2023 年 12 月认证 Python 五级真题解析

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 五级认证真题解析。

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	C	A	D	A	B	A	C	B	D	B	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】卫星通信的主要目的是通过卫星发射的信号和地面接收的信号实现用户间的通信，属于中转信号功能，过滤、攻击和加密不是卫星的主要功能，故选 B。

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】二分答案算法的核心思路是对需要求的答案进行二分，每次将答案范围减半直到找到目标答案为止；在求素数问题上，不存在二分的明显规律，因此使用二分法判断是否是素数不是一种合适的算法，本题选 C。

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、下面Python 代码用于求斐波那契数列 ，该数列第 1 、2 项为 1， 以后各项均是前两项之和 。下面有关说法错误的是(　　)。

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

def fiboB(N):
    if N == 1 or N == 2:
        return 1

    last2, last1 = 1, 1
    for i in range(2,N):
        nowVal = last1 + last2
        last2, last1 = last1, nowVal
    return nowVal
```

A. fiboA( ) 用递归方式， fiboB() 循环方式

B. fiboA( ) 更加符合斐波那契数列的数学定义 ， 直观易于理解 ， 而 fiboB() 需要将数学定义转换为计算机程 序实现

C. fiboA( ) 不仅仅更加符合数学定义 ， 直观易于理解 ， 且因代码量较少执行效率更高

D. fiboB( ) 虽然代码量有所增加 ， 但其执行效率更高

【答案】C

【解析】fiboA()借助函数自我调用求解数列，fiboB()借助简单循环方式求解数列，选项 A 正确；fiboA()直观体现了斐波那契数列中元素间的关系，选项 B 正确；递归算法代码执行效率与递归范围关系更密切，与代码量关系不大，选项 C 错误；fiboA()中的递归算法通常来说执行效率偏低，fiboB()只需要执行一层循环，效率更高，选项 D 正确；故此题选 C。

5、下面Python 代码以递归方式实现合并排序 ，并假设 merge(left,right) 函数能对有序（同样排序规则）的 left 和 right 排序 。横线处应填上代码是( )。

```
def mergeSort(listData):
    if len(listData) <= 1:
        return listData

    Middle = len(listData) // 2
    Left, Right = ____________________________________

    return merge(Left, Right)
```

A. mergeSort(listData[:Middle]), mergeSort(listData[Middle:])

B. mergeSort(listData[:Middle-1]), mergeSort(listData[Middle+1:])

C. mergeSort(listData[:Middle]), mergeSort(listData[Middle+1:])

D. mergeSort(listData[:Middle-1]), mergeSort(listData[Middle:])

【答案】A

【解析】使用递归方式实现合并排序时，整体思路为对数列进行左右两半的分段，再分别对左右两段进行合并排序；题目中 Middle 作为原列表元素位置的中点，在进行列表切片时，需要根据切片规则注意 Middle 处的值是否丢失或重复；B 选项丢失了 Middle-1 及 Middle，C 选项丢失了 Middle，D 选项丢失了 Middle-1，故此题选 A。

6、阅读下面的 Python 代码 ，执行后其输出是(　)。

```python
stepCount = 0
def countIt(Fx):
    def wrapper(*args,**kwargs):
        rtn = Fx(*args,**kwargs)
        global stepCount
        stepCount += 1
        print(stepCount,end="->")
        return rtn
    return wrapper

@countIt
def fracA(N):
    rtn = 1
    for i in range(1,N+1):
        rtn *= i
    return rtn;

@countIt
def fracB(N):
    if N == 1:
        return 1
    return N*fracB(N-1)

print(fracA(5),end="")
print("<===>",end="")
print(fracB(5))
```

A. 1->120<===>2->120

B. 1->120<===>1->120

C. 1->120<===>1->2->3->4->5->120

D. 1->120<===>2->3->4->5->6->120

【答案】A

【解析】代码中 fracA()和 fracB()两个函数，均是求解 N 的阶乘的函数，并把 N 的阶乘作为返回值返回给函数；不同的是 fracA()使用递推进行求解，fracB()使用递归求解；通过装饰器对两个函数功能的链接，第一个 print 输出的先是 stepCount 的值+5 的阶乘的返回值；第二个 print 输出了中转符号；而第三个 print 输出由递归求解的 5 的阶乘值时，由于函数内部自我逐层调用与逐层返回，在装饰器捆绑的 countIt()函数功能下，接连返回了每次递归的次数，最终输出阶乘结果 120；

由于使用 global 关键字生命 stepCount 为全局变量，所以第三个 print 输出过程中，该值从 2 开始一直到 6。本题正确答案为 D。

7、下面的 Python用于对 lstA 排序，使得偶数在前奇数在后，横线处不应填入(　)。

```
def isEven(N):
    return N % 2 == 0

lstA = list(range(1,100))
lstA.sort(_____________________________)
```

A. key = not isEven

B. key = lambda x: isEven(x), reverse = True

C. key = isEven, reverse = True

D. key = lambda x: not isEven(x)

【答案】A

【解析】sort()方法中的 key 参数，可以接收自定义的函数名作为其参数值，也可以接收获取到的某个值来作为排序的依据。isEven()会将获得到的参数奇数返回 0，偶数返回 1；结合 lambda 函数，以此函数返回的 0 或 1 作为排序依据，那么 lambda x: is Even(x) 会使奇数排列在偶数前(0<1)；故选项 B、C 中设置的 reverse 翻转顺序，可以将偶数排列到奇数前，以此实现题目要求；选项 D 中对函数返回值类型做 not 否定，也能够是排序依据发生颠倒，所以 D 选项写法也能够实现效果；但 A 选项写法中，对函数名直接进行 not 否定，由于函数名作为 key 的参数是并非 bool 类型，会引起程序报错，所以本道题目选择 A。

8、下面的 Python 代码用于排序 sA 字符串中每个字符出现的次数（字频），sA 字符串可能很长，此处仅为示 例 。排序要求是按字频降序，如果字频相同则按字符的 ASCII 升序，横线处应填入代码是（　　）。

```
sA = "Simple is better than complex"
charCount = {} #每个字符对应数量
for c in sA:
    charCount[c] = charCount.get(c, 0) + 1
print(sorted(_____))
```

A. charCount, key = lambda x:(-x[1],x[0])

B. charCount.items(), key = lambda x:(-x[1],ord(x[0]))

C. charCount.items(), key = lambda x:(x[1],-ord(x[0]))

D. 触发异常 ，不能对字典进行排序。

【答案】B

【解析】本道题通过字典存储信息，字典中元素的键表示字符，值表示该字符出现次数；通过遍历字符串，将每个获取到的字符作为键存入或访问，如果 get()的返回值非 None 则把该值增加 1，实现了计数的效果；排序依据需要借助字典中键和值设置条件，通过获取 items()，获取到的第 0 项为字符，第 1 项为数值，所以-x[1]为大数的相反数，排列靠前实现按字频降序；ord(x[0])为字符则设置为第二排序依据，实现字符 ASCII 升序，故本题选 B。

9、有关下面Python 代码正确的是（ ）。

```
isEven = lambda x: x % 2 == 0
def isOdd(N):
    return N % 2 == 1
print(type(isEven) ==  type(isOdd), isEven(10),isOdd(10))
```

A. True True False

B. False True False

C. False False True

D. 触发异常

【答案】A

【解析】isEven 和 isOdd()两个函数，经运算后得到的均为布尔值，故 type(isEven)和 type(isOdd)类型相同，第一个输出结果为 True; isEven(10)当传入参数为 10 时，10%2==0 条件成立，故返回 True 值给函数，第二个输出结果为 True；isOdd(10)

给出 10 作为参数时，10%2==1 条件不成立，返回 False 给函数，第三个输出结果为 False，故此题选择 A 选项。

10、下面的 Python 代码实现对 list 的快速排序 ，有关说法 ，错误的是（ ）。

```
def qSort(lst):
    if len(lst) < 2:
        return lst

    pivot = lst[0]
    less = [i for i in lst[1:] if i <= pivot]
    greater = [i for i in lst[1:] if i > pivot]

    return _____
```

A. qSort(less) + qSort(greater) + [pivot]

B. [pivot] + qSort(less) + qSort(greater)

C. qSort(less) + [pivot] + qSort(greater)

D. qSort(less) + pivot + qSort(greater)

【答案】C

【解析】快速排序的思路为，借助分治算法思想，将待排序列表分成三部分：参照值、比参照值小的、比参照值大的；代码中 pivot 选取列表第 0 项为参照值，然后遍历列表将小于等于参照值的数加入 less 列表，大于参照值的数加入 greater 列表；此时排序未完成，需要对 less 列表、greater 列表再次调用 qSort()函数排序，并将结果作为三部分列表返回给函数，顺序应为“小”+“参照”+“大”，故此题选 C 。

11、下面Python 代码中的 isPrimeA() 和 isPrimeB() 都用于判断参数 N 是否素数 ，有关其时间复杂度的正确说 法是（ ）。

```python
def isPrimeA(N):
    if N < 2:return False

    for i in range(2,N // 2 + 1):
        if N % i == 0:return False
    return True

def isPrimeB(N):
    if N < 2:return False

    for i in range(2, int(N ** 0.5) + 1):
        if N % i == 0:return False
    return True
```

A. isPrimeA( ) 的最坏时间复杂度是 O($\frac{N}{2}$) ， isPrimeB( ) 的最坏时间复杂度是 O(logN) ， isPrimeA()优于 isPrimeB()

B. isPrimeA() 的最坏时间复杂度是 O($\frac{N}{2}$) ， isPrimeB( ) 的最坏时间复杂度是 O($N^{1/2}$) ， isPrimeB()绝大多数情况下优于 isPrimeA()

C. isPrimeA() 的最坏时间复杂度是 O($N^{1/2}$) ， isPrimeB( ) 的最坏时间复杂度是 O(N) ， isPrimeA( ) 优于 isPrimeB( )

D. isPrimeA() 的最坏时间复杂度是 O(logN) ， isPrimeB( ) 的最坏时间复杂度是 O(N) ， isPrimeA( ) 优于 isPrimeB( )

【答案】B

【解析】isPrimeA()函数中，for 循环范围为从 2 到 N//2+1，运行次数最大为 N//2，时间复杂度为 O(N/2)；isPrimeB()函数中，for 循环范围为从 2 到 N 的 0.5 次方+1，运行次数最大为 N½，时间复杂度为 O(N½)；通常情况下，N½值要小于 N/2 值，所以 isPrimeB()运行效率通常更高；故此题选 B。

12、下面Python 代码用于有序 list 的二分查找 ，有关说法错误的是（ ）。

```python
def bSearch(lst,Val):

    def _binarySearch(lst, Low, High, Target):
        if Low > High:
            return -1
        Mid = (Low + High) // 2  #求序列中间位置
        if Target == lst[Mid]:
            return Mid
        elif Target < lst[Mid]:  #如目标值小于中间元素，在左半部分查找
            return _binarySearch(lst, Low, Mid - 1, Target)
        else:  #如目标值大于中间元素，在右半部分查找
            return _binarySearch(lst, Mid + 1, High, Target)

    return _binarySearch(lst,0,len(lst),Val)
```

A. 代码采用二分法实现有序 list 的查找

B. 代码采用分治算法实现有序 list 的查找

C. 代码采用递归方式实现有序 list 的查找

D. 代码采用动态规划算法实现有序 list 的查找

【答案】D

【解析】二分查找算法借助于二分法思想，将查找范围不断二分减半，是一种快速的查找算法；在本题中给出的写法中，通过先将列表分为左、中、右三部分并依次查找，再将结果整合到一块返回，体现了分治算法中先分、再治、最后合的特点；而程序定义函数和函数自我调用，体现了递归算法的实现方式，故此题 A、B、C 选项皆正确；而 D 选项动态规划算法，更重要的是体现整体过程中每个步骤需要根据条件选取的决策，跟本题代码关联不大，故选择 D。

13、在上题的算法中，其时间复杂度是（　　）。

A. O(N)

B. O(logN)

C. O(NlogN)

D. $O(N^2)$

【答案】B

【解析】二分查找算法每次会将查找范围缩小一半，假设有 n 个元素，查找进行了 k 次，查找到最后一个元素时找到目标，则有等式 n/2^k=1，即 2^k=n，k=log2N，所以时间复杂度为 O(logN)。

14、下面的 Python 代码用于实现每个字符后紧跟随字符及其出现次数 ，并对紧跟随字符排序，即出现次数最多排在前，形如： {'中 ': [('文 ', 1), ('国 ', 2), ('华 ', 2)]} ，此处 S 仅是部分字符 ，可能很多 ，横线处应 填入代码是（ ）。

```
S = """中国华夏中华中华人民共和国中国人中文"""

dictAfter = {}
for i,hz in enumerate(S[1:]):
    tmpDict = dictAfter.get(S[i], {})
    tmpDict[hz] = tmpDict.get(hz, 0) + 1
    dictAfter[S[i]] = tmpDict
dictAfter = { _____ for x in dictAfter.items()}
print(dictAfter)
```

A. x[0]:x[1].items().sort(key = lambda x:x[1], reverse = True)

B. x[0]:x[1].items().sort(key = lambda x:x[0], reverse = True)

C. x[0]:sorted(x[1].items(), key = lambda x:-x[0])

D. x[0]:sorted(x[1].items(), key = lambda x:-x[1])

【答案】B

【解析】本题思路及方法类似于选择题第 8 题，将字符及字符出现次数作为键值对存入字典后，再使用 items()方法并结合字典推导式获取字典中的键与值，以-x[1]作为排序依据实现根据字频降序排列，再将 x[0]的字符与排序后的元素值组合加入 dictAfter 字典，实现题目效果，此题选 D。

15、有关下面Python 代码的说法正确的是（ ）。

```
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】根据代码显示，前置节点通过 Next 属性指向后置节点，后置节点又通过 Prv 属性指回前置节点，并且代码中三个节点并未构成循环，从概念上属于双向链表，正确答案为 B。

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】在枚举过程中，如果找到某个数字 x 是 N 的因数，那么 N/x 即为 N 的另一个因数，所以重复次数可以不超过 N/2。

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】冒泡排序的特点为遍历待排序数列，依次进行数据间的两两比较，比较的方向是从前向后的，方向上都是单向比较，在两种链表结构种并没有效率差距，所以本题正确。

3、归并排序的时间复杂度是 。（ ）

【答案】正确

【解析】归并排序每次将序列拆分成等长的两部分，然后对这两部分进行排序，最后再将这两部分合并成一个有序序列。每次将序列拆分成两部分的时间复杂度为 O(n/2)，对两部分进行排序的时间复杂度为 O(nlog(n))，因此总的时间复杂度为 O(n) * O(nlog(n)) = O(n*log(n))。

4、在 Python 中 ，当对 list 类型进行 in 运算查找元素是否存在时 ，其查找通常采用二分法。 （ ）

【答案】错误

【解析】python 中使用关键字 in 判断列表内是否包含某元素时，大多数情况使用的是列表类型的 index()方法对其中元素进行依次比对，并不是使用快速查找的二分法。

5、以下 Python 代码能以递归方式实现斐波那契数列 ，该数列第 1 、2 项为 1，以后各项均是前两项之和 。( )

```
def Fibo(N):
    if N == 1 or N == 2:
        return 1
    else:
        m = fiboA(N - 1 )
        n = fiboB(N - 2)
        return m + n
```

【答案】错误

【解析】代码中 fiboA()和 fiboB()两个函数并没有定义，不是在 Fibo()中调用函数本身，无法通过递归的方式求解 N 的前两项的值，故此题错误。

6、贪心算法可以达到局部最优 ，但可能不是全局最优解 。 （ ）

【答案】正确

【解析】贪心算法的核心是，从局部最优上优先考虑，做出当前选择的最优解；算法关键在于贪心策略的选择，某个状态的选择不会影响以后的状态，贪心算法并不能保证对全局所有问题都得到最优解，本题描述正确。

7、如果自定已 class 已经定义了 __lt__() 魔术方法 ，则自动支持内置函数 sorted() 。 （ ）

【答案】正确

【解析】__lt__() 方法是一个比较方法，用于比较两个对象的大小。当使用 sorted() 函数对包含这个类的对象列表进行排序时，会根据 __lt__() 方法的结果进行排序。

8、插入排序有时比快速排序时间复杂度更低 。 （ ）

【答案】正确

【解析】当待排序数列接近有序时，插入排序进行的操作更少，时间复杂度可能略低于快速排序。

9、下面的 Python 代码能实现十进制正整数 N 转换为八进制并输出。 （ ）

```
N = int(input())
rst = "" #保存转换结果
while N != 0:
    rst += str(N % 8)
    N //= 8
print(rst)
```

【答案】错误

【解析】通过对 8 取余后的余数，应该是倒序排列后才是转换成的八进制数，题目中代码应改为输出 rst[::-1]后才是正确的转换结果。

10、Python 代码 print(sorted(list(range(10)), key = lambda x:x % 5)) 执行后将输出 [0, 5, 1, 6, 2, 7, 3, 8, 4, 9] 。 （ ）

【答案】正确

【解析】list(range(10))的结果是[0,1,2,3,4,5,6,7,8,9]，相当于使用 sorted()函数对该列表进行排序；lambda x: x%5 作用为依次从列表中取出元素 x，并以 x%5 作为依据进行排序；故 0,1,2,3,4 对 5 取余变，5,6,7,8,9 对五取余分别为 0,1,2,3,4，所以分别排在了原列表 0,1,2,3,4 后面，最终列表中元素排列顺序为 [0,5,1,6,2,7,3,8,4,9]，此题正确。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1、小杨的幸运数

问题描述

小杨认为，所有大于等于 a 的完全平方数都是他的超级幸运数。

小杨还认为，所有超级幸运数的倍数都是他的幸运数。自然地，小杨的所有超级幸运数也都是幸运数。

对于一个非幸运数，小杨规定，可以将它一直＋1，直到它变成一个幸运数。我们把这个过程叫做幸运化。例如，如果 a＝4，那么 4 是最小的幸运数，而1 不是，但我们可以连续对 1 做 3 次＋1 操作，使其变为 4，所以我们可以说，1 幸运化后的结果是 4。

现在，小样给出 N 个数，请你首先判断它们是不是幸运数；接着，对于非幸运数，请你将它们幸运化。

输入描述

第一行 2 个正整数 a，N。

接下来 N 行 ，每行一个正整数 x ，表示需要判断（幸运化） 的数。

输出描述

输出 N 行，对于每个给定的 x ，如果它是幸运数，请输出 lucky ，否则请输出将其幸运化后的结果。

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

```
2 4
1
4
5
9
```

样例输出 1

```
4
lucky
8
lucky
```

样例解释 1

1 虽然是完全平方数，但它小于 a， 因此它并不是超级幸运数，也不是幸运数。

将其进行3 次＋1 操作后，最终得到幸运数 4。

4 是幸运数，因此直接输 lucky。

5 不是幸运数，将其进行3 次＋1 操作后，最终得到幸运数 8。

9 是幸运数，因此直接输出 lucky。

样例输入 2

```
16 11
1
2
4
8
16
32
64
128
256
512
1024
```

样例输出 2

```
16
16
16
16
lucky
lucky
lucky
lucky
lucky
lucky
lucky
```

数据规模

对于 30%的测试点，保证 a,x≤ 100， N ≤ 100。

对于 60%的测试点，保证 a, x≤ 10^6。

对于所有测试点，保证 a ≤1,000,001; 保证 N ≤2x 10^5，保证 1 ≤ x ≤1,000,001。

【题目大意】本题围绕“完全平方数”这一概念，求出大于要求数字的所有全部完全平方数作为超级幸运数字，并且根据这些完全平方数，求出范围内所有完全平方数的倍数；再将不属于完全平方数的数字进行自增操作，直到变为最近的超级幸运数为止。

【解题思路】

1.按要求输入数据并设定最大范围；

2.从 a 到最大数范围内进行遍历，将每个完全平方数记录到幸运数列表中；同时当获得完全平方数后，将该数字的所有倍数记录到幸运数列表中；

3.从大到小遍历“下一个幸运数”列表，将每个非幸运数的下一个幸运数设置为其后面的第一个幸运数；

4.最后，根据输入的整数 x，如果 x 是幸运数，则输出 "lucky"，否则输出其下一个非幸运数。

【参考程序】

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

max_lucky = 1001 ** 2 + 10

is_lucky = [False for i in range(max_lucky + 1)]
next_lucky = [-1 for i in range(max_lucky + 1)]

for i in range(1, max_lucky):
    if i >= a and int(i ** 0.5) ** 2 == i:
        is_lucky[i] = True
        for j in range(i + i, max_lucky, i):
            is_lucky[j] = True

for i in range(max_lucky - 1, 0, -1):
    if is_lucky[i]:
        next_lucky[i] = i
    else:
        next_lucky[i] = next_lucky[i + 1]

for i in range(n):
    x = int(input())
    assert 1 <= x <= 10 ** 6 + 1
    if next_lucky[x] == x:
        print("lucky")
    else:
        print(next_lucky[x])
```

2、烹饪问题

问题描述

有 N 种食材，编号从 0至N - 1，其中第 i 种食材的美味度为 a_i 。

不同食材之间的组合可能产生奇妙的化学反应。具体来说，如果两种食材的美味度分别为 x 和 y，那么它们的契合度为 x and y。

其中，and 运算为按位与运算 ，需要先将两个运算数转换为二进制，然后在高位补足0，再逐位进行与运算。例如，12 与 6 的二进制表示分别为 1100 和 0110，将它们逐位进行与运算，得到 0100，转换为十进制得到 4，因此 12 and 6 = 4。

在 C++或 Python 中，可以直接使用&运算符表示与运算。

现在，请你找到契合度最高的两种食材，并输出它们的契合度。

输入描述

第一行一个整数 N，表示食材的种数。

接下来一行N 个用空格隔开的整数，依次为 a_0，…，a_{N-1}，表示各种食材的美味度。

输出描述

输出一行一个整数，表示最高的契合度。

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

```
3
1 2 3
```

样例输出 1

```
2
```

样例解释 1

可以编号为 1 , 2 的食材之间的契合度为 2 and 3 ＝ 2 ，是所有食材两两之间最高的契合度。

样例输入 2

```
5
5 6 2 10 13
```

样例输出 2

```
8
```

样例解释 1

可以编号为 3 ，4 的食材之间的契合度为 10 and 13 = 8 ，是所有食材两两之间最高的契合度。

数据规模

对于 40%的测试点 ，保证 $N \le 1,000$；

对于所有测试点 ，保证 $N \le 10^6$ ，$0 \le a_i \le 2,147,483,647$。。

【题目大意】本题围绕数字间的位运算操作，根据给出的数据，求出 N 个数之中，两个数字间按位与的最大值。

【解题思路】

1.双层循环嵌套可以列举所有数字搭配，并进行位运算后记录最大值；但当数据量超大时，会导致程序超时，需要进行优化；

2.根据 ai≤2,147,483,647(2^{31}-1)，最多 31 位二进制，从最高位开始枚举，时间复杂度 O(31N)，可以解决所有数据且不超时；

3.记录每位下的最大值作为答案，并枚举进行位运算后，找到当前位数下满足条件的数值个数；个数小于 2 时，答案变量清零并进行下一次枚举。

【参考程序】

```python
n = int(input())
a = list(map(int, input().split()))
a = [0] + a


def sort_(l, r, k):
    while l <= r:
        while (l <= r) and (a[l] >> k & 1):
            l += 1
        while (l <= r) and (not (a[r] >> k & 1)):
            r -= 1
        if l <= r:
            a[l], a[r] = a[r], a[l]
            l += 1
            r -= 1
    return r


ans = 0
for i in range(31, -1, -1):
    j = sort_(1, n, i)
    if j >= 2:
        ans |= 1 << i
        n = j

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