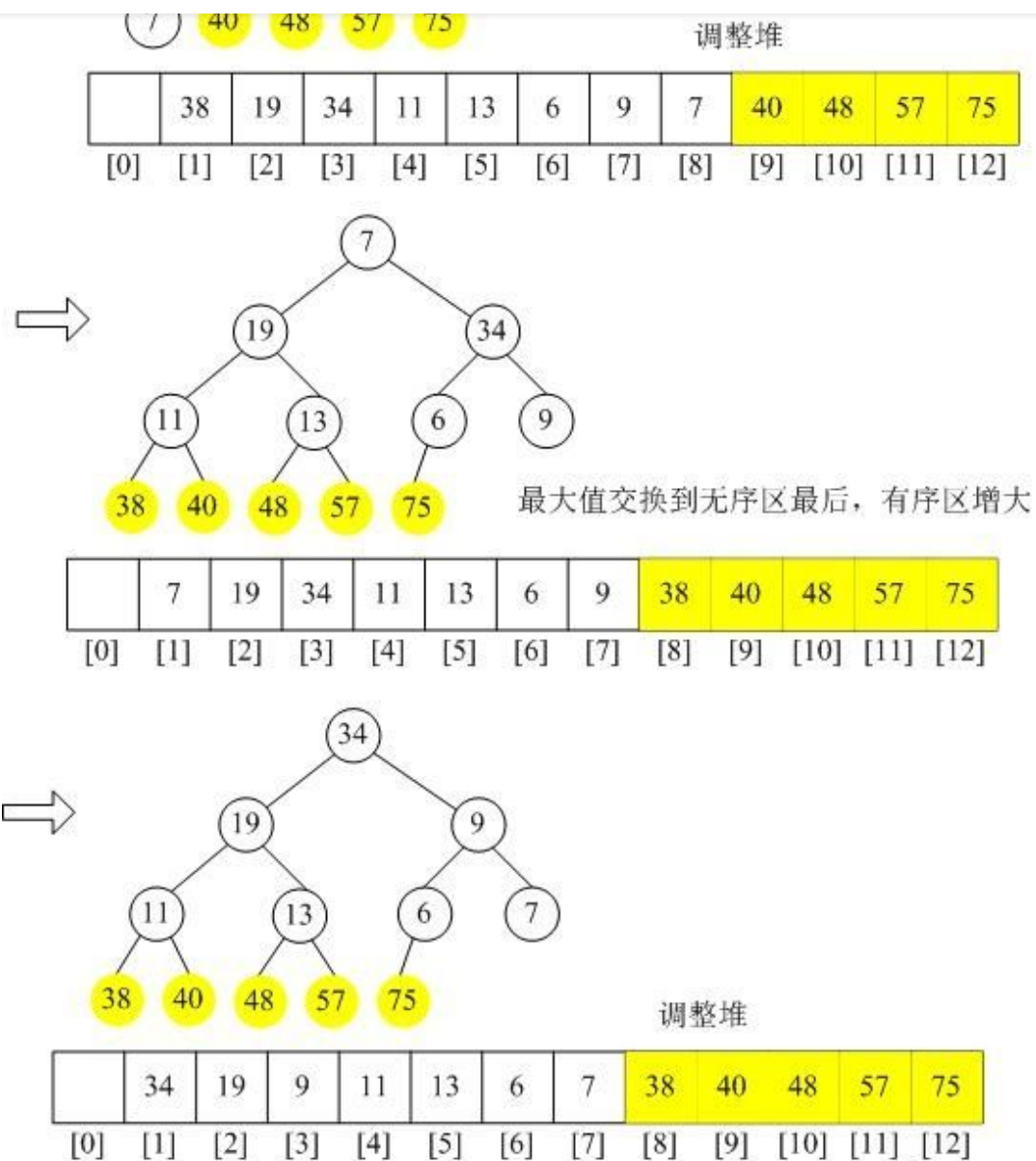


调整堆



最大值交换到无序区最后，有序区增大





知乎 @Zhou

赞同 162

4 条评论

分享

喜欢

收藏

申请转载

...

## 代码展示

代码引自[数据结构示例——堆排序过程](#)。

```
void HeapSort(RecType R[],int n)
{
    int i;
    RecType temp;
    // (1) 循环建立初始堆
    for (i=n/2; i>=1; i--)
        sift(R,i,n);
    //(2)进行n-1次循环,完成推排序
    for (i=n; i>=2; i--)
    {
        temp=R[1];        //将第一个元素同当前区间内R[1]对换
        R[1]=R[i];
        R[i]=temp;
        sift(R,1,i-1);    //筛选R[1]结点,得到i-1个结点的堆
    }
}
```

## 伪代码

引用自[cc.gatech.edu/classes/c...](https://cc.gatech.edu/classes/c...)

```
        exchange A[1] <-> A[i]
        heapsize <- heapsize -1
        Heapify(A, 1)
    }

# 创建堆A
BuildHeap(A) {
    heapsize <- length(A)
    for i <- floor( length/2 ) downto 1
        Heapify(A, i)
}

# 堆化
Heapify(A, i) {
    le <- left(i)
    ri <- right(i)
    if (le<=heapsize) and (A[le]>A[i])
        largest <- le
    else
        largest <- i
    if (ri<=heapsize) and (A[ri]>A[largest])
        largest <- ri
    if (largest != i) {
        exchange A[i] <-> A[largest]
        Heapify(A, largest)
    }
}
```



在下一篇文章中，将着重讲的是桶排序算法。

如果你觉得我的文章有用，顺手点个赞，关注下我的专栏或则留下你的评论吧！



编辑于 2018-11-05

「真诚赞赏，手留余香」

▲ 赞同 162 ▼

● 4 条评论

➤ 分享

♥ 喜欢

★ 收藏

📄 申请转载

...



## 文章被以下专栏收录



AnotherWorld

"Hello world!" From Another world.

## 推荐阅读

轻松搞定十大排序算法 |  
C++版 (上)

作者: opooc 链接:

<https://www.nowcoder.com/discuss>

来源: 牛客网 0、简介本文章, 是为了让读者会用十大排序算法。如果您对本文感兴趣, 欢迎关注我 😊。如对本文有任何疑问或...

牛客网

&lt;Quick Sort&gt;

## 排序算法-快速排序 (Java实现)

方北北

轻松搞定十大排序算法 |  
C++版 (下)

作者: opooc 链接:

<https://www.nowcoder.com/discuss>

来源: 牛客网 上篇链接: 牛客网: 轻松搞定十大排序算法 | C++版 (上) 4、希尔排序算法描述: 1959年Shell发明, 第一个突破 $O(n^2)$ ...

牛客网



发表于编程学习

## 常见数据结构算法总结

1、快速排序法 概念: 排序速度非常快, 采用分治思想 空间复杂度快速排序是一种原地排序, 只需要一个很小的栈作为辅助空间, 空间复杂度为  $O(\log n)$ , 所以适合在数据集比较大的时候使用。时间...

Mr.Do...

发表于数据分析成...

-  legal high 2020-04-26  
把抄的都去掉..还剩下啥....  
👍 12
-  meng hu 2020-04-04  
谢谢  
👍 赞
-  keykey 2019-12-23  
谢谢  
👍 赞
-  fuku 2019-11-08  
谢谢  
👍 赞