Assignment #1: 虚拟机，Shell & 大语言模型

Updated 2309 GMT+8 Feb 20, 2025

2025 spring, Complied by <mark>同学的姓名、院系</mark>

姓名：李彦臻
学号：2300010821
学院：数学科学学院

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/

思路：第一步先通分，把初步的分子 x 和分母 y 算出来；
随后找出 x 的所有正公因数，并验证其是否为 y 的约数；如果是的话，它就是 x 和 y 公共正约数！最后，找出 x 和 y 的最大正约数 t，并将 x 和 y 都除以 t 即可~（注意正负）

代码：

```python
```
```
data = list(map(int,input().split()))
a, b, c, d = data[0], data[1], data[2], data[3]
x, y = a*d + b*c, b*d # 算出分子 x 和分母 y

# 找出 x 的所有正因数
factor_x = []
if int(abs(x)**0.5)**2 == abs(x): # 判断 x 是否是平方数
    status = True
else: status = False
for i in range(1,int(abs(x)**0.5) + 1):
    if x % i == 0:
        factor_x.append(i)
        factor_x.append(abs(x // i)) # 别忘了要用"//", 否则是 float
if status == True:
```

```
    factor_x.remove(abs(x)**0.5)

# 找出 x 和 y 的公因数
factor_both = []
for i in factor_x:
    if y % i == 0:
        factor_both.append(i)
t = max(factor_both) # t 即为 x 和 y 的最大正公因数

# 找出最终的分子和分母
p, q = x // t, y // t
if q > 0:
    print(f"{p}/{q}")
else:
    print(f"{-p}/{-q}")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

1760.袋子里最少数目的球

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

思路：思路类似于奶牛跳方块那道题，定义一个函数 ok(a)来判断一个每一个最小开销 a 是否可能实现，接着再用二分查找来判断每个 1~10^9 内的数是否合题即可~
（查找速度会很快，因为 10^9 小于 2^30 次方，因此至多需要查找 30 次，每次的计算量至多是 10^5，可以把总运算量控制在 10^7 以下）

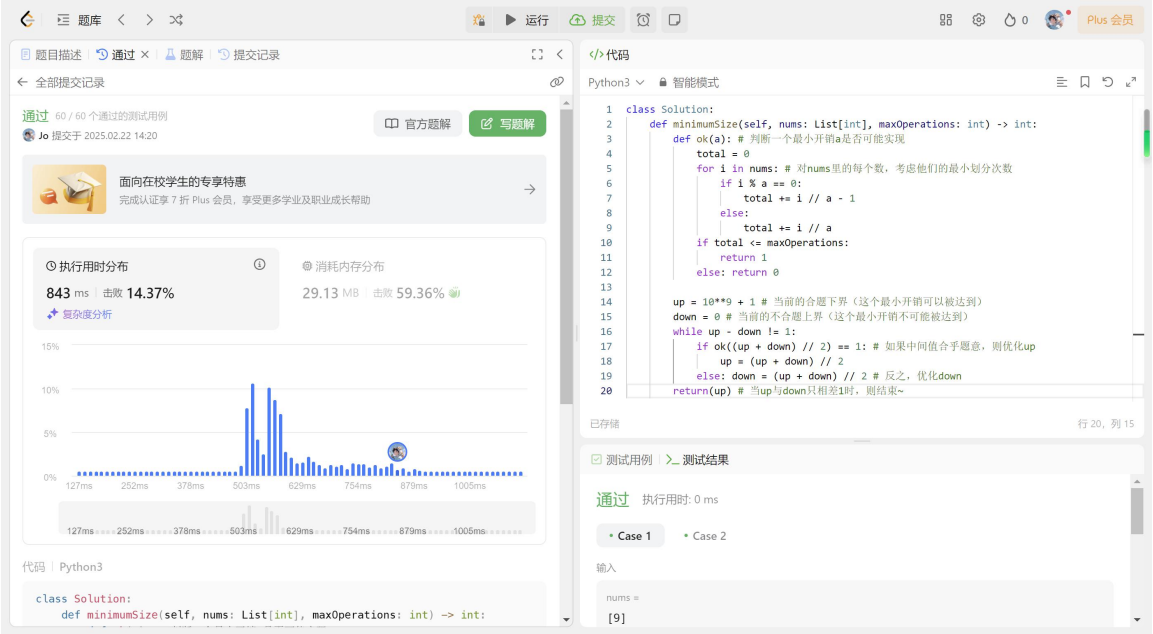
代码：

```python
```

```
class Solution:
    def minimumSize(self, nums: List[int], maxOperations: int) -> int:
        def ok(a): # 判断一个最小开销 a 是否可能实现
            total = 0
            for i in nums: # 对 nums 里的每个数，考虑他们的最小划分次数
                if i % a == 0:
                    total += i // a - 1
                else:
                    total += i // a
            if total <= maxOperations:
                return 1
            else: return 0

        up = 10**9 + 1 # 当前的合题下界（这个最小开销可以被达到）
        down = 0 # 当前的不合题上界（这个最小开销不可能被达到）
        while up - down != 1:
            if ok((up + down) // 2) == 1: # 如果中间值合乎题意，则优化 up
                up = (up + down) // 2
            else: down = (up + down) // 2 # 反之，优化 down
        return(up) # 当 up 与 down 只相差 1 时，则结束~
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



04135: 月度开销

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

思路：方法与奶牛跳格子几乎一模一样，几乎可以说就是翻版奶牛题
本质上就是二分查找，先建立一个函数 ok(a)来判断一个每一个最小开销 a 是否可能实现，接着再用二分查找来判断每个 k~10^9 内的数是否合题即可~
（其中 k 是开销最大的那一天的开销，并且最大月度开销无论如何也得不小于 k，否则 k 这一天无法被容纳进任何一个周期）

代码：

```python

```

```
n, m = map(int,input().split())
data = []
for i in range(n):
    data.append(int(input()))
k = max(data) # 最大月度开销无论如何也得不小于 k，否则 k 这一天无法被容纳进任何一个周期
```

```
def ok(a): # 判断最大月度开销为 a 时有没有可行方案
    total = 0
    t = 1 # 当前的周期数
    i = 0
    while i < n:
        if total + data[i] <= a:
            total += data[i]
        else:
            t += 1 # 新开一个周期
            total = data[i] # 重置 total，并将 data[i]放进下一个周期
        i += 1
    if t > m:
        return 0
    else: return 1

up = 10**9 # 合题的下界（up 及以上的数可以作为最大月度开销）
down = k - 1 # 不合题的上界（down 及以下的数不可能作为最大月度开销）
while up - down != 1:
    if ok((up + down) // 2) == 1: # 如果中间值合乎题意，则优化 up
        up = (up + down) // 2
    else: down = (up + down) // 2 # 反之，优化 down
print(up) # 当 up 与 down 只相差 1 时，则结束~
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

27300: 模型整理

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

思路：第一步，先把所有的模型名字收集起来，并用字典记录下每个模型所对应的所有参数；
第二步，将字典里的 keys（也就是模型名）按照字典序排列好；
第三步，按照规范把字典打印出来，其中在这一步要对每个模型内部的各个大小数据进行排序，按照 M 和 B 分组进行处理即可（注意格式和 float 与 int 需要分开处理的细节！！）
算是一道本质难度不大，但颇为繁琐的题目~

代码：

```python

```
```
def main(data):
    k = data.index("-")
    name = "".join(data[:k])
    num = "".join(data[k + 1:])
    if name not in used:
        used.append(name)
        dic[name] = [num]
    else:
        dic[name].append(num)

n = int(input())
used = [] #已经录入过的模型
dic = {} #模型及其对应的型号
for i in range(n):
    data = list(map(str,input())) #没有空格就不用 split
    main(data)

sorted_dic = dict(sorted(dic.items(), key=lambda item: item[0])) #将模型按照字典序排列

for key in sorted_dic: #最后，按照规范将 dic 打印出来
    finale = [key,":"]
    M_level = [] #Million 量级的型号
    B_level = [] #Billion 量级的型号
```

```
    for value in sorted_dic[key]: #将该模型的型号按照大小排好
        lis = list(map(str,value))
        if lis[-1] == "M":
            if "." in lis[:-1]: #如果是小数形式的型号，用 float
                M_level.append(float("".join(map(str,lis[:-1]))))
            else: #反之，如果是整数形式的型号，用 int
                M_level.append(int("".join(map(str,lis[:-1]))))
        else:
            if "." in lis[:-1]: #如果是小数形式的型号，用 float
                B_level.append(float("".join(map(str,lis[:-1]))))
            else: #反之，如果是整数形式的型号，用 int
                B_level.append(int("".join(map(str,lis[:-1]))))
        M_level.sort()
        B_level.sort()

    for num in M_level: #先把 M 量级的型号全部打印出来
        finale.append(" ")
        finale.append(num)
        finale.append("M")
        finale.append(",")
    for num in B_level: #再把 B 量级的打印出来
        finale.append(" ")
        finale.append(num)
        finale.append("B")
        finale.append(",")

    print("".join(map(str,finale[:-1]))) #不要打印出最后一个逗号
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>


```
def main(data):
    k = data.index("-")
    name = "".join(data[:k])
    num = "".join(data[k + 1:])
    if name not in used:
        used.append(name)
        dic[name] = [num]
    else:
        dic[name].append(num)

n = int(input())
used = [] #已经录入过的模型
dic = {} #模型及其对应的型号
for i in range(n):
    data = list(map(str,input())) #没有空格就不用split
    main(data)

sorted_dic = dict(sorted(dic.items(), key=lambda item: item[0])) #将模型

for key in sorted_dic: #最后，按照规范将dic打印出来
    finale = [key,":"]
    M_level = [] #Million量级的型号
    B_level = [] #Billion量级的型号

    for value in sorted_dic[key]: #将该模型的型号按照大小排好
        lis = list(map(str,value))
        if lis[-1] == "M":
```

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

<mark>如果发现作业题目相对简单，有否寻找额外的练习题目，如“数算 2025spring 每日选做”、LeetCode、Codeforces、洛谷等网站上的题目。</mark>

先回顾一下这次作业的题目：

对于第一题而言，我认为是一道很常规的题目。本质上就是计算分数的加法，然后把结果约分一下就可以了。因此，这道题可以自然而然拆分为两步，第一步先是把他们俩相加起来，方法就是直接通分即可；第二步是对这个分数进行约分，这一步中实际上我们只需要找出上面那个数与下面的数的最大正公约数即可。要做到这一点，只需要把其中一个数的约数全部找出来，然后挨个验证这些约数是不是另外一个数的约数，就能把这两个数的所有公因数全都找出来了；这样一来，也就自然而然能找出他们俩的最大正公因数了。最后，再把他们俩分别除以他们两的最大公约数，再调整一下正负号，就能够做完这道题了。

第二题和第三题的思路很像，甚至题目里的记号都很相似，都是计算某个东西的“最大开销的最小值”。对题目本身而言，他们的本质思路就是二分查找，很常规。如果大家之前做过一道名叫奶牛跳格子的问题的话，那么其实这道题的思路是颇为好想到的，因为只需要把奶牛跳格子那道题的思路原封不动的用到这两题上去就可以了。具体的答题模版在上方已经有所阐述了，就不再赘述了（无非就是建立一个“判断函数”+设置上下限进行二分查找即可）。

第四题的话就是属于那种思维难度很简单，但是操作起来却颇为繁琐的一道题。它的思路和方法都是很好想到的，但是中间的调试过程却非常的麻烦，具体的思路在上方已经写过了，就不赘述了。中间主要需要注意、容易犯错的几个地方是：
第一，需要对模型的名字进行字典排序；为了进行这一步操作，只能套用以前的那个对字典或者元组里某个位置的值进行排序的基本模版（也就是那一长串“dict(sorted(dic.items(), key=lambda item: item[0]))”）才能够进行排序。
第二个需要小心的地方，就是要对每个模型内一部的参数大小进行排序。
这里需要注意两个问题：
第一个是需要把所有的参数大小先提前分为两个小组，第一个是 M 代表的那个小组，第二个是 B 代表的那个小组，对他们两类大小要分开进行排序才好操作一点；
第二就是在比较 M 或者 B 内部的参数大小的时候，会遇到需要把浮点数与整数进行比较的问题，而这个过程中很容易在输出结果的时候把整数也变为浮点数，也就是整数后面多带了一个零。而解决办法就是，在最后输出格式的时候，要挨个儿检查输出的每个数是不是整数，如果是的话，就把后面那个“点 0”给去掉就可以了。

总的来讲，我觉得这次作业的难度并不算大，题目的思路相对来讲都比较常规，但是因为我很久都没有敲代码了，所以对很多语法和知识上的细节都有点忘记了，因此我还是花了不少时间回忆、翻以前的笔记来复习，因此这次的作业也算是一次复健运动了哈哈哈；希望自己能在接下来进一步的刷题训练中，逐渐找回感觉，回到最佳状态！