Assignment #1: 虚拟机，Shell & 大模型

Updated 1317 GMT+8 Feb 20, 2025

2025 spring, Complied by **袁奕 2400010766 数院**

作业的各项评分细则及对应的得分情况

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 没有提供源码：0分	1分
AC代码截图	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
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. **课程平台与提交安排：**
    - 我们的课程网站位于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。

> - 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。
> 3. **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

27653: Fraction类

http://cs101.openjudge.cn/practice/27653/

代码：

```
from math import gcd

class frac():
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def __add__(self, other):
        na = self.a * other.b + self.b * other.a
        nb = self.b * other.b
        return frac(na // gcd(na, nb), nb // gcd(na, nb))
    def __str__(self):
        return f"{self.a}/{self.b}"
    def show(self):
        print(f"{self.a}/{self.b}")

a, b, c, d = map(int, input().split())
print(frac(a, b) + frac(c, d))
```

状态：Accepted

源代码

```
from math import gcd

class frac():
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def __add__(self, other):
        na = self.a * other.b + self.b * other.a
        nb = self.b * other.b
        return frac(na // gcd(na, nb), nb // gcd(na, nb))
    def __str__(self):
        return f"{self.a}/{self.b}"
    def show(self):
        print(f"{self.a}/{self.b}")

a, b, c, d = map(int, input().split())
print(frac(a, b) + frac(c, d))
```

1760.袋子里最少数目的球

https://leetcode.cn/problems/minimum-limit-of-balls-in-a-bag/

思路：

代码：

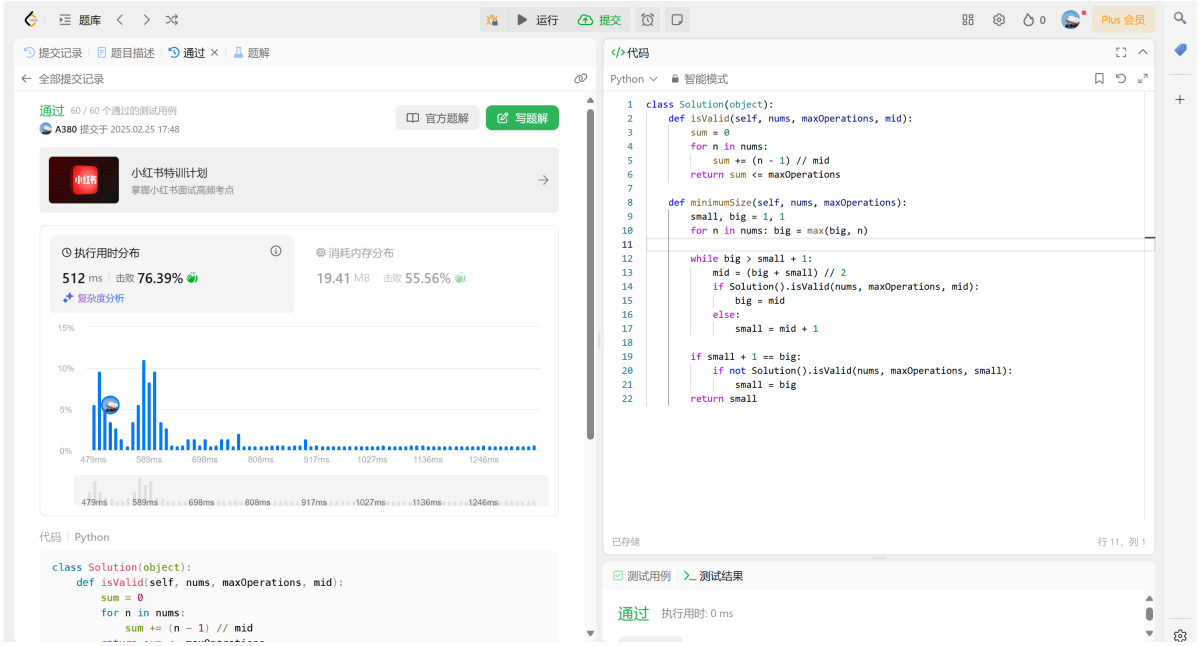
```
class Solution(object):
    def isValid(self, nums, maxOperations, mid):
        sum = 0
        for n in nums:
            sum += (n - 1) // mid
        return sum <= maxOperations

    def minimumSize(self, nums, maxOperations):
        small, big = 1, 1
        for n in nums: big = max(big, n)

        while big > small + 1:
            mid = (big + small) // 2
            if Solution().isValid(nums, maxOperations, mid):
                big = mid
            else:
                small = mid + 1

        if small + 1 == big:
```

```
            if not Solution().isValid(nums, maxOperations, small):
                small = big
        return small
```



04135: 月度开销

http://cs101.openjudge.cn/practice/04135

思路:

代码:

```
def isValid(nums, mid, m):
    cnt, sum = 1, 0
    for i, num in enumerate(nums):
        new_sum = sum + num
        if new_sum <= mid:
            sum = new_sum
        else:
            sum = num
            cnt += 1
        # 第 i 个循环完成后, cnt 和 sum 分别是前 i 项的
    return cnt <= m

n, m = map(int, input().split())
nums = []
small, big = 0, 0
for _ in range(n):
    num = int(input())
    nums.append(num)
    small = max(small, num)
    big += num

```

```
while small + 1 < big:
    mid = (small + big) // 2
    if isValid(nums, mid, m):
        big = mid
    else :
        small = mid + 1

if small < big:
    if not isValid(nums, small, m):
        small = big

print(small)
```

状态: Accepted

源代码

```
def isValid(nums, mid, m):
    cnt, sum = 1, 0
    for i, num in enumerate(nums):
        new_sum = sum + num
        if new_sum <= mid:
            sum = new_sum
        else:
            sum = num
            cnt += 1
    return cnt <= m

n, m = map(int, input().split())
nums = []
small, big = 0, 0
for _ in range(n):
    num = int(input())
    nums.append(num)
    small = max(small, num)
    big += num

while small + 1 < big:
    mid = (small + big) // 2
    if isValid(nums, mid, m):
        big = mid
    else :
        small = mid + 1

if small < big:
    if not isValid(nums, small, m):
        small = big

print(small)
```

27300: 模型整理

http://cs101.openjudge.cn/practice/27300/

Remark : 其中 OOP 写法类似 CPP 中自定义的 cmp 比较函数

代码:

```
class token():
    def __init__(self, val : str):
        self.val = val
    def num(self):
        if self.val[-1] == "M":
            return float(self.val[:-1])
        return float(self.val[:-1]) * 1e3
    def __lt__(self, other):
       return token.num(self) < token.num(other)
    def __str__(self):
        return self.val

n = int(input())
pair = {}
for _ in range(n):
    s = input()
    mid = s.index("-")
    name, num = s[:mid], token(s[mid + 1:])
    if name in pair:
        pair[name].append(num)
    else:
        pair[name] = [num]

for name, num in pair.items():
    pair[name] = sorted(num)

names = list(pair.keys())
names = sorted(names)
for name in names:
    print(name, end = ": ")
    print(*pair[name], sep = ", ")
```

状态：Accepted

源代码

```
class token():
    def __init__(self, val : str):
        self.val = val
    def num(self):
        if self.val[-1] == "M":
            return float(self.val[:-1])
        return float(self.val[:-1]) * 1e3
    def __lt__(self, other):
       return token.num(self) < token.num(other)
    def __str__(self):
        return self.val

n = int(input())
pair = {}
for _ in range(n):
    s = input()
    mid = s.index("-")
    name, num = s[:mid], token(s[mid + 1:])
    if name in pair:
        pair[name].append(num)
    else:
        pair[name] = [num]

for name, num in pair.items():
    pair[name] = sorted(num)

names = list(pair.keys())
names = sorted(names)
for name in names:
    print(name, end = ": ")
    print(*pair[name], sep = ", ")
```

Q5. 大语言模型（LLM）部署与测试

本任务旨在本地环境或通过云虚拟机（如 https://clab.pku.edu.cn/ 提供的资源）部署大语言模型（LLM）并进行测试。用户界面方面，可以选择使用图形界面工具如 https://lmstudio.ai 或命令行界面如 https://www.ollama.com 来完成部署工作。

测试内容包括选择若干编程题目，确保这些题目能够在所部署的LLM上得到正确解答，并通过所有相关的测试用例（即状态为Accepted）。选题应来源于在线判题平台，例如 OpenJudge、Codeforces、LeetCode 或洛谷等，同时需注意避免与已找到的AI接受题目重复。已有的AI接受题目列表可参考以下链接：
https://github.com/GMyhf/2025spring-cs201/blob/main/AI_accepted_locally.md

请提供你的最新进展情况，包括任何关键步骤的截图以及遇到的问题和解决方案。这将有助于全面了解项目的推进状态，并为进一步的工作提供参考。

Q6. 阅读《Build a Large Language Model (From Scratch)》第一章

作者：Sebastian Raschka

请整理你的学习笔记。这应该包括但不限于对第一章核心概念的理解、重要术语的解释、你认为特别有趣或具有挑战性的内容，以及任何你可能有的疑问或反思。通过这种方式，不仅能巩固你自己的学习成果，也能帮助他人更好地理解这一部分内容。

2. 学习总结和个人收获

假期和最近做了如下题目：

204	01760: Disk Tree	Trie	-	http://cs101.openjudge.cn/practice/01760/
204	124.二叉树中的最大路径和	dfs	Tough	https://leetcode.cn/problems/binary-tree-maximum-path-sum/
204	199.二叉树的右视图	bfs	Medium	https://leetcode.cn/problems/binary-tree-right-side-view/
203	01703: 发现它，抓住它	disjoint set	-	http://cs101.openjudge.cn/practice/01703/
203	01611: The Suspects	disjoint set	-	http://cs101.openjudge.cn/practice/01611/
203	01182: 食物链	disjoint set	Tough	http://cs101.openjudge.cn/practice/01182
202	01145: Tree Summing	-	-	http://cs101.openjudge.cn/practice/01145/
202	02788: 二叉树（2）	-	-	http://cs101.openjudge.cn/practice/02788/
202	02756: 二叉树（1）	-	-	http://cs101.openjudge.cn/practice/02756/
201	02524: 宗教信仰	disjoint set	必须会	http://cs101.openjudge.cn/dsapre/02524/
201	02499: Binary Tree	-	-	http://cs101.openjudge.cn/practice/02499/
201	02255: 重建二叉树	-	-	http://cs101.openjudge.cn/practice/02255/
131	27625: AVL树至少有几个结点	-	-	http://cs101.openjudge.cn/practice/27625/
131	05455: 二叉搜索树的层次遍历	-	-	http://cs101.openjudge.cn/practice/05455/
131	22275: 二叉搜索树的遍历	-	-	http://cs101.openjudge.cn/practice/22275/
130	18164: 剪绳子	-	必须会	http://cs101.openjudge.cn/practice/18164/
130	22161: 哈夫曼编码树	-	Tough	http://cs101.openjudge.cn/practice/22161/
130	晴问9.7: 向下调整构建大顶堆	-	-	https://sunnywhy.com/sfbj/9/7
129	04078: 实现堆结构	-	必须会	http://cs101.openjudge.cn/practice/04078/
129	25145: 猜二叉树（按层次遍历）	-	-	http://cs101.openjudge.cn/practice/25145/
129	24729: 括号嵌套树	-	-	http://cs101.openjudge.cn/practice/24729/
128	01577: Falling Leaves	-	-	http://cs101.openjudge.cn/practice/01577/
128	22158: 根据二叉树前中序序列建树	-	必须会	http://cs101.openjudge.cn/practice/22158/
128	24750: 根据二叉树中后序序列建树	-	必须会	http://cs101.openjudge.cn/practice/24750/
127	02775: 文件结构"图"	-	Tough	http://cs101.openjudge.cn/practice/02775/
127	25140: 根据后序表达式建立队列表达式	-		http://cs101.openjudge.cn/practice/25140/
127	102.二叉树的层序遍历	-	Easy	https://leetcode.cn/problems/binary-tree-level-order-traversal/
126	06646:二叉树的深度	-	-	http://cs101.openjudge.cn/practice/06646/
126	108.将有序数组转换为二叉搜索树	-	Easy	https://leetcode.cn/problems/convert-sorted-array-to-binary-search-tree/
126	100.相同的树	-	Easy	https://leetcode.cn/problems/same-tree/
125	08581: 扩展二叉树	-	-	http://cs101.openjudge.cn/practice/08581/
125	543.二叉树的直径	-	Easy	https://leetcode.cn/problems/diameter-of-binary-tree/
125	101.对称二叉树	-	Easy	https://leetcode.cn/problems/symmetric-tree/
124	27637: 括号嵌套二叉树	-	-	http://cs101.openjudge.cn/practice/27637/
124	226.翻转二叉树	-	Easy	https://leetcode.cn/problems/invert-binary-tree/
124	104.二叉树的最大深度	-	Easy	https://leetcode.cn/problems/maximum-depth-of-binary-tree/
123	27638: 求二叉树的高度和叶子数目	-	-	http://cs101.openjudge.cn/practice/27638/
123	94.二叉树的中序遍历	-	Easy	https://leetcode.cn/problems/binary-tree-inorder-traversal/
		-	-	tree begin
		-	-	tree begin
123	3095.或值至少K的最短子数组I	滑动窗口	-	https://leetcode.cn/problems/shortest-subarray-with-or-at-least-k-i/
122	04137: 最小新整数	monotonous-stack	-	http://cs101.openjudge.cn/practice/04137/
122	27925: 小组队列	queue	-	http://cs101.openjudge.cn/practice/27925/
122	155.最小栈	OOP辅助栈	Medium	https://leetcode.cn/problems/min-stack/
121	02299: Ultra-QuickSort	merge sort	Tough	http://cs101.openjudge.cn/practice/02299/
121	sy322: 跨步迷宫	bfs	Medium	https://sunnywhy.com/sfbj/8/2/322
121	sy323: 字符迷宫	bfs	Medium	https://sunnywhy.com/sfbj/8/2/323
120	03151: Pots	bfs	-	http://cs101.openjudge.cn/practice/03151/
120	sy320: 迷宫问题	bfs	Medium	https://sunnywhy.com/sfbj/8/2/320
120	05902: 双端队列	queue	-	http://cs101.openjudge.cn/practice/05902/
119	04067: 回文数字	queue	-	http://cs101.openjudge.cn/practice/04067/
119	19.删除链表的倒数第N个结点	快慢指针	Medium	https://leetcode.cn/problems/remove-nth-node-from-end-of-list/
119	24591:中序表达式转后序表达式	stack	必须会	http://cs101.openjudge.cn/practice/24591/
118	02746: 约瑟夫问题	queue	-	http://cs101.openjudge.cn/practice/02746/
118	24588: 后序表达式求值	stack	-	http://cs101.openjudge.cn/practice/24588/
118	20.删除链表元素	linked-list	-	http://dsbpython.openjudge.cn/dspythonbook/P0020/
117	02734: 十进制到八进制	stack	-	http://cs101.openjudge.cn/practice/02734/
117	4.插入链表元素	linked-list	-	http://dsbpython.openjudge.cn/2024allhw/004/
117	sy296: 后缀表达式	stack	Easy	https://sunnywhy.com/sfbj/7/1/296
116	03704: 括号匹配问题	stack	必须会	http://cs101.openjudge.cn/practice/03704
116	sy295: 可能的出栈序列	stack	Medium	https://sunnywhy.com/sfbj/7/1/295
116	sy294: 合法的出栈序列	stack	Easy	https://sunnywhy.com/sfbj/7/1/294
115	27653:Fraction类	OOP	-	http://cs101.openjudge.cn/practice/27653
115	sy293: 栈的操作序列	stack	Easy	https://sunnywhy.com/sfbj/7/1/293
115	118.杨辉三角	dp	Easy	https://leetcode.cn/problems/pascals-triangle/
114	01321: 棋盘问题	backtracking	-	http://cs101.openjudge.cn/practice/01321/
114	234.回文链表	快慢指针	Easy	https://leetcode.cn/problems/palindrome-linked-list/
114	206.反转链表	linked-list	Easy	https://leetcode.cn/problems/reverse-linked-list/
113	05345: 位查询	implementation	-	http://cs101.openjudge.cn/practice/05345/
113	35.搜索插入位置	binary search	Easy	https://leetcode.cn/problems/search-insert-position/
113	20.有效的括号	stack	Easy	https://leetcode.cn/problems/valid-parentheses/

以及自己写了一个程序用来规范Latex Mathmode Format, [yuanyi-350/Markdown]