CCF 编程能力等级认证
Grade Examination of Software Programming

Python 五级

2024 年 12 月

1 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	A	C	C	B	C	C	A	D	A	A	A	A	D

第 1 题 下面的程序中，x,y都是正整数，完成的算法是（ ）

```
def chenadai(x, y):
    while y:
        x, y = y, x % y
    return x
```

- ☐ **A.** 最小公倍数
- ☐ **B.** 最大公约数
- ☐ **C.** 数字x能够整除y的最小数
- ☐ **D.** 数字x与数字y的所有公约数

第 2 题 下列程序中实现的是（ ）

```
def chenadai(x, y):
    return x * y // (x,y的最大公约数)
```

- ☐ **A.** 实现了求x,y的最小公约数
- ☐ **B.** 实现了求x,y的最大公约数
- ☐ **C.** 实现了求x,y的最小公倍数
- ☐ **D.** 实现了求x,y的平均值

第 3 题 欧几里得算法又称作辗转相除算法，下面程序中是这种算法的是（ ）

- ☐ **A.**

```
def gcd(a,b):
    if b == 0:
        return a
    return gcd(b, a % b)
```

- ☐ **B.**

```
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        a,b = b,a%b
    return b
```

C.

```
def gcd(a,b):
    if b == 0:
        return a
    return gcd(a, a % b)
```

D.

```
def gcd(a,b):
    if b == 0:
        return a
    return gcd(b, b % a)
```

第 4 题 下列程序是二分法的程序，横线处应该填上（ ）。

```
def binary_search(arr, target):
    left = 0
    right = len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        ____________________
    return -1
```

A.

```
if arr[mid] == target:
    return mid
elif arr[mid] > target:
    right = mid - 1
```

B.

```
if arr[mid] == target:
    return mid
elif arr[mid] > target:
    right = mid
else:
    left = mid
```

C.

```
    if arr[mid] == target:
        return mid
    elif arr[mid] > target:
        right = mid - 1
    else:
        left = mid + 1
```

- [ ] **D.**

```
if arr[mid] == target:
    return mid
else:
    left = mid + 1
```

第 5 题 下面折半查找程序的时间复杂度为（ ）

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

- [ ] **A.** $O(n * logn)$
- [ ] **B.** $O(n)$
- [ ] **C.** $O(logn)$
- [ ] **D.** $O(n^2)$

第 6 题 下列程序中，使用了埃氏筛法，横线处应该填写的是（）

```
def aishishai(n):
    if n < 2:
        return []
    prime = [True] * (n + 1)
    prime[0] = prime[1] = False
    ____________________________________
        if prime[p]:
            for i in range(p * p, n + 1, p):
                prime[i] = False
    return [p for p in range(n + 1) if prime[p]]
```

- [ ] **A.** `for p in range(2, n ** 0.5 + 1):`
- [ ] **B.** `for p in range(2, int(n ** 0.5) + 1):`
- [ ] **C.** `for p in range(2, int(n ** 0.5) + 0.5):`

☐ **D.** `for p in range(2, n ** 0.5 + 0.5):`

第 7 题 18到100之间的所有素数的和为多少( )

☐ **A.** 1060

☐ **B.** 1004

☐ **C.** 1002

☐ **D.** 1052

第 8 题 下面程序是对2024进行唯一分解，最后的结果应该是（ ）。

```
def weiyi(n):
    factors = {}
    for i in range(2, n+1):
        while n % i == 0:
            if i in factors:
                factors[i] += 1
            else:
                factors[i] = 1
            n //= i
    return factors
```

☐ **A.** 1,2,3,13,23

☐ **B.** 2,7,11,23

☐ **C.** 2:3,11:1,23:1

☐ **D.** 2,3,13,23

第 9 题 下面关于循环链表的说法正确的是（ ）。

☐ **A.** 循环链表的最后一个结点指向头结点，形成一个闭环

☐ **B.** 必须通过特定结点才可以遍历整个链表

☐ **C.** 不属于链式存储结构

☐ **D.** 在长度为n的顺序表下标为i的位置前插入一个元素（$1 \le i \le n+1$），元素的移动次数为n-i+1

第 10 题 下列归并算法程序中，横线处应该填入的是（ ）

```
def merge_sort(array):
    if len(array) == 1:
        return array
    ________________________
    return merge(left, right)


def merge(left, right):
    left_index, right_index, merge_array = 0, 0, list()
    while left_index < len(left) and right_index < len(right):
        if left[left_index] <= right[right_index]:
            merge_array.append(left[left_index])
            left_index += 1
```

```
        else:
            merge_array.append(right[right_index])
            right_index += 1
    merge_array = merge_array + left[left_index:] + right[right_index:]
    return merge_array
```

- [ ] **A.**

```
left = merge_sort(array[:len(array)-1//2])
right = merge_sort(array[len(array)-1//2:])
```

- [ ] **B.**

```
left = merge_sort(array[len(array)//2-1])
right = merge_sort(array[len(array)//2:])
```

- [ ] **C.**

```
left = merge_sort(array[len(array)//2-1])
right = merge_sort(array[len(array)//2])
```

- [ ] **D.**

```
left = merge_sort(array[:len(array)//2])
right = merge_sort(array[len(array)//2:])
```

第 11 题 关于算法复杂度，下列说法不正确的是（）

`

- [ ] **A.** 简单的for循环的时间复杂度是$O(1)$
- [ ] **B.** 归并排序的时间复杂度是$O(nlog(n))$
- [ ] **C.** 选择排序中的内循环时间复杂度是$(O(n^2))$
- [ ] **D.** 递归的 Fibonacci 数列计算时间复杂度是$O(2^n)$

第 12 题 下列程序中，实现了16进制转到8进制。横线处应该填入的是（ ）

```
def dec_conversion_n(n, base):
    str_list = "0123456789ABCDEF"
    if n < base:
        return str_list[n]
    else:
        ____________________________
```

- [ ] **A.**

```
return dec_conversion_n(n // base, base) + str_list[n % base]
```

- [ ] **B.**

```
return dec_conversion_n(n // base, n) + str_list[n % base]
```

- [ ] **C.**

```
return dec_conversion_n(n // base, base) + str_list[n // base]
```

☐ **D.**

```
return dec_conversion_n(n // base, n) + str_list[n // base]
```

第 13 题 水仙花数是指一个 3 位数，它的每个数位上的数字的 3次幂之和等于它本身。下面代码是计算100到n之间有多少个水仙花数的程序，横线处应该填写的一行或多行代码是（ ）。

```
n = int(input("输入一个正整数N:"))
sum = 0
for i in range(100,n+1):
    ____________________
print(sum)
```

☐ **A.**

```
ge = i%10
shi = i//10%10
bai = i//100
if i == ge*ge*ge+shi*shi*shi+bai*bai*bai:
    sum+=1
```

☐ **B.**

```
ge = i%10
shi = i%10%10
bai = i//100
if i == ge*ge*ge+shi*shi*shi+bai*bai*bai:
    sum+=1
```

☐ **C.**

```
ge = i%10
shi = i//10%10
bai = i%100
if i == ge*ge*ge+shi*shi*shi+bai*bai*bai:
    sum+=1
```

☐ **D.**

```
ge = i%10
shi = i%10%10
bai = i%100
if i == ge*ge*ge+shi*shi*shi+bai*bai*bai:
    sum+=1
```

第 14 题 下面程序输出的是

```python
def func(x):
    if x%2 == 1:
        return x+1
    else:
        return func(x-1)
print(func(9))
print(func(6))
```

☐ **A.**

```
10
6
```

☐ **B.**

```
9
6
```

☐ **C.**

```
10
7
```

☐ **D.**

```
9
7
```

第 15 题 旋转数组是一种常见的数据结构问题，通常是指一个有序数组经过旋转后，使得所有元素逆序排列。整数数组 nums 按升序排列，数组中的值互不相同。在预先未知的某个下标 k（0 <= k < nums.length）上进行了旋转，使数组变为 [nums[k], nums[k + 1],..., nums[n - 1], nums[0], nums[1],..., nums[k - 1]]（下标从 0 开始计数）。

现在给定旋转后的数组 nums 和一个整数 target，如果 nums 中存在这个目标值 target，则返回它的下标，否则返回 -1。

下面程序中（）处应填入的程序是：例如，给定一个数组 [4,5,6,7,0,1,2]，它可能经过旋转变为 [0,1,2,4,5,6,7]。二分查找算法搜索旋转排序数组的程序，下面横线中，应填入的一行或多行代码是（ ）

```python
def search(nums, target):
    left, right = 0, len(nums) - 1

    while left <= right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return True


        if nums[mid] > nums[right]:

            if nums[mid] > target or nums[left] <= target:
                right = mid - 1
            else:
                left = mid + 1

        elif nums[mid] < nums[right]:

```

```
            ____________________
        else:

            right -= 1

    return False

```

☐ **A.**

```
if nums[mid] < target or nums[right] >= target:
    left = mid - 1
else:
    right = mid + 1
```

☐ **B.**

```
if nums[mid-1] < target and nums[right+1] >= target:
    left = mid + 1
else:
    right = mid - 1
```

☐ **C.**

```
if nums[mid] < target or nums[right] >= target:
    left = mid
else:
    right = mid
```

☐ **D.**

```
if nums[mid] < target and nums[right] >= target:
    left = mid + 1
else:
    right = mid - 1
```

2　判断题（每题 2 分，共 20 分）

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

第 1 题 两个整数的最大公约数等于其中较小的那个数和两数相除余数的最大公约数

第 2 题 任何一个大于1的自然数都可以分解成若干个不同的质数的乘积，且分解方式是唯一的。

第 3 题 要得到不大于某个自然数 n（不等于0）的所有素数，只要在2至 n 中将不大于$\sqrt{n}$ 的素数的倍数全部划去即可

第 4 题 任何一个大于1的自然数，要么所有质因子都小于等于 $\sqrt{n}$，要么只有一个质因子大于 $\sqrt{n}$，其余质因子都小于 $\sqrt{n}$。

第 5 题 贪心算法的空间复杂度通常是 $O(1)$

第 6 题 归并排序的空间复杂度为 $O(n)$

第 7 题 时间复杂度对比 $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2 \log n) < O(n^2) < O(n^3)$。

第 8 题 对于任意整数 $a \equiv a \pmod{m}$。

第 9 题 下列程序输出的是 `21`

```
def digui(n):
    if n <= 1:
        return n
    else:
        return digui(n-1) + digui(n-2)

print(digui(8))
```

第 10 题 CCF(十六进制) = 1653(13进制)

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：奇妙数字
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题面描述

小杨认为一个数字 x 是奇妙数字当且仅当 $x = p^a$，其中 p 为任意质数且 a 为正整数。例如，$8 = 2^3$，所以 8 是奇妙数字，而 6 不是。

对于一个正整数 n，小杨想要构建一个包含 m 个奇妙数字的集合 $\{x_1, x_2, \ldots, x_m\}$，使其满足以下条件：

- 集合中不包含相同的数字。
- $x_1 * x_2 * \ldots * x_m$ 是 n 的因子（即 $x_1, x_2, \ldots, x_m$ 这 m 个数字的乘积是 n 的因子）。

小杨希望集合包含的奇妙数字尽可能多，请你帮他计算出满足条件的集合最多包含多少个奇妙数字。

3.1.2 输入格式

第一行包含一个正整数 n，含义如题面所示。

3.1.3 输出格式

输出一个正整数，代表满足条件的集合最多包含的奇妙数字个数。

3.1.4 样例

```
128
```

```
3
```

3.1.5 样例解释

关于本样例，符合题意的一个包含 3 个奇妙数字的集合是 $2,4,8$。首先，因为 $2=2^1$，$4=2^2$，$8=2^3$，所以 $2,4,8$ 均为奇妙数字。同时，$2*4*8=64$ 是 128 的因子。
由于无法找到符合题意且同时包含 4 个奇妙数字的集合，因此本样例的答案为 3。

子任务编号	数据点占比	n
1	20%	≤ 10
2	20%	≤ 1000
3	60%	$\leq 10^{12}$

对于全部数据，保证有 $2 \leq n \leq 10^{12}$。

3.1.6 参考程序

```python
def calc(x):
    ans = 0
    tmp = 1
    while x >= tmp:
        ans += 1
        x -= tmp
        tmp += 1
    return ans

n = int(input())
ans = 0
i = 2

while i * i <= n:
    if n % i == 0:
        cnt = 0
        while n % i == 0:
            cnt += 1
            n //= i
        ans += calc(cnt)
    i += 1

if n != 1:
    ans += calc(1)

print(ans)
```

3.2 编程题 2

- **试题名称**：武器强化
- **时间限制**：2.0 s
- **内存限制**：512.0 MB

3.2.1 题面描述

小杨有 n 种武器和 m 种强化材料。第 i 种强化材料会适配第 p_i 种武器，小杨可以花费 c_i 金币将该材料对应的适配武器修改为任意武器。

小杨最喜欢第 1 种武器，因此他希望适配该武器的强化材料种类数**严格大于**其他的武器，请你帮小杨计算为了满足该条件最少需要花费多少金币。

3.2.2 输入格式

第一行包含两个正整数 n, m，含义如题面所示。

之后 m 行，每行包含两个正整数 p_i, c_i，代表第 i 种强化材料的适配武器和修改花费。

3.2.3 输出格式

输出一个整数，代表能够使适配第 1 种武器的强化材料种类数**严格大于**其他的武器最少需要花费的金币。

3.2.4 样例

```
4 4
1 1
2 1
3 1
3 2
```

```
1
```

3.2.5 样例解释

花费 1，将第三种强化材料的适配武器由 3 改为 1。此时，武器 1 有 2 种强化材料适配，武器 2 和武器 3 都各有 1 种强化材料适配。满足适配第 1 种武器的强化材料种类数**严格大于**其他的武器。

子任务编号	数据点占比	n	m
1	20%	2	≤ 1000
2	20%	≤ 1000	2
3	60%	≤ 1000	≤ 1000

对于全部数据，保证有 $1 \leq n, m \leq 1000, 1 \leq p_i \leq n, 1 \leq c_i \leq 10^9$。

3.2.6 参考程序

```python
def calc(aim, n, cnt, cs):
    cur_cnt = cnt[0]
    res = 0
    tmp = []
    for i in range(1, n):
        buy = max(len(cs[i]) - aim + 1, 0)
        for j in range(buy):
            res += cs[i][j]
        cur_cnt += buy
        for j in range(buy, len(cs[i])):
            tmp.append(cs[i][j])

    tmp.sort()
    for i in range(aim - cur_cnt):
```

```
        res += tmp[i]

    return res

n, m = map(int, input().split())
cnt = [0] * n
cs = [[] for _ in range(n)]

for _ in range(m):
    p, c = map(int, input().split())
    cnt[p - 1] += 1
    cs[p - 1].append(c)

for i in range(n):
    cs[i].sort()

ans = 1000000000000000000
for i in range(max(cnt[0], 1), m + 1):
    ans = min(ans, calc(i, n, cnt, cs))

print(ans)
```