[2025年第十六届蓝桥杯软件赛省赛Python大学C组真题]

一：填空题

1.偏蓝

题目背景

本站蓝桥杯 2025 省赛测试数据均为洛谷自造，与官方数据可能存在差异，仅供学习参考。

题目描述

小蓝特别喜欢蓝色。最近，小蓝学习了颜色在计算机中的一种表示方法：用三个 0 至 255 之间的整数（包含 0 和 255）分别表示颜色的红、绿、蓝三个分量。

在这种颜色的表示方法下，小蓝定义了一种颜色是偏蓝的，是指蓝色分量大于红色分量，且蓝色分量大于绿色分量。例如，红、绿、蓝分别为 10、10、11 时是偏蓝的；红、绿、蓝分别为 100、200、200 时不是偏蓝的。

小蓝想知道，有多少种不同的颜色是偏蓝的。两种颜色如果在红、绿、蓝中至少有一个分量值不同，就认为是不同的。

输入格式

无

输出格式

这是一道结果填空的题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只需要**编写一个程序输出这个整数**，输出多余的内容将无法得分。

输入输出样例

无

答案：5559680

钦定 b，则我们要统计满足 $r < b, g < b$ 的数量，这里贡献了 b^2，故我们需要求：

$$\sum_{i=0}^{255} b^2.$$

使用平方和公式计算 $\frac{255 \times 256 \times 511}{6} = 5559680$，输出即可。

2.2025

题目描述

求 1 ～ 20250412 中，有多少个数可以通过改变其数字顺序后含有 2025。

例如，5220、21520 可以，而 205、225、2200、222555111 则不行。

提示：要求的数就是含有至少 1 个 0、2 个 2、1 个 5 的数。

输入格式

无

输出格式

这是一道结果填空题，你只需要算出结果后提交即可。本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

输入输出样例

无

答案：506754

解题方法：统计从 1 到 20250412 中满足至少有 1 个 0、至少 2 个 2 和至少 1 个 5 的数字的个数。

二：编程题

3.2025图形

小蓝要画一个 2025 图形。图形的形状为一个 h × w 的矩形，其中 h 表示图形的高，w 表示图形的宽。当 h = 5,w = 10 时，图形如下所示：

2025202520

0252025202

2520252025

5202520252

2025202520

图形的规律是：第一行用 2025 重复填入，第二行开始，每行向左移动一个 字符，用 2025 重复填入。

给定 h, w ，请输出对应的图形。

输入格式

输入的第一行包含两个正整数 h,w ，用一个空格分隔。

输出格式

输出若干行，表示对应的图形。

样例输入

复制

```
4 5
```

样例输出

复制

```
20252
02520
25202
52025
```

```python
h, w = map(int, input().split())
for i in range(h):
    start_index = i
    num_str = "2025"
    result = ""
    for j in range(w):
        result += num_str[(start_index + j) % 4]
print(result)
```

输入处理：

`h, w = map(int, input().split())` 这行代码用于接收用户输入的两个整数 `h` 和 `w`，分别表示图形的高和宽。`input().split()` 会将用户输入的字符串按空格分割成字符串列表，`map(int,...)` 会将列表中的每个字符串转换为整数，最后通过解包赋值给 `h` 和 `w`。

图形生成与输出：

外层的 `for i in range(h)` 循环控制行数，即遍历每一行。

对于每一行，`start_index = i` 确定该行起始字符在 `"2025"` 中的位置。

`num_str = "2025"` 定义了重复的数字字符串。

内层的 `for j in range(w)` 循环控制列数，在每一行中，`result += num_str[(start_index + j) % 4]` 用于确定当前位置的字符，通过 `(start_index + j) % 4` 来循环取 `"2025"` 中的字符，构建当前行的字符串。

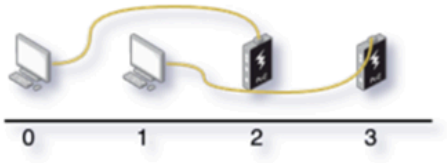
最后 `print(result)` 输出每一行的字符串。

4.最短距离

题目描述

在一条一维的直线上，存在着 n 台显示器和 n 个电源插座。老师给小蓝布置了个任务：负责将每台显示器通过电源线与一个插座相连接（每个插座最多只能给一台显示器供电）；同时，老师希望所消耗的电源线的长度尽可能的少， 请你帮小蓝计算下电源线的最小消耗长度为多少?

为了便于计算，你只需要考虑直线距离即可。



输入格式

输入的第一行包含一个正整数 n 。

接下来 n 行，每行包含一个整数 xi ，依次表示每台显示器的坐标。

接下来 n 行，每行包含一个整数 yi ，依次表示每个插座的坐标。

输出格式

输出一行包含一个整数表示答案。

样例输入

复制

```
2
0
1
2
3
```

样例输出

复制

```
4
```

提示

【评测用例规模与约定】

对于 20% 的评测用例，$1 \leqslant n \leqslant 10$ ，$0 \leqslant x_i, y_i \leqslant 100$ ；

对于 40% 的评测用例，$1 \leqslant n \leqslant 100$ ，$0 \leqslant x_i, y_i \leqslant 10^3$ ；

对于 60% 的评测用例，$1 \leqslant n \leqslant 1000$ ，$0 \leqslant x_i, y_i \leqslant 10^5$ ；

对于 80% 的评测用例，$1 \leqslant n \leqslant 10000$ ，$0 \leqslant x_i, y_i \leqslant 10^9$ ；

对于所有评测用例，$1 \leqslant n \leqslant 50000$ ，$0 \leqslant x_i, y_i \leqslant 10^9$ 。

```python
n = int(input())
monitor_positions = [int(input()) for _ in range(n)]
socket_positions = [int(input()) for _ in range(n)]

monitor_positions.sort()
socket_positions.sort()
```

```
min_length = 0
for i in range(n):
    min_length += abs(monitor_positions[i] - socket_positions[i])

print(min_length)
```

输入处理：
n = int(input()) 读取显示器和插座的数量 n。
monitor_positions = [int(input()) for _ in range(n)] 通过列表推导式，逐行读取 n 个显示器的坐标，并存储在列表 monitor_positions 中。
socket_positions = [int(input()) for _ in range(n)] 同理，逐行读取 n 个插座的坐标，存储在列表 socket_positions 中。
排序操作：
monitor_positions.sort() 对显示器坐标列表进行升序排序。
socket_positions.sort() 对插座坐标列表进行升序排序。
计算最小长度：
min_length = 0 初始化电源线的最小总长度为 0 。
通过 for i in range(n) 遍历每个显示器和插座的对应位置，min_length += abs(monitor_positions[i] - socket_positions[i]) 计算当前对应显示器和插座之间的距离（使用 abs 函数取绝对值确保距离为正），并累加到总长度 min_length 中。
结果输出：
print(min_length) 输出计算得到的电源线最小消耗长度。

样例验证
对于样例输入：
2
0
1
2
3

排序后，显示器坐标 [0, 1]，插座坐标 [2, 3]。
第一次循环：i = 0，abs(0 - 2) = 2 ，此时 min_length = 2。
第二次循环：i = 1，abs(1 - 3) = 2 ，min_length = 2 + 2 = 4 ，与样例输出一致。

5.倒水

输入格式

输入的第一行包含两个正整数 n, k ， 用一个空格分隔。

第二行包含 n 个正整数 $a_1, a_2, \cdots, a_n$ ， 相邻整数之间使用一个空格分隔。

输出格式

输出一行包含一个整数表示答案。

样例输入

复制

```
7 3
8 5 5 2 2 3 4
```

样例输出

复制

```
3
```

提示

【样例说明】

其中一种方案：a_1 往 a_4 倒入 3 单位；a_2 往 a_5 倒入 2 单位；a_3 往 a_6 倒入 1 单位；最终每个瓶子里的水：5, 3, 4, 5, 4, 4, 4 ， 最小值为 3 。

【评测用例规模与约定】

对于 40% 的评测用例，$1 \leqslant n, a_i \leqslant 100$ ；

对于所有评测用例，$1 \leqslant n, a_i \leqslant 100000$ ， $1 \leqslant k \leqslant n$ 。

```python
import sys
import copy

input = lambda: sys.stdin.readline().strip()
n, k = map(int, input().split())
ls = list(int(x) for x in input().split())
r = float(1e9)
nums = []


def test(n, nums):
    for ls in nums:
        for i in range(len(ls) - 1):
            if ls[i] > n:
                ls[i + 1] += ls[i] - n
                ls[i] = n
            elif ls[i] < n:
                return False
        if ls[-1] < n:
            return False
    return True
```

```python
for i in range(k):
    tmp = []
    s = 0
    for j in range(i, n, k):
        tmp.append(ls[j])
        s += ls[j]
    r = min(r, s // len(tmp))
    nums.append(tmp)
l = 0
ans = 0
while l < r:
    m = (r + l + 1) // 2
    if test(m, copy.deepcopy(nums)):
        ans = max(m, ans)
        l = m
    else:
        r = m - 1

print(ans)
```

注意到对于一个任意次操作后所有瓶子中的水的最小值 $\min\{a_1, a_2, \cdots, a_n\}$，小于它的值也一定可行，因此答案具有单调性，并且可以 $O(n)$ 检查一个值是否可行，因此可以使用二分答案。

将每一种相同颜色的水（也就是相隔 k 的数）分为一组，预处理得到每一组数平均值的最小值，记为 m，对 m 进行二分即可。那么时间复杂度就是 $O(n \log m)$，可以通过本题。

6.小说

题目描述

小蓝是一位网络小说家。现在他正在撰写一部新的推理小说，这部小说有 n 个不同的人物。

小说的每一章都有以下三种情节的一种：

1、A 发现 B 不知道真相。

2、A 发现 B 知道真相。

3、A 知道了真相。

为了保证读者的协调和新鲜感，小蓝的小说还要满足以下要求：

1、“ B 发现 A 不知道真相”不能在“ A 知道了真相”后。

2、“ B 发现 A 知道真相”不能在“ A 知道了真相”前。

3、“ B 发现 A 不知道真相”不能在“ B 发现 A 知道真相”后。

4、相邻的两章情节类型不同，例如如果第一章是 A 发现 B 不知道真相那 么第二章就不能是 C 发现 D 不知道真相。

5、完全相同的情节不能出现两次。

现在小蓝希望知道，他最多能写多少章。

输入格式

输入的第一行包含一个正整数 n ，表示小说人数。

输出格式

输出一行包含一个整数表示答案，即小蓝最多能写多少章小说。

样例输入

复制

```
2
```

样例输出

复制

```
6
```

提示

【样例说明 1】

以下是一种可能的情况：

1、B 发现 A 不知道真相。

2、A 知道了真相。

3、B 发现 A 知道真相。

4、A 发现 B 不知道真相。

5、B 知道了真相。

6、A 发现 B 知道真相。

小蓝一共能写 6 章。

【样例输入 2】

3

【样例输出 2】

13

【评测用例规模与约定】

对于 30% 的评测用例，$n \leqslant 5$；

对于所有评测用例，$1 \leqslant n \leqslant 10^9$。

```python
n = int(input())
if n == 1:
    print(1)
else:
    print(2 * n * n - 3 * n + 4)
```

要解决这个问题，我们需要理解小说的情节限制，并寻找出现最多章节的组合。我们需要考虑以下几点：

情节类型：

情节1：A 发现 B 不知道真相。
情节2：A 发现 B 知道真相。
情节3：A 知道了真相。
限制条件：

情节1不能在情节3的情节之后出现。
情节2不能在情节3的情节之前出现。
情节1不能在情节2之后出现。
相邻两章不能是相同类型的情节。
情节不能重复。
分析：

情节3的出现会影响情节1和情节2的顺序。
情节的顺序需要保持上述限制。
情节的组合需要确保每个情节的顺序符合要求，同时尽可能多地使用不同的人物组合。
计算：

对于每个人物组合（A，B），我们可以有三种情节类型。
为了在不违反任何规则的情况下最大化章节数，我们需要巧妙地安排情节。
公式推导：

每个人物可以与其他n-1个人物形成情节，因此有n*(n-1)种不同的人物对。
对于每对人物，我们可以有三种情节类型。
由于情节不能重复，我们可以为每个人物对安排三种情节。
情节的顺序需要满足限制，因此我们需要考虑如何安排这些情节。
最大章节数：

综合考虑所有限制，最大化章节数的公式为：2*n*n - 3*n + 4。
这个公式考虑了所有可能的人物对组合和情节类型，同时满足不同类型的相邻章节限制。
注意事项
确保每个情节都符合顺序和重复限制。
确保情节的切换遵循相邻类型不同的规则。
情节的排列需要考虑所有可能的情节类型组合。

7.拼好数

题目描述

我们将含有不少于 6 个 6 的数视为一个好数。例如 666666, 162636465666 是好数，12366666 不是好数。

给定 n 个正整数 a_i，你可以把这些数分成若干组拼起来，每组内的数可以 按任意顺序拼，但一组最多只能有 3 个数。求最多可以得到多少个好数。

输入格式

输入的第一行包含一个正整数 n 。

第二行包含 n 个正整数 $a_1, a_2, \cdots, a_n$ ，相邻整数之间使用一个空格分隔。

输出格式

输出一行包含一个整数表示答案，即最多可以得到的好数的数量

样例输入

复制

```
3
66 66 66
```

样例输出

复制

```
1
```

提示

【样例输入 2】

7 666666 16166 6696 666 6 6 6

【样例输出 2】

2

【评测用例规模与约定】

对于 70% 的评测用例，$1 \leqslant n \leqslant 20$；

对于所有评测用例，$1 \leqslant n \leqslant 1000$，$0 \leqslant a_i \leqslant 10^9$。

```
n = int(input())
ans = 0  # 最终结果
arr = [0] * (n + 1)
cnt = [0] * 11

for i in range(1, n + 1):
    arr[i] = int(input())
    c = 0  # 每个数的6的个数
    num = arr[i]
    while num > 0:
        if num % 10 == 6:
            c += 1
        num //= 10
    cnt[c] += 1
    if c >= 6:
        ans += 1

while cnt[5] > 0:
    # 5+1->5+2->5+3->5+4->5+5
    for i in range(1, 6):
        if cnt[i] > 0:
            cnt[i] -= 1
            ans += 1
            break
    cnt[5] -= 1

while cnt[4] > 0:
    # 4+1+1
    if cnt[1] > 1:
        cnt[1] -= 2
        ans += 1
    else:
        for i in range(2, 5):
            # 4+2->4+3->4+4
            if cnt[i] > 0:
                cnt[i] -= 1
                ans += 1
                break
    cnt[4] -= 1

while cnt[3] > 0:
    # 3+1+2
    if cnt[1] > 0 and cnt[2] > 0:
```

```
            cnt[1] -= 1
            cnt[2] -= 1
            ans += 1
        # 3+3
        elif cnt[3] > 1:
            cnt[3] -= 1
            ans += 1
        # 3+2+2
        elif cnt[2] >= 2:
            cnt[2] -= 2
            ans += 1
        cnt[3] -= 1

while cnt[2] > 2:
    # 2+2+2
    ans += 1
    cnt[2] -= 3

print(ans)
```

这道题先用一个cnt[]记录6的个数，对于>=6的数自己就可以组成一个好数，可以直接计入最终结果，那么 接下来只用考虑各个数位6的个数小于等于5的数了。

- 从5开始拿，选哪些数和5组合呢？优先拿已有的最小的，便是1，如果1不存在，再拿2，以此类推，直到5。记住每成功拿出一个，对应的cnt[]都要减1，以免重复拿。

- 再从4开始拿，确定要拿4，优先拿已有的最小的，便是1，拿三个数就是411，拿两个数就是4和任一个大于1且小于5的数就行。

- 再拿3，无非就是312,33,322，这里先拿33再拿322因为322加起来是7,312加起来是6，先拿最节省的组合。

- 最后拿2，那只能222的组合了。

- 优先顺序是 5+1->5+2->5+3->5+4->5+5->4+1+2-> 4+2->4+3->4+4->3+3-> 3+2+2->2+2+2

8.二进制

【问题描述】

给定一个由 0,1,2,3⋯ 的二进制表示拼接而成的长度无限的 01 串。其前若干位形如 011011100101110111⋯ 。

请求出这个串的前 x 位里有多少个 1 。

【输入格式】

输入的第一行包含一个正整数 x 。

【输出格式】

输出一行包含一个整数表示答案。

【样例输入】

```
7
```

【样例输出】

```
5
```

【样例说明】

给定的串的前 7 位为 0110111 。

```python
x = int(input())
ans = 0
rem = x
n = 0
while rem > 0:
    bin_n = bin(n)[2:]  # 去掉 '0b' 前缀
    len_bin = len(bin_n)
    if len_bin <= rem:
        ans += bin_n.count('1')
        rem -= len_bin
    else:
        ans += bin_n[:rem].count('1')
        rem = 0
    n += 1
print(ans)
```

初始化答案 ans 为 0，剩余位数 rem 为 x。从数字 \(n = 0\) 开始循环：获取 n 的二进制表示（去掉前缀 0b）。计算二进制表示的长度 len_bin。如果 len_bin 小于等于 rem，则将该二进制表示中 1 的个数加到 ans，并减少 rem。否则，取该二进制表示的前 rem 位，计算其中 1 的个数加到 ans，并将 rem 置为 0。处理完一个数字后，n 自增 1。当 rem 为 0 时，循环结束，输出 ans。