

[LintCode领扣题解 \(/problem\)](#) / 堆化 · Heapify

堆化 · Heapify

中文

[LintCode 版权所有 \(/problem/?tags=lintcode-copyright\)](#)[堆 \(/problem/?tags=heap\)](#)

描述

给出一个整数数组，堆化操作就是把它变成一个最小堆数组。

对于堆数组A，A[0]是堆的根，并对于每个A[i]，A[i * 2 + 1]是A[i]的左儿子并且A[i * 2 + 2]是A[i]的右儿子。

样例

样例 1

输入 : [3,2,1,4,5]
输出 : [1,2,3,4,5]
解释 : 返回任何一个合法的堆数组

挑战

O(n)的时间复杂度完成堆化

说明

什么是堆？什么是堆化？如果有很多种堆化的结果？

- 堆是一种数据结构，它通常有三种方法：push，pop 和 top。其中，“push”添加新的元素进入堆，“pop”删除堆中最小/最大元素，“top”返回堆中最小/最大元素。
- 把一个无序整数数组变成一个堆数组。如果是最小堆，每个元素A[i]，我们将得到A[i * 2 + 1] >= A[i]和A[i * 2 + 2] >= A[i]
- 返回其中任何一个。

在线评测地址: <https://www.lintcode.com/problem/heapify/> (<https://www.lintcode.com/problem/heapify/>)

收起题目描述 ^

语言类型

[ALL \(13\)](#)[python \(7\)](#)[java \(4\)](#)[cpp \(2\)](#)[上传题解](#)

令狐冲

更新于 10/12/2020, 1:04:56 PM

Heapify 的具体实现方法。时间复杂度为 $O(n)$ ，使用的是 sift-down 之所以是 $O(n)$ 是因为从第 $N/2$ 个位置开始往下 sift-down，那么就有大约 $N/4$ 个数在 sift-down 中最多交换 1 次， $N/8$ 个数最多交换 2 次， $N/16$ 个数最多交换 3 次。所以 $O(N/4 * 1 + N/8 * 2 + N/16 * 3 + \dots + 1 * \log N) = O(N)$

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param: A: Given an integer array
    @return: nothing
    """
    def heapify(self, A):
        for i in range(len(A) // 2, -1, -1):
            self.siftdown(A, i)

    def siftdown(self, A, index):
        n = len(A)
        while index < n:
            left = index * 2 + 1
            right = index * 2 + 2
            minIndex = index
            if left < n and A[left] < A[minIndex]:
                minIndex = left
            if right < n and A[right] < A[minIndex]:
                minIndex = right

            if minIndex == index:
                break

            A[minIndex], A[index] = A[index], A[minIndex]
            index = minIndex

```

👍 获赞 10

💬 2 条评论

你的口袋题库
2000+算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 7/25/2020, 12:45:02 PM

Given an integer array, heapify it into a min-heap array. For a heap array A, A[0] is the root of heap, and for each A[i], A[i * 2 + 1] is the left child of A[i] and A[i * 2 + 2] is the right child of A[i].

java

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
// Version Linpz
public class Solution {

```

```
/**
 * @param A: Given an integer array
 * @return: void
 */
private void siftDown(int[] A, int k) {
    while (k * 2 + 1 < A.length) {
        int son = k * 2 + 1;
        if (k * 2 + 2 < A.length && A[son] > A[k * 2 + 2])
            son = k * 2 + 2;
        if (A[son] >= A[k])
            break;

        int temp = A[son];
        A[son] = A[k];
        A[k] = temp;
        k = son;
    }
}

public void heapify(int[] A) {
    for (int i = (A.length - 1) / 2; i >= 0; i--) {
        siftDown(A, i);
    }
}
}

// Version 1: this cost O(n)
public class Solution {
    /**
     * @param A: Given an integer array
     * @return: void
     */
    private void siftDown(int[] A, int k) {
        while (k < A.length) {
            int smallest = k;
            if (k * 2 + 1 < A.length && A[k * 2 + 1] < A[smallest]) {
                smallest = k * 2 + 1;
            }
            if (k * 2 + 2 < A.length && A[k * 2 + 2] < A[smallest]) {
                smallest = k * 2 + 2;
            }
            if (smallest == k) {
                break;
            }
            int temp = A[smallest];
            A[smallest] = A[k];
            A[k] = temp;

            k = smallest;
        }
    }



    public void heapify(int[] A) {
        for (int i = A.length / 2; i >= 0; i--) {
            siftDown(A, i);
        } // for
    }
}

// Version 2: This cost O(nlogn)
public class Solution {
    /**
     * @param A: Given an integer array
     * @return: void
     */
    private void siftUp(int[] A, int k) {
        while (k != 0) {
```

```
        int father = (k - 1) / 2;
        if (A[k] > A[father]) {
            break;
        }
        int temp = A[k];
        A[k] = A[father];
        A[father] = temp;

        k = father;
    }
}

public void heapify(int[] A) {
    for (int i = 0; i < A.length; i++) {
        siftup(A, i);
    }
}
```

 获赞 3 15 条评论

令狐冲

更新于 7/8/2020, 8:27:56 PM

Given an integer array, heapify it into a min-heap array. For a heap array A, $A_0()$ is the root of heap, and for each $A_i()$, $A_{i * 2 + 1}()$ is the left child of $A_i()$ and $A_{i * 2 + 2}()$ is the right child of $A_i()$.

C++

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param A: Given an integer array
     * @return: void
     */
    void min_heapify(vector<int> &A, int i, int len){
        int l = 2 * i + 1;
        int r = 2 * i + 2;
        int largest = i;
        if( l < len)
            if(A[l] < A[i])
                largest = l;

        if( r < len )
            if( A[r] < A[largest])
                largest = r;

        if(largest != i) {
            swap(A[i], A[largest]);
            min_heapify(A, largest, len);
        }
    }
    void heapify(vector<int> &A) {
        // write your code here
        int len = A.size();
        for(int i = len / 2; i >= 0; i--)
            min_heapify(A, i, len);
    }
};
```

👍 获赞 0

💬 2 条评论



Tin

更新于 7/6/2020, 11:27:52 PM

sift_up很简单是了解heap的入门。这题要是碰到了, 就是要考你sift_down, 所以, 你要是不会写, 就背, 要背, 就背这个最简洁的。

另外, 你要能答的出为什么sift_up是 $O(n\log N)$, 而sift_down是 $O(N)$ 。

另: 是sift_down, 次优是sift_up。不是 shift_down, shift_up, 不是 bubble_up, 不是 percolate_up。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:

    def heapify(self, nums):

        for i in reversed(range((len(nums))//2)):
            self.sift_down(nums, i)

    def sift_down(self, nums, index):

        n = len(nums)
        while index * 2 + 1 < n:

            son_index = index * 2 + 1
            if son_index + 1 < n and nums[son_index] > nums[son_index+1]:
                son_index = son_index + 1
            if nums[son_index] >= nums[index]:
                break
            nums[index], nums[son_index] = nums[son_index], nums[index]
            index = son_index

```

👍 获赞 2 💬 添加评论



九章用户GE5JQH

更新于 6/9/2020, 7:03:50 AM

使用java解答的作法, 從尾到頭掃一遍, 如果遇到本身的值比parent element的值小的 就一直跟parent element的值交換, 直到交換到最頂層或是本身的值大於parent element的值

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param: A: Given an integer array
    @return: nothing
    """
    def heapify(self, nums):
        for i in range(1, len(nums)):
            self.percolate_up(i, nums)

    def percolate_up(self, i, nums):
        while (i - 1) // 2 >= 0 and nums[i] < nums[(i - 1) // 2]:
            nums[i], nums[(i - 1) // 2] = nums[(i - 1) // 2], nums[i]
            i = (i - 1) // 2

```

👍 获赞 2 💬 添加评论

**TONG**

更新于 6/9/2020, 7:03:50 AM

java中siftdown的版本, 检查前一半的元素与son的大小关系, 据此update smallest的位置, 交换, 不断重复

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: A: Given an integer array
    @return: nothing
    """
    def heapify(self, A):
        # write your code here
        for i in range(len(A) / 2, -1, -1):
            self.siftdown(A, i)

    def siftdown(self, A, pos):
        while pos < len(A):
            smallest = pos
            if pos * 2 + 1 < len(A) and A[pos * 2 + 1] < A[smallest]:
                smallest = pos * 2 + 1
            if pos * 2 + 2 < len(A) and A[pos * 2 + 2] < A[smallest]:
                smallest = pos * 2 + 2
            if pos == smallest:
                break

            A[smallest], A[pos] = A[pos], A[smallest]

            pos = smallest
```

👍 获赞 2

💬 添加评论

**EricC**

更新于 6/9/2020, 7:03:54 AM

丑陋版本的heapify, 但是复杂度确实是 O(n) 具体请看code中的注解, 含计算。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Revised heapify with recursion.    O (n)
 *
 * Heapify down from A.length -> 0.
 *   n/2 - n : Leafs, Do nothing.
 *   n/4 - n/2: heapify down 1 level maximum (find the min from left and right, then swap).
 *   n/8 - n/4: heapify down 2 level maximum
 *   ...
 *   0 :    heapify down logn maximum.
 *
 * Complexity: n/2 * 1 + n/4 * 2 + ... + n/(2 ^ h) * lg h
 *             = n (1/2 + 1/4 ...)
 *             = n.
 */
public class Solution {
    /**
     * @param A: Given an integer array
     * @return: nothing
     */
    public void heapify(int[] A) {
        heapifyDown(A, A.length/4, A.length/2);
    }

    private void heapifyDown(int[] A, int start, int end) {
        if (end == 0) { // this gaurantees start == 0 was not excluded.
            return;
        }

        for (int i = start; i < end; i++) {
            int j = i;
            while (j < A.length) { // if left or right does not exist, make them largest
                int left = (2 * j + 1) < A.length ? A[2 * j + 1] : Integer.MAX_VALUE;
                int right = (2 * j + 2) < A.length ? A[2 * j + 2] : Integer.MAX_VALUE;

                // case 1, already in good shape
                if (A[j] < Math.min(left, right)) {
                    break;
                }
                // case 2, need to find the min, and swap
                int min_index = left < right ? (2 * j + 1) : (2 * j + 2);
                swap(A, j, min_index);
                j = min_index;
            }
        }

        heapifyDown(A, start/2, start); // current level is done, heapify next level
    }

    private void swap(int[] A, int a, int b) {
        int temp = A[a];
        A[a] = A[b];
        A[b] = temp;
    }
}

```

👍 获赞 1 💬 添加评论

**Jet**

更新于 6/9/2020, 7:03:53 AM

sift up: $O(n \log n)$ 新构建一个数组，依次从需要 heapify 的数组取值出来，加入新数组，并 check 新加入的数是否比父亲节点小，若小，交换，直至到 heap 顶点

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param A: Given an integer array
     * @return: nothing
     */
    void heapify(vector<int> &A) {
        // write your code here
        vector<int> result;
        for(int i=0;i<A.size();i++){
            result.push_back(A[i]);
            heap(result);
        }
        A=result;
    }
    void heap(vector<int>& A){
        if(A.size()==0||A.size()==1){
            return;
        }
        int last=A.size()-1;
        while(last>0){
            if(A[last]>A[(last-1)/2]){
                return;
            }
            int temp=A[last];
            A[last]=A[(last-1)/2];
            A[(last-1)/2]=temp;
            last=(last-1)/2;
        }
    }
};
```

👍 获赞 1

💬 添加评论

**九章用户U6HP8S**

更新于 8/4/2020, 9:52:18 PM

sift down iterative写法，less is more

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {

    private void siftDown(int[] nums, int idx) {
        int child;
        while (2 * idx + 1 < nums.length) {
            child = 2 * idx + 1; // set child to be the left child

            // compare the left child with the right child to find the minimum one
            if (child + 1 < nums.length && nums[child + 1] < nums[child])
                child += 1;

            if (nums[idx] <= nums[child]) return;

            // swap their values
            nums[idx] ^= nums[child];
            nums[child] ^= nums[idx];
            nums[idx] ^= nums[child];

            // continue to sift down
            idx = child;
        }
    }

    /**
     * @param A: Given an integer array
     * @return: nothing
     */
    public void heapify(int[] A) {
        // we don't need to sift down the leaf nodes since they trivially satisfy
        // the heap property already. Only need to start from the second last level
        for (int i = (A.length - 1) / 2; i >= 0; --i) {
            siftDown(A, i);
        }
    }
}
```

👍 获赞 0

💬 1 条评论



九章用户FNSAOY

更新于 6/9/2020, 7:04:21 AM

Python3 的代码

答案中使用了percolateDown(), 自底向上。由于所有的叶节点本身都能够被看成是一个min heap, 所以只需要从最后一个有children的节点开始(此节点在array中的index是: $(\text{len}(\text{input_array}) - 1 - 1) // 2$), 一直向input array的左边遍历(即遍历到根节点)。每一次循环做了如下的事情: 由于对于当前节点(currentNode)来说, 其left sub heap和right sub heap本身已经是min heap了, 所以将left sub heap + currentNode + right sub heap组合起来建立一个最小堆的过程, 就是针对currentNode调用一次percolateDown的过程。

时间复杂度: $O(n)$ 如果是使用percolateUp(), 将新加入的节点向上浮动建堆的话, 时间复杂度是: $O(n \log n)$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param: A: Given an integer array
    @return: nothing
    """
    def heapify(self, A):
        # O(n) time, need to use percolateDown()
        # O(nlogn) time, use offer/append and percolateUp()

        startIndex = (len(A) - 1 - 1) // 2
        length = len(A)
        currentIndex = startIndex
        while currentIndex >= 0:

            self.percolateDown(A, currentIndex, startIndex, length)

            currentIndex -= 1

    def percolateDown(self, A, index, startIndex, length):
        currentIndex = index
        while currentIndex <= startIndex:
            # if A[currentIndex] < min(A[currentIndex * 2 + 1], A[currentIndex * 2 + 2]):
            #     return
            if currentIndex * 2 + 2 < length:
                if A[currentIndex * 2 + 1] < A[currentIndex] or A[currentIndex * 2 + 2] < A[currentIndex]:
                    if A[currentIndex * 2 + 1] < A[currentIndex * 2 + 2]:
                        A[currentIndex * 2 + 1], A[currentIndex] = A[currentIndex], A[currentIndex * 2 + 1]
                        currentIndex = currentIndex * 2 + 1
                    else:
                        A[currentIndex * 2 + 2], A[currentIndex] = A[currentIndex], A[currentIndex * 2 + 2]
                        currentIndex = currentIndex * 2 + 2
            else:
                return
            elif currentIndex * 2 + 1 < length:
                if A[currentIndex * 2 + 1] < A[currentIndex]:
                    A[currentIndex * 2 + 1], A[currentIndex] = A[currentIndex], A[currentIndex * 2 + 1]
                    currentIndex = currentIndex * 2 + 1
            else:
                return
            return
```

👍 获赞 0

💬 添加评论



九章用户XM6F39

更新于 6/9/2020, 7:04:18 AM

Modified from the Java Version Time Complexity: O(nlogn)

👍 获赞 0 💬 添加评论

<https://www.jiuzhang.com/problem/heapify/> Page 12 of 15

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param A: Given an integer array
     * @return: nothing
     */
    public void heapify(int[] A) {
        // 从堆的倒数第二层开始, 堆的倒数第一层都是单个node, 必须是堆。
        for (int i = (A.length - 2) / 2; i >= 0; i--) {
            shiftDown(A, i);
        }
    }

    private void shiftDown(int[] A, int index) {
        int len = A.length;
        int leftIndex = index * 2 + 1;
        int rightIndex = leftIndex + 1;

        if (rightIndex < len && A[rightIndex] < A[leftIndex] && A[rightIndex] < A[index]) {
            swap(A, rightIndex, index);
            shiftDown(A, rightIndex);
        } else if (leftIndex < len && A[leftIndex] < A[index]) {
            swap(A, leftIndex, index);
            shiftDown(A, leftIndex);
        }
    }

    private void swap(int[] A, int a, int b) {
        int temp = A[a];
        A[a] = A[b];
        A[b] = temp;
    }
}
```

邀请有礼
invitation/shi



👍 获赞 0 💬 添加评论



Teddy

更新于 6/9/2020, 7:04:02 AM

来个递归版本的 省掉那些罗里罗嗦的while 循环 递归深度: logN

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:
    """
    @param: A: Given an integer array
    @return: nothing
    """
    def heapify(self, A):
        # write your code here
        for i in range(len(A)):
            self.siftUp(A, i)

        # for i in range(len(A) - 1, -1, -1):
        #     self.siftDown(A, i)

    def siftDown(self, A, k):

        if 2 * k + 1 >= len(A):
            return

        left, right = 2 * k + 1, 2 * k + 2

        next_idx = left

        if right < len(A) and A[right] < A[left]:
            next_idx = right

        if A[k] < A[next_idx]:
            return

        A[k], A[next_idx] = A[next_idx], A[k]
        self.siftDown(A, next_idx)

    def siftUp(self, A, k):

        if k <= 0:
            return

        parent = (k - 1) // 2
        if A[parent] < A[k]:
            return

        A[parent], A[k] = A[k], A[parent]
        self.siftUp(A, parent)
```

👍 获赞 0 💬 添加评论

进阶课程

| | | | |
|---|--|--|------------------------------------|
| 直播+互动 | 直播+互动 | 直播+互动 | 互动课 |
| 九章算法班 2021 版 8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22.... | 系统架构设计 System Design 2021 版 成为百万架构师必上。30 课时带你快速掌握 18大系统架构设计知识点与面... | 九章算法面试高频题冲刺班 每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer | 面向对象设计 OOD 应届生及亚马逊面试必考，IT求职必备基础 |