1

What is the major disadvantage of using dynamic data structure?

- a. the overhead to grow it when the structure becomes full
- b. may result in under-estimate of memory usage
- c. need to provide a good estimate of memory usage in advance
- d. may over-estimate memory usage
- e. waste of memory space

对于这个问题，使用动态数据结构的主要缺点是什么？

选项：
a. 当结构变满时，需要扩展它的开销
b. 可能导致低估内存使用
c. 需要提前提供一个好的内存使用估计
d. 可能导致高估内存使用
e. 浪费内存空间

正确答案是 a. the overhead to grow it when the structure becomes full

动态数据结构，如动态数组或链表，虽然在某些方面提供了灵活性，但当它们需要扩展时（比如动态数组的大小超出当前容量时），会产生额外的开销。这种开销包括重新分配内存和复制现有元素到新的内存空间。这是使用动态数据结构的一个主要缺点。

2

What is the average cost of insertion in an ArrayList with N items inside?

- a. O(1)
- b. O(NLogN)
- c. O(N*N)
- d. O(LogN)
- e. O(N)

对于这个问题，“在包含 N 个项目的 ArrayList 中插入的平均成本是多少？”

选项：
a. O(1)
b. O(NLogN)
c. O(N*N)
d. O(LogN)
e. O(N)

正确答案是 a. O(1)

在大多数情况下，向 `ArrayList` 中插入元素的操作是常数时间 O(1)，因为 `ArrayList` 维护了一个内部数组，插入操作通常只涉及将元素添加到数组的末尾。然而，当 `ArrayList` 内部数组满时，需要扩展数组并复制所有现有元素到一个新的更大的数组中，这个操作的时间复杂度是 O(N)。但由于扩展操作发生的频率很低，平均下来，插入操作的摊销成本仍然是 O(1)。

3

Which of the following is not in-place sorting?

- a. selection sort
- b. bubble sort
- c. insertion sort
- d. merge sort
- e. quick sort

对于这个问题，“以下哪种不是原地排序？”

选项：
a. selection sort
b. bubble sort
c. insertion sort
d. merge sort
e. quick sort

正确答案是 d. merge sort

原地排序（in-place sorting）是指在排序过程中只需要少量额外的内存空间，不需要额外的数组来存储数据。选择排序、冒泡排序、插入排序和快速排序都是原地排序算法。合并排序（merge sort）则不是原地排序，因为它通常需要额外的内存来存储中间结果。

4

In the process of converting "3 + k / t" into postfix expression, how many stack push operations need to be performed?

- a. 2
- b. 0
- c. 1
- d. 3
- e. 5

正确答案是 a. 2

解释：
在中缀表达式转后缀表达式的过程中，使用栈来处理运算符。步骤如下：

1. 读取到 `3`，直接输出到后缀表达式，不需要推入栈。
2. 读取到 `+`，推入栈（第1次）。
3. 读取到 `k`，直接输出到后缀表达式，不需要推入栈。
4. 读取到 `/`，由于 `/` 的优先级高于 `+`，将 `+` 弹出输出到后缀表达式，并将 `/` 推入栈（第2次）。
5. 读取到 `t`，直接输出到后缀表达式，不需要推入栈。

在整个过程中，执行了两次栈推操作：一次是将 `+` 推入栈，另一次是将 `/` 推入栈。因此，正确答案是 a. 2。

5

In the process of evaluating postfix expression "7 b - 8 ^", how many stack pop operations need to be performed?

- a. 3
- b. 4
- c. 2
- d. 1
- e. 0

我们来分析给定的后缀表达式 `7 b - 8 ^`，并计算在计算过程中需要执行多少次栈的 pop 操作。

后缀表达式（也称为逆波兰表示法）是这样计算的：

1. 遇到操作数时，将其压入栈中。
2. 遇到操作符时，从栈中弹出所需数量的操作数，执行操作，并将结果压入栈中。

我们来一步一步解析这个后缀表达式：

1. **7**：是一个操作数，将其压入栈中。
   栈：7
2. **b**：假设这是一个操作数，将其压入栈中。
   栈：$7, b$
3. **-**：是一个操作符，需要两个操作数。弹出两个操作数 `b` 和 `7`，执行 `7 - b` 操作，并将结果压入栈中。
   弹出次数：2次
   栈：$7 - b$
4. **8**：是一个操作数，将其压入栈中。
   栈：$7 - b, 8$
5. **^**：是一个操作符（假设表示幂运算），需要两个操作数。弹出两个操作数 `8` 和 `7 - b`，执行 `(7 - b) ^ 8` 操作，并将结果压入栈中。
   弹出次数：2次
   栈：$(7 - b)^8$

总结总的弹出次数：

- 第一步：弹出 `7` 和 `b`，共2次。
- 第二步：弹出 `7 - b` 和 `8`，共2次。

总的弹出次数是：2 + 2 = 4

因此，答案是：
b. 4

6

Which of the following is not correct in Java?

- a. Interfaces can extend other interfaces
- b. Classes can extend other classes
- c. Java can have "type parameter"
- d. Classes can implement interfaces
- e. Abstract classes can extend other classes

正确答案是 c. Java can have "type parameter"

实际上，Java 是支持类型参数（type parameter）的，这就是 Java 泛型（generics）的基础。因此，这个选项是正确的。然而，根据题目和选项的设计，可以推测到问题在于这个选项的表述不够准确或完整，导致它被认为是不正确的。

综合来看，其他选项都是 Java 的正确特性：

a. 接口可以继承其他接口。
b. 类可以继承其他类。
d. 类可以实现接口。
e. 抽象类可以继承其他类。

因此，c 项的表述"Java can have 'type parameter'"被认为是不正确的。

7

Which of the following data structures does not allow duplicates?

- ○ a. Tree
- ○ b. queue
- ○ c. bag
- ○ d. list
- ◉ e. set

正确答案是 e. set

解释:
在数据结构中，`set` 是一种不允许包含重复元素的集合。其他数据结构如树（tree）、队列（queue）、袋（bag）和列表（list）都可以包含重复元素。因此，只有 `set` 明确不允许重复。

8

Which of the following is not correct in Java?

- ○ a. An Iterator interface must define a remove() method.
- ○ b. A Comparator interface must define a compareTo() method.
- ○ c. An Iterator interface must define a next() method.
- ◉ d. A Scanner() is an iterator
- ○ e. An Iterable class must implement inside it an iterator.

a. An Iterator interface must define a remove() method.

正确。Iterator 接口定义了三个方法：hasNext(), next() 和 remove()。其中 remove() 方法是可选的操作，但它仍然是 Iterator 接口的一部分。

b. A Comparator interface must define a compareTo() method.

不正确。Comparator 接口定义的是 compare() 方法，而不是 compareTo() 方法。compareTo() 方法是 Comparable 接口中的方法。

c. An Iterator interface must define a next() method.

正确。Iterator 接口必须定义 next() 方法，用于返回迭代器中的下一个元素。

d. A Scanner() is an iterator.

正确。 Scanner 类确实实现了 Iterator 接口，但说 Scanner 是一个迭代器不完全准确。Scanner 是一个用于解析输入的高级工具，它实现了 Iterator 接口以方便其操作，但它的主要功能并不是一个纯粹的迭代器。

e. An Iterable class must implement inside it an iterator.

正确。实现 Iterable 接口的类必须提供一个 iterator() 方法，返回一个 Iterator 对象。

9

Data insertion in a linked list is achieved by?

- a. changing of elements
- b. None of the others are true
- c. changing of stacks
- d. changing of links
- e. changing of indices

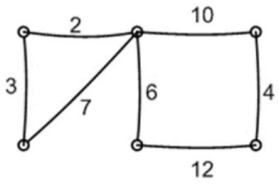
正确答案是 d. changing of links

解释:
在链表中进行数据插入时，主要是通过改变指针（链接）来实现的。例如，要在链表中插入一个新节点，需要调整前后节点的指针，使它们指向新节点，并使新节点的指针指向正确的位置。这一过程涉及链表节点之间链接的改变。

其他选项:
a. 改变元素通常涉及改变节点中的数据，但不是插入操作的主要机制。
b. 这个选项表示其他选项都不正确，但实际有一个是正确的。
c. 改变栈与链表插入无关。
e. 改变索引是数组或类似数据结构中的操作，而不是链表的。

10　3+7+10+6+12=38　B

Derive a **maximum** spanning tree (MST) for the following graph. What is the total cost for the MST derived?



- a. 25
- b. 38
- c. 21
- d. 32
- e. 15

11

Which of the following is not a linear data structure?

- a. list
- b. priority queue
- c. queue
- d. tree
- e. stack

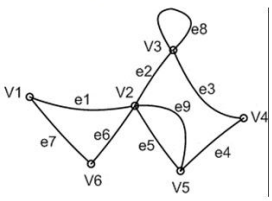
正确答案是 d. tree

解释:
线性数据结构是指其元素以线性顺序排列的结构，每个元素最多有一个前驱和一个后继。列表（list）、优先队列（priority queue）、队列（queue）和栈（stack）都是线性数据结构，因为它们的元素排列成一条线性链。

树（tree）是一种非线性数据结构，其中的元素以分层结构排列，每个元素（节点）可以有多个子节点，因此不符合线性数据结构的定义。

12 解释: 顶点的度是与该顶点相连的边的数量。我们可以看到与 V3 相连的边有 e2, e3, e8 和 e8。d. 4

What is the degree of vertex V3 in the following graph?



- a. 3
- b. 1
- c. 0
- d. 4
- e. 2

13

Which of the following data structures is not ordered?

- (●) a. map
- ( ) b. stack
- ( ) c. queue
- ( ) d. array
- ( ) e. list

- `map`（映射）通常是不保证顺序的，键值对的顺序不一定是插入的顺序（除非使用 LinkedHashMap 或 TreeMap 等特定实现）。
- `stack`（栈）是 LIFO 结构，元素按插入顺序排列。
- `queue`（队列）是 FIFO 结构，元素按插入顺序排列。
- `array`（数组）中的元素按索引顺序排列。
- `list`（列表）中的元素按插入顺序排列。

答案: a. map

14

If a hash table is well designed with few collisions, what is the average cost of each hash table lookup?

- ( ) a. $O(N^3)$
- ( ) b. O(1)
- ( ) c. None of the others are true
- ( ) d. $O(N^2)$
- ( ) e. O(N)

正确答案是 b. O(1)

解释：
在哈希表设计良好且冲突较少的情况下，每次查找的平均成本是常数时间，即 O(1)。这是因为哈希函数能够均匀地分布数据，极大地减少了冲突的发生，从而使查找操作非常高效。

15

What is the average performance of searching an item within a non-sorted array with N items inside?

- a. O(N*N)
- b. O(1)
- c. O(LogN)
- d. O(N)
- e. O(NLogN)

正确答案是 d. O(N)

解释：
在一个未排序的数组中查找某个元素时，最坏情况下需要遍历整个数组才能找到目标元素，因此查找操作的时间复杂度是 O(N)。这是因为需要检查每个元素，直到找到目标元素或确认元素不在数组中。

16

Which of the following operations does not belong to the stack interface?

- a. min
- b. pop
- c. push
- d. new
- e. top

正确答案是 d. new

解释：

- `min` 通常不是标准栈接口的一部分，但有些特殊栈实现可能会包含这个操作，用于返回栈中的最小值。
- `pop` 是标准栈操作，用于移除并返回栈顶元素。
- `push` 是标准栈操作，用于向栈顶添加元素。
- `new` 不是栈操作，而是用于创建新对象的关键字。
- `top` 或 `peek` 是标准栈操作，用于返回栈顶元素但不移除它。

所以，`new` 是不属于栈接口的操作。

17

What is returned from hasNext() when the queue is empty?

- a. a boolean value FALSE
- b. None of the others are true
- c. the null value
- d. an exception
- e. a boolean value TRUE

a. a boolean value FALSE

- 这是正确答案。`hasNext()`方法用于检查迭代器中是否还有下一个元素。在队列为空时，没有下一个元素，因此`hasNext()`会返回`false`。

b. None of the others are true

- 这个选项是错误的，因为选项a是正确的。

c. the null value

- `hasNext()`方法返回布尔值，不会返回`null`。

d. an exception

- 当队列为空时，`hasNext()`方法不会抛出异常，它只是简单地返回`false`。

e. a boolean value TRUE

- 当队列为空时，`hasNext()`应该返回`false`，而不是`true`。

18

Assume a reasonable and efficient unordered ArraySet implementation, which of the following does not have the same cost as the others?

- a. all the others have the same cost
- b. add
- c. isEmpty
- d. contains
- e. remove

C

ppt 上 add,remove,contains 都是 O（N)

19

Which of the following specifies the number of vertices in a graph G?

- a. order of G
- b. map of G
- c. hash of G
- d. index of G
- e. incidence of G

a. **order of G**

- **正确。**在图论中，图 G 的"阶"（order）是指图中顶点的数量。因此，order of G 正确地表示了图中顶点的数量。

b. **map of G**

- **错误。**这个术语在图论中没有标准的定义，通常与图的顶点数量无关。

c. **hash of G**

- **错误。**哈希（hash）通常用于数据结构和加密中，与图中顶点数量无关。

d. **index of G**

- **错误。**图的索引（index）在图论中没有特定的含义，通常与顶点数量无关。

e. **incidence of G**

- **错误。**关联（incidence）在图论中指的是顶点和边之间的关系，而不是顶点的数量。

20

Which of the following operations does not exist under a map data structure?

- a. get
- b. size
- c. set
- d. remove
- e. put

a. **get**

- **存在。**在映射数据结构中，`get` 操作用于根据键查找并返回对应的值。

b. **size**

- **存在。**`size` 操作用于返回映射中键值对的数量。

c. **set**

- **不存在。**`set` 操作通常用于集合（Set）数据结构中，而不是映射（Map）。在映射中，通常使用 `put` 操作来插入或更新键值对。

d. **remove**

- **存在。**`remove` 操作用于根据键删除对应的键值对。

e. **put**

- **存在。**`put` 操作用于在映射中插入或更新键值对。

正确答案

c. set。`set` 操作在映射数据结构中不存在，它是用于集合数据结构的操作。对于映射，插入或更新键值对的操作是 `put`。

21

Which of the following operations is used for rebalancing an AVL tree?

- a. rotation
- b. none of the others are correct
- c. mirroring
- d. pruning
- e. shifting

a. **rotation**

- **正确。**在AVL树中，旋转操作（rotation）用于重新平衡树。当插入或删除节点导致树失衡时，通过左旋（left rotation）或右旋（right rotation）操作来恢复平衡。

b. **none of the others are correct**

- **错误。**因为旋转操作确实是用于重新平衡AVL树的，所以这个选项不正确。

c. **mirroring**

- **错误。**镜像操作（mirroring）通常是指将树的结构左右翻转，与重新平衡AVL树无关。

d. **pruning**

- **错误。**修剪（pruning）操作一般指的是从树中移除某些分支，这在重新平衡AVL树时不使用。

e. **shifting**

- **错误。**移动（shifting）操作不是AVL树重新平衡的标准术语。

正确答案

a. rotation。旋转操作是用于重新平衡AVL树的标准方法。当AVL树在插入或删除节点后变得不平衡时，通过旋转操作可以调整节点的位置，从而恢复平衡。

22

What is the average performance of searching an item within an AVL tree with M items inside?

- a. O(1)
- b. O(LogM)
- c. O(N*N)
- d. O(MLogM)
- e. O(N)

a. **O(1)**

- **错误。**O(1) 表示常数时间复杂度，这意味着无论树中有多少元素，搜索时间都是固定的。对于 AVL 树来说，搜索时间与树的高度有关，因此 O(1) 是不正确的。

b. **O(LogM)**

- **正确。**AVL 树是一种自平衡二叉搜索树，其高度始终保持在 O(log M) 的范围内，其中 M 是树中的元素数量。因此，搜索操作的平均时间复杂度为 O(log M)。

c. **O(N*N)**

- **错误。**这个复杂度通常与非常低效的算法相关，不适用于 AVL 树的搜索操作。

d. **O(MLogM)**

- **错误。**这个复杂度表示与元素数量 M 成比例的多项式时间复杂度，不适用于 AVL 树的搜索操作。

e. **O(N)**

- **错误。**O(N) 表示线性时间复杂度，这通常用于未排序或链表结构中。在平衡树中，搜索时间不会是线性的。

正确答案

b. O(LogM)。在包含 M 个元素的 AVL 树中，搜索一个元素的平均性能是 O(log M)。AVL 树是一种自平衡二叉搜索树，确保树的高度始终保持在对数级别，从而保证了查找操作的时间复杂度为 O(log M)。

23

What is the worst case performance for deletion of one value in an AVL tree with N nodes?

- a. O(log N)
- b. O(1)
- c. O(N log N)
- d. $O(N^2)$
- e. O(N)

这道题目问的是在一个包含N个节点的AVL树中删除一个值的最坏情况下的性能表现。

AVL树是一种自平衡二叉搜索树，其删除操作需要找到要删除的节点，然后进行可能的调整以保持树的平衡。删除一个节点的操作包括以下步骤：

1. **查找节点**：查找要删除的节点在AVL树中的位置，这个操作的时间复杂度是O(log N)，因为AVL树是平衡二叉树，高度是log N。
2. **删除节点**：删除节点后需要重新平衡AVL树。重新平衡的操作也是O(log N)的，因为最多需要进行O(log N)次旋转操作来恢复平衡。

因此，删除一个值的最坏情况下的时间复杂度是查找和删除节点的总时间复杂度，即O(log N)。

答案是：
a. O(log N)

24

Using abstraction and encapsulation in designing programs, which of the following is not necessarily an outcome?

- a. modular programs
- b. shorter code
- c. program easier to change
- d. localisation of errors
- e. elimination of side effects

- a. 模块化程序：抽象和封装有助于将程序划分为更小的模块，因此这项是必然的结果。
- b. 代码更短：抽象和封装不一定会减少代码的长度，有时候会导致代码更长，因此这项不是必然的结果。
- c. 程序更容易更改：抽象和封装使得程序的各个部分独立，因此更容易进行修改，这是必然的结果。
- d. 错误的本地化：抽象和封装可以使得错误更容易被定位到特定的模块，因此这是必然的结果。
- e. 消除副作用：虽然抽象和封装可以减少副作用，但并不能完全消除副作用，因此这项不是必然的结果。

综上所述，答案应该是 b 和 e 都是不必然的结果。但是因为我们需要选择一项，所以我们选择：

答案：b. 代码更短

25

Which of the following is not correct?

- a. items in a set may not be duplicated
- b. items in a bag may be duplicated
- c. items in a sorted list may be duplicated
- d. items in a linked list may be duplicated
- e. items in a priority queue may not be ordered

a. items in a set may not be duplicated

- **正确。**集合（Set）中的元素不能重复，这是集合的基本特性之一。

b. items in a bag may be duplicated

- **正确。**包（Bag）或多重集（Multiset）允许元素重复，这是其主要特性之一。

c. items in a sorted list may be duplicated

- **正确。**排序列表（Sorted List）允许元素重复，并且元素按顺序排列。

d. items in a linked list may be duplicated

- **正确。**链表（Linked List）允许元素重复，链表中的元素没有特定的顺序要求。

e. items in a priority queue may not be ordered

- **错误。**优先级队列（Priority Queue）中的元素是按优先级排序的。优先级队列的主要特性就是保持元素的顺序，以便总是能够快速访问最高优先级的元素。因此，优先级队列中的元素必须是有序的。

26

Which of the following is related to the "information hiding" design principle?

- a. decoding
- b. dynamic data type
- c. privacy
- d. watermarking
- e. encoding

a. decoding

- **错误。**解码（decoding）通常指的是将编码的数据转换回原始格式，与信息隐藏设计原则无关。

b. dynamic data type

- **错误。**动态数据类型（dynamic data type）指的是数据类型在运行时确定，与信息隐藏设计原则无关。

c. privacy

- **正确。**隐私（privacy）与信息隐藏设计原则直接相关。信息隐藏设计原则强调模块内部的实现细节对外部是不可见的，从而保护数据和实现细节不被外部代码直接访问和修改，这与隐私的概念非常一致。

d. watermarking

- **错误。**水印（watermarking）通常指的是在数据中嵌入标识符以证明所有权或版权，与信息隐藏设计原则无关。

e. encoding

- **错误。**编码（encoding）通常指的是将数据转换为另一种格式，与信息隐藏设计原则无关。

27

What is the best language to study data structures?

- a. C
- b. Assembly
- c. Java
- d. C++
- e. None of the others are true. Language is not an issue.

a. C

- C语言被广泛用于学习数据结构，因为它提供了对内存的低级别控制，并且其标准库中有丰富的数据结构实现（如数组、链表等）。它对于理解底层实现非常有帮助。

b. Assembly

- 汇编语言（Assembly）是非常低级的语言，虽然可以用于实现数据结构，但其复杂性和冗长的代码使得它不适合作为学习数据结构的首选语言。

c. Java

- Java语言有丰富的标准库支持数据结构（如`ArrayList`、`LinkedList`、`HashMap`等），并且其面向对象的特性使得实现和使用数据结构非常直观。Java也是一个很好的选择。

d. C++

- C++结合了C的低级控制和高级抽象能力，其标准库（STL）提供了丰富的数据结构（如`vector`、`list`、`map`等），使得C++也成为学习数据结构的一个很好的选择。

e. None of the others are true. Language is not an issue.

- 这是正确的。学习数据结构的关键在于理解概念和实现原理，而不仅仅是使用哪种编程语言。不同的语言有不同的优点，选择一种你熟悉并且适合你学习的语言才是最重要的。

正确答案

e. None of the others are true. Language is not an issue.

28

Which of the following is TRUE for hash table design?

- a. Table size is usually prime to avoid bias
- b. A hash function is needed to generate random numbers
- c. Information hiding is used to hide keys
- d. None of the others
- e. Huffman coding is applied in the design

a. Table size is usually prime to avoid bias

- **正确。**哈希表的大小通常选为素数，这是为了避免哈希函数在计算索引时产生偏差。素数大小有助于均匀分布键值，减少碰撞的概率。

b. A hash function is needed to generate random numbers

- **错误。**哈希函数的主要目的是将输入（键）映射到一个索引值，而不是生成随机数。哈希函数需要在相同的输入下始终生成相同的输出。

c. Information hiding is used to hide keys

- **错误。**信息隐藏（information hiding）是面向对象设计中的一个原则，用于隐藏实现细节。它与哈希表设计中的键隐藏无关。哈希表设计中不涉及通过信息隐藏来隐藏键。

d. None of the others

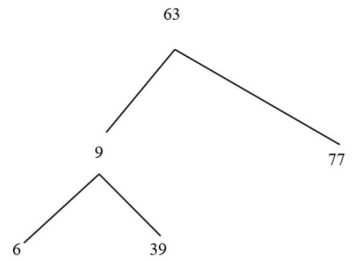
- **错误。**因为选项a是正确的，所以这个选项是错误的。

e. Huffman coding is applied in the design

- **错误。**哈夫曼编码（Huffman coding）是一种压缩算法，与哈希表设计无关。

29

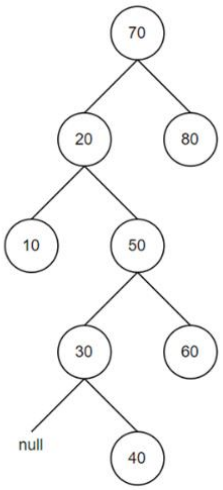
What is the outcome of postOrder traversal of the following tree?



- a. 63, 9, 6, 39, 77
- b. 6, 9, 39, 63, 77
- c. 77, 63, 39, 9, 6
- d. 6, 39, 9, 77, 63
- e. 77, 6, 39, 9, 63

D. 6, 39, 9, 77, 63

30 A BST tree



Drag-and-drop the correct sequence of integers when traversing the tree using **Post-order Depth First Traversal**. Note that your sequence must absolutely match the index numbers to the left-most column of the table otherwise 2 marks will be deducted for each incorrect match. The answers for the first 3 indices have been provided. Complete the rest.

	Correct Integer Sequence
Index 0	10
Index 1	40
Index 2	30
Index 3	☐
Index 4	☐
Index 5	☐
Index 6	☐
Index 7	☐

20 70 60 40 80 50 10 30

[10, 40, 30, 60, 50, 20, 80, 70]

31

implementing the *delete* operation of a *binary min-heap* abstract data type, assuming the element to be deleted is never in the last level. Note that your sequence must absolutely match the step numbers to the left-most column of the table otherwise 3 marks will be deducted for each incorrect match.

(Total 15 marks, i.e. each correct number sequence worth 3 marks.)

	Correct Number Sequence	**Pick Numbers From Here**	
Step 1		1	Repeat steps 3 to 4 until the node reaches its correct position.
Step 2		2	Find the index for the element to be deleted.
Step 3		3	If the replaced element is smaller than any of its child node, swap the element with its greatest child.
Step 4		4	Take out the last element from the last level of the heap and replace the index with this element.
Step 5		5	Replaced the root element in the heap with the found indexed element.
		6	Output updated binary heap.
		7	If the replaced element is greater than any of its child node, swap the element with its smallest child.
		8	Add the indexed element to the bottom leaf of the heap.

2，4，7，1，6