Assignment #2: DSA & OOP

Updated 1748 GMT+8 Sep 15, 2025

2025 fall, Complied by 林奕妃、环境科学与工程学院

作业的各项评分细则及对应的得分

标准	等级	得分
按时提交	完全按时提交：1分 提交有请假说明：0.5分 未提交：0分	1分
源码、耗时（可选）、解题思路（可选）	提交了4个或更多题目且包含所有必要信息：1分 提交了2个或以上题目但不足4个：0.5分 少于2个：0分	1分
AC代码截图	提交了4个或更多题目且包含所有必要信息：1分 提交了2个或以上题目但不足4个：0.5分 少于：0分	1分
清晰头像、PDF文件、MD/DOC附件	包含清晰的Canvas头像、PDF文件以及MD或DOC格式的附件：1分 缺少上述三项中的任意一项：0.5分 缺失两项或以上：0分	1分
学习总结和个人收获	提交了学习总结和个人收获：1分 未提交学习总结或内容不详：0分	1分
总得分： 5	总分满分：5分	

> **说明：**
>
> 1. **解题与记录：**
>
>   对于每一个题目，请提供其解题思路（可选），并附上使用Python或C++编写的源代码（确保已在OpenJudge， Codeforces，LeetCode等平台上获得Accepted）。请将这些信息连同显示“Accepted”的截图一起填写到下方的作业模板中。（推荐使用Typora https://typoraio.cn 进行编辑，当然你也可以选择Word。）无论题目是否已通过，请标明每个题目大致花费的时间。
>
> 2. **课程平台：**课程网站位于Canvas平台（https://pku.instructure.com ）。该平台将在第2周选课结束后正式启用。在平台启用前，请先完成作业并将作业妥善保存。待Canvas平台激活后，再上传你的作业。
>
> 3. **提交安排：**提交时，请首先上传PDF格式的文件，并将.md或.doc格式的文件作为附件上传至右侧的“作业评论”区。确保你的Canvas账户有一个清晰可见的本人头像，提交的文件为PDF格式，并且“作业评论”区包含上传的.md或.doc附件。

> 4. **延迟提交**：如果你预计无法在截止日期前提交作业，请提前告知具体原因。这有助于我们了解情况并可能为你提供适当的延期或其他帮助。
>
> 请按照上述指导认真准备和提交作业，以保证顺利完成课程要求。

1. 题目

20742: 泰波拿契数（15分钟）

http://cs101.openjudge.cn/practice/20742/

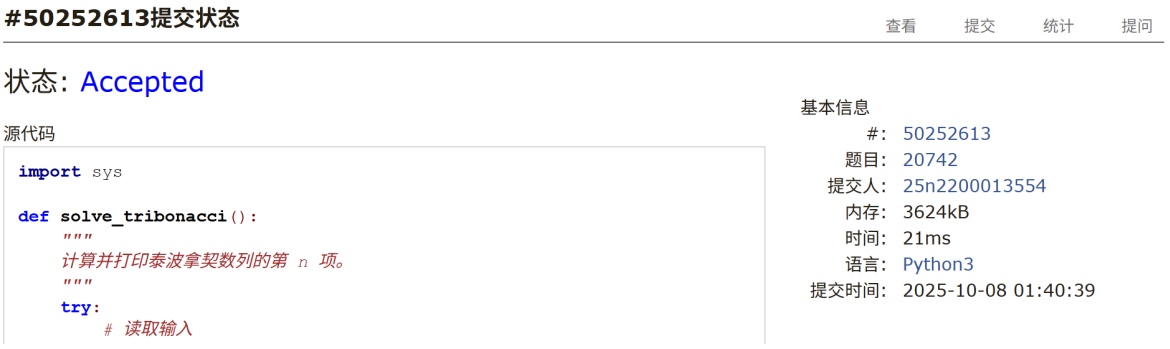
思路：设定好初值后后续数字按照迭代计算即可。

代码：

```
import sys

def solve_tribonacci():
    """
    计算并打印泰波拿契数列的第 n 项。
    """
    try:
        # 读取输入
        n = int(sys.stdin.readline())
    except (ValueError, IndexError):
        # 处理可能的空输入或格式错误
        return

    # 1. 处理基本情况 (Base Cases)
    # 根据题目约束 1 <= n <= 30, n=0 的情况其实不会出现
    if n == 0:
        print(0)
        return
    if n == 1 or n == 2:
        print(1)
        return

    # 2. 迭代计算
    # 初始化前三项 T0, T1, T2
    a, b, c = 0, 1, 1

    # 从第 3 项开始循环计算到第 n 项
    # 循环 n-2 次即可 (因为 T2 已经有了)
    for _ in range(3, n + 1):
        # 计算下一项
        # Tn = T(n-3) + T(n-2) + T(n-1)
        next_term = a + b + c

        # 更新 a, b, c 的值，向前滚动
        a = b
        b = c
        c = next_term
```

```
    # 循环结束后，c 中存放的就是 Tn 的值
    print(c)

# 调用函数执行
solve_tribonacci()
```

代码运行截图（至少包含有"Accepted"）



#50252613提交状态

查看 提交 统计 提问

状态：Accepted

源代码

```
import sys

def solve_tribonacci():
    """
    计算并打印泰波拿契数列的第 n 项。
    """
    try:
        # 读取输入
```

基本信息

#:	50252613
题目:	20742
提交人:	25n2200013554
内存:	3624kB
时间:	21ms
语言:	Python3
提交时间:	2025-10-08 01:40:39

58A. Chat room（20分钟）

greedy/strings, 1000, http://codeforces.com/problemset/problem/58/A

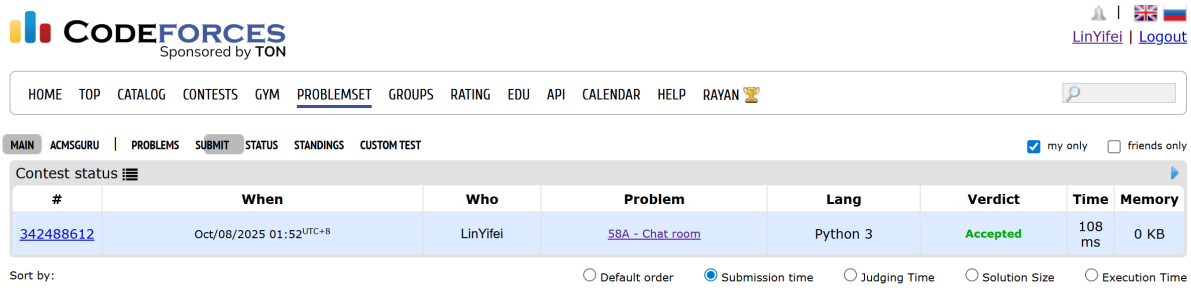
思路：问题要求判断一个单词是否具有hello作为子序列，可以使用贪心算法进行匹配。

代码：

```
import sys

def solve():
    """
    解决 Chat room 问题
    """
    # 读取输入的单词
    s = sys.stdin.readline().strip()

    # 目标单词
    target = "hello"

    # 指向 target 单词中当前要匹配的字符的索引
    target_ptr = 0

    # target 的长度
    target_len = len(target)

    # 遍历输入的字符串 s
    for char in s:
        # 如果我们已经找完了 "hello" 的所有字符，就没必要继续了
        if target_ptr == target_len:
            break

        # 如果当前 s 中的字符和我们正在寻找的 target 字符匹配
```

```
        if char == target[target_ptr]:
            # 那么我们就去寻找 target 的下一个字符
            target_ptr += 1

    # 遍历结束后，检查是否找完了 target 的所有字符
    if target_ptr == target_len:
        print("YES")
    else:
        print("NO")

# 调用函数执行
solve()
```

代码运行截图 （至少包含有"Accepted"）



118A. String Task（15分钟）

implementation/strings, 1000, http://codeforces.com/problemset/problem/118/A

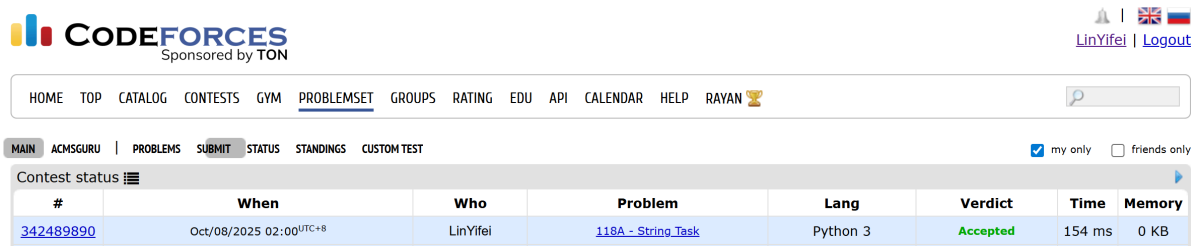
思路：遍历输入字符串的每一个字符，然后根据这个字符是元音还是辅音来决定如何处理。

代码：

```
import sys

def solve():
    """
    解决 String Task 问题
    """
    # 读取输入的字符串
    s = sys.stdin.readline().strip()

    # 根据题目定义，'y' 也是元音。
    # 使用集合 (set) 可以快速查找。
    vowels = {'a', 'o', 'y', 'e', 'u', 'i'}

    # 用于存储结果的列表
    result_list = []

    # 遍历输入字符串的每一个字符
    for char in s:
        # 1. 统一转换为小写，便于判断和处理
```

```
        lower_char = char.lower()

        # 2. 判断是否为辅音
        if lower_char not in vowels:
            # 3. 如果是辅音，按要求添加 '.' 和小写字母
            result_list.append('.')
            result_list.append(lower_char)

    # 4. 将列表中的所有元素连接成一个字符串并打印
    print("".join(result_list))

# 调用函数执行
solve()
```

代码运行截图 （至少包含有"Accepted"）



22359: Goldbach Conjecture（25分钟）

http://cs101.openjudge.cn/practice/22359/

思路：由于题目提供的数字有范围，因此可以将该范围内的所有素数都罗列出来逐个进行比对直至找到要求的素数对，根据输入数字可划定遍历的范围至输入数字的1/2即可。

代码：

```
import sys

# --- 预处理：使用埃拉托斯特尼筛法找出 10000 以内的所有素数 ---

# 设置最大值上限
MAX_NUM = 10000

# 创建一个布尔列表，is_prime[i] 为 True 表示 i 是素数
is_prime = [True] * (MAX_NUM + 1)
is_prime[0] = is_prime[1] = False  # 0 和 1 不是素数

# 执行筛法
# 我们只需要遍历到 sqrt(MAX_NUM) 即可
for p in range(2, int(MAX_NUM**0.5) + 1):
    # 如果 p 仍然是素数
    if is_prime[p]:
        # 那么 p 的所有倍数都不是素数
        # 我们可以从 p*p 开始标记，因为比 p*p 小的 p 的倍数
        # (例如 2p, 3p) 肯定已经被更小的素数 (2, 3) 标记过了
```

```
            for multiple in range(p * p, MAX_NUM + 1, p):
                is_prime[multiple] = False

def solve():
    """
    主解决函数
    """
    try:
        # 读取输入的和
        s = int(sys.stdin.readline())
    except (ValueError, IndexError):
        return

    # --- 寻找素数对 ---
    # 从最小的素数 2 开始，遍历到 s/2
    for a in range(2, s // 2 + 1):
        b = s - a

        # 使用预先计算好的列表来检查 a 和 b 是否都是素数
        # 这是一个 O(1) 的查找操作
        if is_prime[a] and is_prime[b]:
            # 找到第一对就输出并结束
            print(a, b)
            return

# 调用函数执行
solve()
```

代码运行截图 （至少包含有"Accepted"）

#50252622提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys

# --- 预处理：使用埃拉托斯特尼筛法找出 10000 以内的所有素数 ---

# 设置最大值上限
MAX_NUM = 10000
```

基本信息

#: 50252622
题目: 22359
提交人: 25n2200013554
内存: 3660kB
时间: 26ms
语言: Python3
提交时间: 2025-10-08 02:06:52

E23563: 多项式时间复杂度（20分钟）

http://cs101.openjudge.cn/pctbook/E23563/

思路：需要从表示多项式的字符串中找到非零系数最高次幂，可以遍历查找但要注意排除类似0n^100项。

代码

```
import sys

def solve():
```

```
"""
解决多项式时间复杂度问题
"""
try:
    # 读取输入行并移除可能存在的空白符
    line = sys.stdin.readline().strip()
    if not line:
        return
except (IOError, ValueError):
    return

# 1. 按 '+' 分割字符串，得到所有项
terms = line.split('+')

# 初始化最高次幂为 -1，这样可以正确处理常数项 (n^0)
max_power = -1

# 2. 遍历每一个项
for term in terms:
    coeff = 0
    power = 0

    # --- 3. 解析项，提取系数和指数 ---
    if 'n' in term:
        # 项包含变量 'n'
        parts = term.split('n')
        coeff_str = parts[0]
        power_str = parts[1]

        # 解析系数
        if coeff_str == "":
            coeff = 1
        else:
            coeff = int(coeff_str)

        # 解析指数
        if power_str == "": # 形如 '5n'
            power = 1
        else: # 形如 '5n^3'，power_str 是 '^3'
            power = int(power_str[1:]) # 取 '^' 之后的部分

    else:
        # 项不包含 'n'，是常数项
        coeff = int(term)
        power = 0

    # --- 4. 根据非零系数更新最高次幂 ---
    if coeff != 0:
        if power > max_power:
            max_power = power

# 5. 处理所有项系数都为 0 的情况
if max_power == -1:
    max_power = 0

# 6. 按格式输出结果
```

```
    print(f"n^{max_power}")

# 调用函数执行
solve()
```

(至少包含有"Accepted")

#50252625提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys

def solve():
    """
    解决多项式时间复杂度问题
    """
    try:
```

基本信息

#: 50252625
题目: E23563
提交人: 25n2200013554
内存: 3656kB
时间: 22ms
语言: Python3
提交时间: 2025-10-08 02:15:14

24684: 直播计票（20分钟）

http://cs101.openjudge.cn/practice/24684/

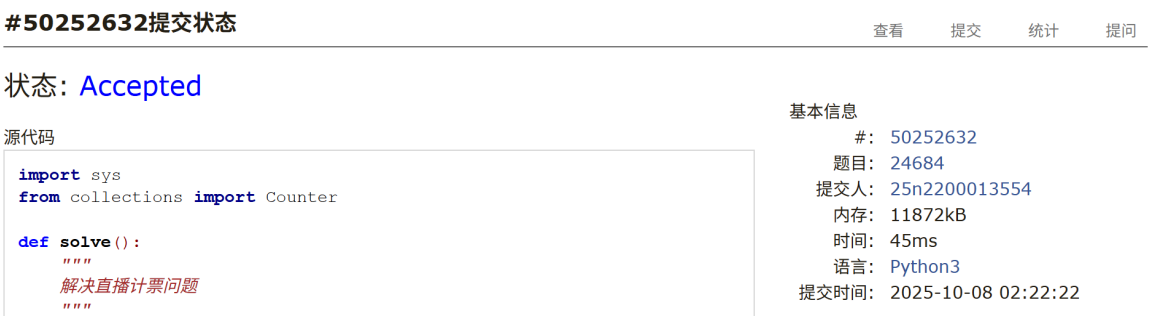
思路：先进行计数再从中找到个数最多的号码

代码

```
import sys
from collections import Counter

def solve():
    """
    解决直播计票问题
    """
    # 1. 读取输入
    line = sys.stdin.readline().strip()

    # 处理空输入的边界情况
    if not line:
        return

    # 将输入的字符串行分割并转换为整数列表
    votes = map(int, line.split())

    # 2. 使用 Counter 自动完成计票
    vote_counts = Counter(votes)

    # 3. 找到最大票数
    max_votes = max(vote_counts.values())

    # 4. 找出所有得票数等于最大票数的获胜者
    winners = []
    # .items() 会返回 (选项, 票数) 的元组对
```

```
    for option, count in vote_counts.items():
        if count == max_votes:
            winners.append(option)

    # 5. 按编号从小到大排序
    winners.sort()

    # 6. 格式化输出
    # map(str, winners) 将列表中的所有整数转换为字符串
    # " ".join(...) 将字符串列表用空格连接成一个字符串
    print(" ".join(map(str, winners)))

# 调用函数执行
solve()
```

（至少包含有"Accepted"）



#50252632提交状态

查看　提交　统计　提问

状态：Accepted

源代码

```
import sys
from collections import Counter

def solve():
    """
    解决直播计票问题
    """
```

基本信息

#:	50252632
题目:	24684
提交人:	25n2200013554
内存:	11872kB
时间:	45ms
语言:	Python3
提交时间:	2025-10-08 02:22:22

2. 学习总结和个人收获

python语法相对于c语言更加简洁，有许多好用且方便的函数可调用，我对于这个并不是很了解。同时本次作业中例如素数题目涉及一些数学知识，需要我具有相关的数学理解才能更好地设计代码求解，这对我也有一些挑战。此前在计算方法课程中也有学到计算的时间复杂度等相关知识，对应采用的是C++代码，本课程老师也有讲解算法的时间复杂度与空间复杂度相关知识。老师选取的题目都很有学习价值，我要努力补齐不熟悉的知识。