2023 年 12 月认证 Python 六级真题解析

CCF 编程能力等级认证，英文名 Grade Examination of Software Programming（以下简称 GESP），由中国计算机学会发起并主办，是为青少年计算机和编程学习者提供学业能力验证的平台。GESP 覆盖中小学全学段，符合条件的青少年均可参加认证。GESP 旨在提升青少年计算机和编程教育水平，推广和普及青少年计算机和编程教育。

GESP 考察语言为图形化（Scratch）编程、Python 编程及 C++编程，主要考察学生掌握相关编程知识和操作能力，熟悉编程各项基础知识和理论框架，通过设定不同等级的考试目标，让学生具备编程从简单的程序到复杂程序设计的编程能力，为后期专业化编程学习打下良好基础。

本次为大家带来的是 2023 年 12 月份 Python 六级认证真题解析。

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	C	B	D	D	C	D	D	B	B	D	B	D	D	B

1、通讯卫星在通信网络系统中主要起到（） 的作用 。

A. 信息过滤

B. 信号中继

C. 避免攻击

D. 数据加密

【答案】B

【解析】本题主要是考察计算机网络相关的知识点。通讯卫星在通信网络系统中主要起到信号中继的作用，即使用中继设备来增强或传递信号。B 选项中信息过滤通常由其他网络设备来处理，如防火墙或路由器。C 选项中，网络安全通常涉

及其他专门的设备和协议。通讯卫星可以传递经过加密的数据，但它本身并不负责数据加密。D 选项中数据加密通常是由通信的两端设备负责的，而卫星主要负责传递加密后的数据。故本题选择 B 选项。

2、小杨想编写一个判断任意输入的整数 N 是否为素数的程序 ，下面哪个方法不合适？ （ ）

A. 埃氏筛法

B. 线性筛法

C. 二分答案

D. 枚举法

【答案】C

【解析】本题主要考查初等数论中对素数的判定。A 选项中埃氏筛法（埃拉托斯特尼筛法）是用于生成素数的一种经典算法，该算法的基本思想是从小到大遍历自然数，将素数的倍数标记为非素数。通过这种方式，最终留下的未被标记的数就是素数。B 选项中线性筛法，也是一种用于生成素数的高效算法。D 选项中枚举法是一种直观而朴素的方法，用于判断一个整数是否为素数，它的基本思想是逐一检查该整数是否有除了 1 和它本身以外的其他因数，如果有，那么该整数就不是素数；否则，它就是素数。C 选项中二分答案通常用于在有序数据中搜索某个目标值，但在判断素数时，并不涉及有序数据的搜索。故本题选择 C 选项。

3、内排序有不同的类别 ，下面哪种排序算法和冒泡排序是同一类？ （ ）

A. 希尔排序

B. 快速排序

C. 堆排序

D. 插入排序

【答案】B

【解析】本题主要考查几种排序算法。冒泡排序和快速排序都是内排序中比较排序类算法，并且属于交换排序的类型。A 选项希尔排序和 D 选项插入排序是属

于比较排序类算法中的插入排序类型。而 C 选项堆排序是属于比较排序类算法中的选择排序类型。故本题选择 B 选项。

4、关于 Python 类和对象的说法 ， 错误的是(　　)。

A. 在 Python 中 ， 一切皆对象， 即便是字面量如整数 5 等也是对象

B. 在 Python 中 ， 可以自定义新的类 ， 并实例化为新的对象

C. 在 Python 中， 内置函数和自定义函数 ， 都是类或者对象

D. 在 Python 中 ， 不可以在自定义函数中嵌套定义新的函数

【答案】D

【解析】本题主要考查学生对面向对象中类和对象的理解。A 选项，在 Python 中，一切皆对象，这是 Python 的一个基本特性，它使得所有的数据类型、函数、方法甚至于类都可以被视为对象。B 选项，在 Python 中，可以自定义新的类，并实例化为新的对象。这是面向对象编程的特点，Python 支持面向对象(OOP)的编程。C 选项，在 Python 中，函数也是对象，包括内置函数和自定义函数。这是因为在 Python 中，一切皆为对象，函数是一种可调用的对象。D 选项中，在 Python 中，是可以在自定义函数中嵌套定义新的函数的。Python 允许在函数内定义函数，这被称为嵌套函数。故本题选择 D 选项。

5、有关下面Python 代码的说法 ， 正确的是(　　)。

```
1  class Point:
2      def __init__(self,X,Y):
3          self.x = X
4          self.y = Y
5  class Rect:
6      def __init__(self,lefttop_Point,rightbottom_Point):
7          self.left_top = lefttop_Point
8          self.right_bottom = rightbottom_Point
9      def __contains__(self,xy):
10         if self.left_top.x <= xy.x <= self.right_bottom.x \
11                 and self.right_bottom.y <= xy.y <= self.left_top.y:
12             return True
13
14         return False
15
16 rectA = Rect(Point(10,10),Point(20,5))
17 print(Point(15,8) in rectA)
```

A. 第 17行代码执行后将报错， 因为 Rect 类没有定义 in 运算符

B. 第 16行代码将 Point 对象作为参数 ， 将导致错误

C. in 是成员运算符 ，不适用于 Rect 类

D. 由于 Rect 类定义了 __contain__魔术方法， 因此第 17行代码能正确执行

【答案】D

【解析】本题主要考察面向对象中魔法方法的使用。__contains__是 Python 中的一个魔术方法（特殊方法），用于支持 in 运算符。当对象使用 in 运算符时，解释器会尝试调用该对象的__contains__方法，以确定指定的元素是否包含在对象中。也就是说，如果一个类实现了__contains__方法，那么该类的实例可以使用 in 运算符。__contains__方法应该返回一个布尔值，表示对象是否包含给定的值。另外，面向对象编程支持一个类实例化出的对象作为参数传递给当前类的构造函数或其他方法，这样就在两个类之间建立了关联。故本题选择 D 选项。

6、有关下面Python 代码的说法 ，正确的是( )。

```
1 class newClass(object):
2     objCounter = 0
3     def __init__(self):
4         newClass.objCounter += 1
5
6 classA = newClass()
7 classB = newClass()
8 print(newClass.objCounter)
9 print(classA.objCounter)
```

A. 第 8行代码错误 ，第 9行正确

B. 第 9行代码错误 ，第 8行代码正确

C. 第 8 、9 两行代码都正确

D. 第 4行代码可修改为 objCounter += 1

【答案】C

【解析】本题主要考查学生对面向对象中类属性的掌握。代码中先定义了一个类 newClass，定义了一个类属性 objCounter，以及构造函数，接下来实例化出两个对象，第 8 行中，打印了 newClass.objCounter 的值，第 9 行打印了实例对象 a 的 objCounter 属性的值，两行代码都正确。第 4 行代码，通过 newClass.objCounter+=1 改变类的属性值，如果修改成 objCounter+=1 会出错，故本题选择 C 选项。

7、有关下面Python 代码的说法 ，错误的是(　　)。

```
class biTreeNode:
    def __init__(self, val=None, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class biTree(object):
    def __init__(self, root=None):
        self.root = root
```

A. 上列 Python 代码适用于构造各种二叉树

B. 代码 Root = biTree(biTreeNode(5)) 构造二叉树的根节点

C. 代码 Root = biTree( ) 可以构造空二叉树，此时 Root 对象的 Root 属性值为 None

D. 代码 Root = biTree(biTreeNode( ))可以构造空二叉树 ，此时 Root 对象的 root 属性为 Node

【答案】D

【解析】首先我们先分析给出的代码：biTreeNode 类定义了一个二叉树的节点，它有三个属性：val、left 和 right。biTree 类定义了一个二叉树，它有一个属性：root。

biTreeNode 类定义了二叉树的节点，而 biTree 类定义了二叉树的结构，可以用于构造各种二叉树，A 选项正确；B 选项中首先使用 biTreeNode(5)创建一个值为 5 的节点，然后使用 biTree()构造一个二叉树，其根节点是刚刚创建的值为 5 的节点，所以也正确；C 选项中，代码 Root = biTree()创建一个新的二叉树时，没有给定参数，所以 root 值为默认值 None，所以 C 选项也正确；D 选项中，使用 biTree(biTreeNode())创建二叉树时，它仍然是一个空的二叉树，但此时 Root 对象的 root 属性值为一个空 biTreeNode 对象，而不是“Node”。故本题选择 D 选项。

8、基于上题类的定义 ，有关下面Python 代码的说法错误的是（　　）。

```python
def Search(root, val):
    if root is None:
        return None

    if root.val == val:
        return root
    else:
        rtn = search(root.left, val)

    if rtn != None:
        return rtn

    return search(root.right, val)
```

A. 代码中 Search（ ）函数如果查找到查找值的节点 ，则返回该节点的对象

B. 代码中 Search（ ）函数先搜索左子树 ，如果搜索不到指定值 ，则搜索右子树

C. 代码中 Search（ ）函数采用递归方式实现二叉树节点的搜索

D. 代码中 Search（ ）函数采用动态规划方法实现二叉树节点的搜索

【答案】D

【解析】代码中的 Search 函数接收两个参数：root 和 val。如果 root 是 None（也就是说，当前节点为空），则函数返回 None。如果 root.val 等于要搜索的值 val，则返回当前节点。否则，首先在左子树中搜索值，如果找到，返回该节点。如果在左子树中没有找到，则在右子树中搜索。并且在这个例子中，只是通过递归在二叉树中搜索一个值，并没有使用动态规划的方法。故本题 D 选项错误。

9、有关下面Python 代码的说法正确的是（ ）。

```python
class Node:
    def __init__(self, Val, Prv=None, Nxt = None)
        self.Value = Val
        self.Prev = Prv
        self.Next = Nxt

firstNode = Node(10)
firstNode.Next = Node(100,firstNode)
firstNode.Next.Next = Node(111,firstNode.Next)
```

A. 上述代码构成单向链表

B. 上述代码构成双向链表

C. 上述代码构成循环链表

D. 上述代码构成指针链表

【答案】B

【解析】首先分析下这段代码，定义了一个名为 Node 的类，它有三个属性：Value、Prev 和 Next。Value 用于存储节点的值，Prev 和 Next 分别代表该节点的前一个和后一个节点。接下来，创建了一个名为 firstNode 的节点，并为其赋值为 100；接下来设置了 firstNode 的 Next 属性为另一个新创建的节点，其值为 100，并将 firstNode 设置为这个新节点的 Prev 属性；最后设置了新创建的节点的 Next 属性为另一个新创建的节点，其值为 111，并将新创建的节点（值为 100）设置为这个新节点的 Prev 属性。故本题构成了一个双向链表，选择 B 选项。

10、对 hello world 使用霍夫曼编码（Huffman Coding），最少 bit（比特）为（　　）。

A. 4

B. 32

C. 64

D. 88

【答案】B

【解析】霍夫曼编码，其编码方式是根据字符出现的概率来确定的。具体来说，出现概率越高的字符，其对应的编码长度越短；出现概率越低的字符，其对应的编码长度越长。霍夫曼编码的流程主要包括以下几个步骤：

1、统计字符集中每个字符出现的频率

2、根据字符出现的频率构建霍夫曼树

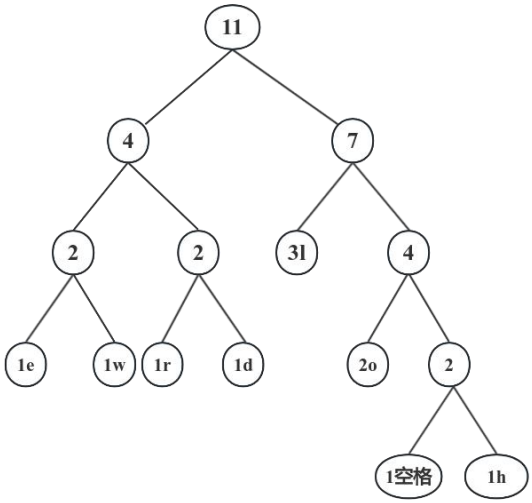
3、根据霍夫曼树为每个字符分配二进制编码

我们根据上述的这三个步骤来构建下本题的霍夫曼树：

1、统计字符集中每个字符出现的频率：

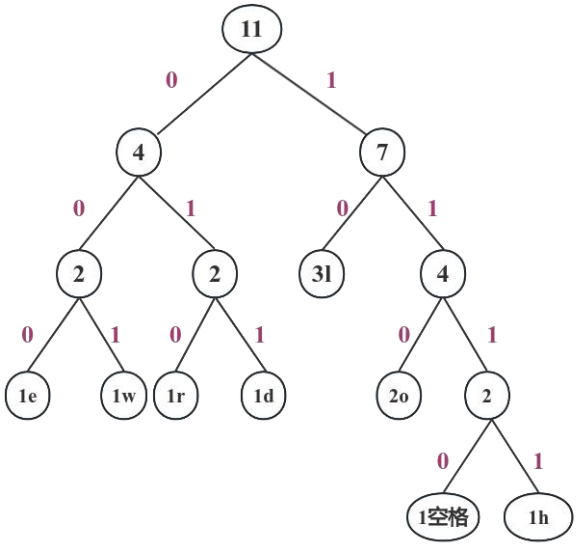
hello world 中各字符出现的次数：h（1）、e（1）、l（3）、o（2）、w（1）、r（1）、d（1）、空格（1）。

2、根据字符出现的频率构建霍夫曼树：



3、根据霍夫曼树为每个字符分配二进制编码：

针对霍夫曼树，从根节点开始，我们将左叶子节点编为 0，右叶子节点编为 1，如下：



每个字符的霍夫曼编码即为从根节点到此字符的路径，

e 编码 000，占 3 位，出现次数 1，总占位 3*1=3；

w 编码 001，占 3 位，出现次数 1，总占位 3*1=3；

r 编码 010，占 3 位，出现次数 1，总占位 3*1=3；

d 编码 011，占 3 位，出现次数 1，总占位 3*1=3；

l 编码 10，占 2 位，出现次数 3，总占位 2*3=6；

o 编码 110，占 3 位，出现次数 2，总占位 3*2=6；

空格编码 1110，占 4 位，出现次数 1，总占位 4*1=4；

h 编码 1111，占 4 位，出现次数 1，总占位 4*1=4；

总占用：3*4+6*2+4*2=32，故本题选择 B 选项。

11、下面的 fiboA（ ）和 fiboB（ ）两个函数分别实现斐波那契数列，该数列第 1、第 2 项值为 1，其余各项分别为前两项之和。下面有关说法错误的是（ ）。

```
def fiboA(N):
    if N == 0:
        return 1
    if N == 1:
        return 1
    return fiboA(N - 1) + fiboA(N - 2)

def fiboB(N):
    dp = [-1] * (N + 1)
    dp[0] = 1
    dp[1] = 1
    for i in range(2, N + 1):
        dp[i] = dp[i - 1] + dp[i - 2]
    return dp[N]
```

A. fiboA（ ）采用递归方式实现斐波那契数列

B. fiboB（ ）采用动态规划算法实现斐波那契数列

C. 当 N 值较大时，fiboA（ ）存在大量重复计算

D. 由于 fiboA（ ）代码较短 ，其执行效率较高

【答案】D

【解析】fiboA()通过递归的方式实现的斐波那契数列，fiboB()通过动态规划算法实现的斐波那契数列，从时间复杂度和空间复杂度的角度来看，动态规划方法比递归方法更高效。故本题 D 选项错误

12、有关下面Python 代码不正确的说法是（ ）。

```
def getDepth(root):
    if root is None:
        return 0
    left_depth = getDepth(root.left)
    right_depth = getDepth(root.right)
    if left_depth < right_depth:
        return right_depth + 1
    else:
        return left_depth + 1
```

A. 该代码可用于求解二叉树的深度

B. 代码中函数 getDepth( ) 的参数 root 表示根节点 ，非根节点不可以作为参数

C. 代码中函数 getDepth( ) 采用了递归方法

D. 代码中函数 getDepth( )可用于求解各种形式的二叉树深度 ，要求该二叉树节点至少有 left 和 right 属性

【答案】B

【解析】首先来分析下这段代码：这段代码定义了一个函数 getDepth，它采用递归的方式计算二叉树的深度。它首先会检查根节点是否为空，如果为空则返回 0。然后它递归的计算左子树和右子树的深度，并返回两者中的较大值加 1。B 选项，除了根节点之外，其它的非根节点也可以作为参数传递给函数。D 选项，给定的代码明确地使用了 root.left 和 root.right 来访问节点的左子节点和右子节点。这意味着它适用于有这两个属性的二叉树节点，否则会报错。故本题选择 B 选项。

13、下面有关树的存储 ，错误的是（ ）.

A. 完全二叉树可以用list 存储

B. 一般二叉树都可以用list 存储 ，空子树位置可以用None 表示

C. 满二叉树可以用list 存储

D. 树数据结构 ，都可以用list 存储

【答案】D

【解析】A 选项中完全二叉树可以用 list 存储，完全二叉树是一种特殊的二叉树，可以按照层次顺序依次存储在一个线性的列表中。B 选项中一般二叉树可以使用列表存储，但需要采用适当的方式来表示树的结构，并对列表中的元素进行合理的编号，空子树位置可以用 None 表示。C 选项中满二叉树是一种特殊的二叉树，所有非叶子节点都有两个子节点，满二叉树的节点数量是确定的，可以按照层次顺序存储在一个线性的列表中，因此也可以用列表存储。D 选项中一般的树结构并不适合直接使用列表来存储，树结构可能具有不定的分支数，而列表是线性的结构，无法直接表示树状关系，对于一般的树，更常见的做法是使用节点和指针或引用来表示。故本题选择 D 选项。

14、下面有关 Python 中 in 运算符的时间复杂度的说法 ，错误的是（ ）。

A. 当 in 运算符作用于 dict 时 ，其时间复杂度为 O(1)

B. 当 in 运算符作用于 set 时 ，其时间复杂度为 O(1)

C. 当 in 运算符作用于 list 时 ，其时间复杂度为 O(N)

D. 当 in 运算符作用于 str 时 ，其时间复杂度为 O(N)

【答案】D

【解析】字典 dict 和集合 set 底层都是由哈希表实现的，所以其时间复杂度为 O(1)；对于列表来说，是一个线性结构，需要一个一个遍历查看，所以时间复杂度为 O(N)；而对于字符串 str，采用一种朴素匹配算法，效率较低，时间复杂度为 O(m*n)，故本题 D 选项错误。

15、下面有关 bool( ) 函数的说法 ，正确的是（ ）。

A. 如果自定义类中没有定义魔术方法 __bool__( )，将不能对该类的对象使用 bool( )函数

B. 如果自定义类中没有定义魔术方法 __bool__( )，将查找有无魔术方法 __len__( )函数 ，如果有__len__( ) 则按 __len__( ) 的值进行处理 ，如果该值为 0 则返回 False ， 否则 True，如果没有__len__( ),则返回值为 True

C. bool( )函数如果没有参数 ，返回值为 True

D. 表达式 bool(int)的值为 False

【答案】B

【解析】A 选项，这个说法是不正确的，如果一个类没有定义__bool__()方法，Python 会尝试查找__len__()方法。如果找到了__len__()方法，并且它的返回值是非零，那么 bool()函数会返回 True，否则返回 False。 如果一个类的__len__()或__bool__()均未定义，则其所有实例都将被视为具有真值。所以 B 选项正确。C 选项，bool() 函数如果没有参数，返回值为 False。D 选项中 bool(int)的结果为 True，故本题选择 B 选项。

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	√	×	√	√	×	×	√	√

1、小杨想写一个程序来算出正整数 N 有多少个因数，经过思考他写出了一个重复没有超过 N/2 次的循环就能够算出来了。 （ ）

【答案】正确

【解析】本题首先要理解什么是因数，因数是能够整除给定数字的整数。例如，对于数字 12，因数有 1、2、3、4、6 和 12。要计算一个数字 N 的因数个数，我们可以使用一个循环，从 1 遍历到 N/2(因为大于 N/2 的数除了它本身不可能是 N 的因数)，在循环中，我们检查每个数字 i 是否是 N 的因数，如果 N 能够被 i 整除，那么 i 就是 N 的一个因数。例如，如果 N 是 100，那么我们只需要检查 1 到 50 的数字是否能够整除 100。因此，对于给定的正整数 N，我们可以通过一个循环，循环次数不超过 N/2 次，来计算出 N 的因数个数。故本题中正确。

2、同样的整数序列分别保存在单链表和双向链中，这两种链表上的简单冒泡排序的复杂度相同。 （ ）

【答案】正确

【解析】冒泡排序是一种简单的排序算法，其基本思想是反复比较相邻的两个元素，如果他们的顺序错误就把他们交换过来，直到没有再需要交换的数为止。这个过程需要对每个元素进行比较和交换。单链表和双向链表在存储结构上有所不同，单链表每个节点只有一个指向下一个节点的链接，而双向链表除了有一个指向下一个节点的链接外，还有一个指向前一个节点的链接。但是，对于冒泡排序来说，这两种链表结构并没有实质性的影响。因为冒泡排序需要遍历整个链表，访问每个节点一次，比较相邻的节点并进行交换。这个过程在单链表和双向链表上是一样的。故本题正确。

3、在面向对象中，方法（Event）在 Python 的 class 中表现为 class 内定义的函数。( )

【答案】正确

【解析】在面向对象编程中，方法（Event）在 Python 的类中是通过在类内部定义函数来表现的。类是创建对象的模板，而方法则是与对象关联的函数。故本题正确。

4、在下面的 Python 代码被执行将报错，因为 newClass 没有__init__（）魔术方法。 （ ）

```
class newClass:
    pass

myNewClass = newClass()
```

【答案】错误

【解析】我们定义类时，可以编写构造函数__init__（）对类的实例进行初始化操作，不过构造函数__init__（）不是必须要有的。在这段代码中定义了一个名为 newClass 的空类，然后创建了这个类的一个实例 myNewClass，newClass 类是空的，它不需要任何初始化操作，所以没有定义__init__()方法也不会导致错误。故本题错误。

5、如果某个 Python 对象（ object）支持下标运算符（方括号运算符），则该对象在所对应 class 中定义了名为 __getitem__的魔术方法 。( )

【答案】正确

【解析】在 Python 中，我们可以使用下标运算符（由方括号表示，例如 obj[index]）来访问序列类型的对象（如列表、元组和字符串）中的元素。此外，这个运算符也可以用于字典来获取键对应的值。当我们使用下标运算符来访问一个对象时，会自动调用该对象的__getitem__()方法。所以，为了使一个对象支持下标运算符，我们需要在这个对象的类中定义__getitem__()方法。故本题正确。

6、深度优先搜索（DFS ， Depth First Search 的简写）属于图算法 ， 其过程是对每一个可能的分支路径深入到不 能再深入为止 ，而且每个节点只能访问一次 。( )

【答案】正确

【解析】深度优先搜索算法，也称为 DFS 算法，是一种遍历图或树的搜索算法，它先沿着一条路径一直走到底，然后回溯到上一个节点，继续沿着另一条路径走到底，直到所有节点都被遍历。深度优先算法有“后进先出”、“尽可能深的搜索”、“每个节点只访问一次”这几个特性。故本题正确。

7、哈夫曼编码（Huffman Coding）具有唯一性，因此有确定的压缩率。(　　)

【答案】错误

【解析】哈夫曼编码，其编码方式是根据字符出现的概率来确定的。具体来说，出现概率越高的字符，其对应的编码长度越短，出现概率越低的字符，其对应的编码长度越长。并且哈夫曼编码并不是唯一的，因为对于同一个字符集和对应的概率分布，可以构造出多个不同的哈夫曼树，从而得到不同的哈夫曼编码。所以，哈夫曼编码并不具有唯一性。故本题错误。

8、Python 虽然不支持指针和引用语法，但变量的本质是数据的引用（reference），因此可以实现各种 C/C++数据结构。在下面Python 代码中，由于删除了变量 a，因此 a 所对应的数据也随之删除，故第 4行代码被执行时，将报错。　(　)

```
1 a, b = [1,2,3], [3,4,5]
2 c = [a, b]
3 del a
4 print(c)
```

【答案】错误

【解析】第一行代码给变量 a 和 b 进行赋值，第二行给变量 c 赋值，需要清楚的是数据保存在内存中，而变量只是一个链接，指向内存地址，变量的赋值的本质是，新的变量也指向内存中的相同地址，只有所有链接都被删除了，那块内存才会被回收，那块内存区域中保存的数据就不复存在了。del 语句删除的是变量，而不是数据。所以第三行 del a，使用 del 语句，删除的只是变量到对象的引用和变量名称本身，del 作用在变量上，而不是数据对象上。所以第四行打印 c 的结果应该是不被第三行的删除影响的，不会报错，结果为[[1, 2, 3], [3, 4, 5]]。故本题错误。

9、二叉搜索树查找的平均时间复杂度为 O(logN) 。　(　)

【答案】正确

【解析】二叉搜索树查找过程如下：

1.从根节点开始。

2.如果当前节点的值等于目标值，查找成功。

3.如果目标值小于当前节点的值，则在左子树中继续查找。

4.如果目标值大于当前节点的值，则在右子树中继续查找。

重复步骤 2-4，直到查找成功或搜索路径为空（即目标值不存在于树中）。

在最好的情况下（即树是完全平衡的），查找的深度为 O(log N)，其中 N 是树中节点的数量。在最坏的情况下（即树退化为链表），查找的深度为 O(N)。在平均情况下，二叉搜索树保证了树的平衡性，查找的平均时间复杂度仍然是 O(log N)。

10、二叉搜索树可以是空树（没有任何节点）或者单节点树（只有一个节点），或者多节点 。如果是多节点 ， 则左节点的值小于父节点的值 ，右节点的值大于父节点的值， 由此推理 ，右节点树的值都大于根节点的值 ，左节点 树的值都小于根节点的值 。 （ ）

【答案】正确

【解析】这道题目主要考察二叉搜索树的性质。首先，二叉搜索树可以是空树，即没有任何节点。其次，二叉搜索树也可以是单节点树，这种情况下只有根节点，没有左子树或右子树。如果二叉搜索树是多节点树，那么它的左子树上所有节点的值都必须小于根节点的值，而右子树上所有节点的值都必须大于根节点的值。这是二叉搜索树最核心的性质。故本题正确。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1、闯关游戏

问题描述

你来到了一个闯关游戏。

这个游戏总共有 N 关 ，每关都有 M 个通道，你需要选择一个通道并通往后续关卡。其中 ， 第 i 个通道可以让你前进 a_i 关，也就是说 ，如果你现在在第 x 关 ，那么选择第 i 个通道后，你将直接来到第 $x+a_i$ 关（特别地，如果 $x + a_i \geq N$，那么你就通关了）。此外，当你顺利离开第 S 关时，你还将获得 b_s 分。

游戏开始时，你在第 0 关。请问，你通关时最多能获得多少总分？

输入描述

第一行两个整数 N, M ，分别表示关卡数量和每关的通道数量。

接下来一行 M 个用单个空格隔开的整数 $a_0, a_1, \cdots, a_{M-1}$ 。保证 $1 \le a_i \le N$ 。

接下来一行 N 个用单个空格隔开的整数 $b_0, b_1, \cdots, b_{N-1}$ 。保证 $|b_i| \le 10^5$ 。

输出描述

一行一个整数，表示你通关时最多能够获得的分数。

特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

样例输入 1

```
6 2
2 3
1 0 30 100 30 30
```

样例输出 1

```
131
```

样例解释 1

你可以在第 0 关选择第 1 个通道，获得 1 分并来到第 3 关;随后再选择第 0 个通道，获得 100 分并来到第 5 关;最后任选一个通道，都可以获得 30 分并通关。如此，总得分为 1+ 100 + 30 = 131。

样例输入 2

```
6 2
2 3
1 0 30 100 30 -1
```

样例输出 2

```
101
```

样例解释 2

请注意，一些关卡的得分可能是负数。

数据规模

对于 20%的测试点 ，保证 $M = 1$ 。

对于 40%的测试点 ，保证 $N \le 20$ ；保证 $M \le 2$。

对于所有测试点 ，保证 $N \le 10^4$ ；保证 $M \le 100$。

【题目大意】

输入关卡数、每关的通道数量，以及每个通道可以前进的关数和每关通过可以得到的分数，找出从第 0 关开始，通关之后能获得的最高分数。

【解题思路】

本题主要考察学生对动态规划这一利用之前计算的结果来求解更大的问题的解决问题方法的使用。

1.完成输入，关卡数量 N，通道数量 M，每个通道前进的关卡数a_i，以及每个关卡获得的分数b_s。

2.使用列表 f 来记录每个关卡通关时的最大得分。从第一个关卡开始，逐次计算每个关卡通关时的最大得分。对于每个关卡，遍历所有可选的通道，找到最大值更新 f[i]（如果当前分数已经计算过（即 f[i]不是 None），则取当前分数和前一关分数的较大值；否则，将前一关分数设置为当前分数）。

3.在计算的过程中，如果达到通关条件时更新最大总分数 ans。

【参考程序】

```python
n, m = input().split()
n, m = int(n), int(m)

a = list(map(int, input().split()))
b = list(map(int, input().split()))

max_b = max(a)

f = [b[0]]

ans = - 10 ** 18

for i in range(1, n):
    f.append(None)
    for step in a:
        if i - step >= 0 and f[i - step] is not None:
            f[i] = max(f[i], f[i - step]) if f[i] is not None else f[i - step]
    if f[i] is not None:
        f[i] += b[i]
    if f[i] is not None and i + max_b >= n:
        ans = max(ans, f[i])
print(ans)
```

2、工作沟通

问题描述

某公司有 N 名员工，编号从 0至N - 1 。其中，除了 0 号员工是老板，其余每名员工都有一个直接领导。我们假设编号为 i 的员工的直接领导是 f_i。

该公司有严格的管理制度，每位员工只能受到本人或本人直接领导或间接领导的管理。具体来说，规定员工x 可以管理员工y，当且仅当 x=y，或 $x = f_y$ ，或 x 可以管理 f_y。特别地，0 号员工老板只能自我管理，无法由其他任何员工管理。

现在，有一些同事要开展合作，他们希望找到一位同事来主持这场合作，这位同事必须能够管理参与合作的所有同事。如果有多名满足这一条件的员工，他们希望找到编号最大的员工。你能帮帮他们吗？

输入描述

第一行一个整数 N，表示员工的数量。

第二行N-1 个用空格隔开的正整数，依次为 f_1 , f_2 ，...， f_{N-1}。

第三行一个整数 Q ，表示共有 Q 场合作需要安排。

接下来 Q 行，每行描述一场合作：开头是一个整数 m（$2 \le m \le N$），表示参与本次合作的员工数量；接着是 m 个整数，依次表示参与本次合作的员工编号（保证编号合法且不重复）。

保证公司结构合法，即不存在任意一名员工，其本人是自己的直接或间接领导。

输出描述

输出 Q行，每行一个整数，依次为每场合作的主持人选。

特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

样例输入 1

```
5
0 0 2 2
3
2 3 4
3 2 3 4
2 1 4
```

样例输出 1

```
2
2
0
```

样例解释

对于第一场合作，员工 3，4 有共同领导 2，可以主持合作。

对于第二场合作，员工 2 本人即可以管理所有参与者。

对于第三场合作，只有 0 号老板才能管理所有员工。

样例输入 2

```
7
0 1 0 2 1 2
5
2 4 6
2 4 5
3 4 5 6
4 2 4 5 6
2 3 4
```

样例输出 2

```
2
1
1
1
0
```

数据规模

对于 50%的测试点，保证 $N \le 50$。

对于所有测试点，保证 $3 \le N \le 300$ ； $Q \le 100$。

【题目大意】

某公司员工之间有一定的管理制度，只能受到本人、直接领导或者间接领导的管理。现在有一些合作场合，需要找到一位员工来主持，这位员工必须能够管理参与合作的所有同事，要求找到编号最大的员工。

【解题思路】

本题主要考察通过使用深度优先搜索（DFS）来判断每个员工是否能够管理参与合作的所有同事这一内容。

1.完成输入员工数量 n 和员工 i 的直接领导 father，然后创建一个列表 child，通过遍历 father 列表，填充 child 员工列表。

2.接下来输入合作的数量 q，然后进入循环处理每个合作场合。每次先完成输入参与本次合作的员工数量，以及参与本次合作的员工编号。

3.然后使用深度优先搜索（DFS）来判断每个员工是否能够管理参与合作的所有同事。在每次 DFS 后，通过检查是否所有参与员工都被访问到，来判断当前员工是否能够管理合作的同事。如果是，则更新 ans 为当前员工的编号。

【参考程序】

```python
n = int(input())
father = [-1] + list(map(int, input().split(' ')))
child = [[] for i in range(n)]
for i in range(1, n):
    child[father[i]].append(i)

q = int(input())
for _ in range(q):
    x = list(map(int, input().split(' ')))[1:]
    ans = -1
    for r in range(n):
        vis = [False for i in range(n)]
        def dfs(node):
            vis[node] = True
            for c in child[node]:
                dfs(c)
        dfs(r)
        flag = True
        for y in x:
            if not vis[y]:
                flag = False
                break
        if flag:
            ans = r
    print(ans)
```

【联系我们】

1. GESP 微信：关注“CCF GESP”公众号，将问题以文字方式留言即可得到回复。

2. GESP 邮箱：gesp@ccf.org.cn

注：请在邮件中详细描述咨询的问题并留下考生的联系方式及姓名、身份证号，以便及时有效处理。

3. GESP 电话：0512-67656856

咨询时间：周一至周五(法定节假日除外）: 上午 8:30-12:00；下午 13:00-17:30

GESP 第五期认证报名已启动，扫描下方二维码，关注 GESP 公众号即可报名