Assignment #1: 虚拟机，Shell & 大模型

Updated 1317 GMT+8 Feb 20, 2025

2025 spring, Complied by <mark>汤伟杰，信息管理系</mark>

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

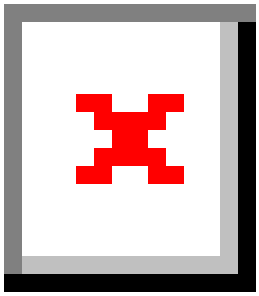
http://cs101.openjudge.cn/practice/27653/

思路： 初始化提供分子和分母，这里直接使用了 math.gcd 来约分，并且要重构两个函数，一个是加号，自定义为分数的加法；一个是_ _ str _ _，由于每次运行 print()时 python 都会默认使用 _ _ str _ _，所以也要进行输出格式的重写，或者定义一个展示函数调用输出。

代码：

```
from math import gcd
class Fraction:
    def __init__(self,top,bot):
        d=gcd(top,bot)
        self.top=top//d
        self.bottom=bot//d

    def __add__(self,b):
        new_top=self.top*b.bottom+self.bottom*b.top
        new_bottom=self.bottom*b.bottom
        return Fraction(new_top,new_bottom)

    def __str__(self):
        return str(self.top)+'/'+str(self.bottom)

    # def show(self):
    #     print(str(self.top)+'/'+str(self.bottom))

t1,b1,t2,b2=map(int,input().split())
a,b=Fraction(t1,b1),Fraction(t2,b2)
print(a+b)
# ans=a+b
# ans.show()
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



1760.袋子里最少数目的球

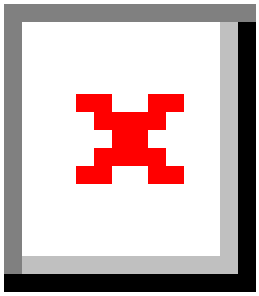
https://leetcode.cn/problems/minimum-limit-of-balls-in-a-bag/

思路： 与 月度开销 一样的思路，check 函数借鉴了题解（因为没想到怎么计算）mid 定义为“每个盒子最多放 mid 个球”，目的是让 mid 尽可能小。对于一个盒子的 n 个球，要达到 mid 需要变成 ceil(n/mid)个，也就是操作 ceil(n/mid)-1 次。

代码：

```
class Solution:
    def minimumSize(self, nums: List[int], maxOperations: int) -> int:
        def check(mid):
            cnt=0
            for n in nums:
                cnt+=math.ceil(n/mid)-1
            return cnt<=maxOperations

        left=1
        right=max(nums)
        while left<=right:
            mid=(left+right)//2
            if not check(mid):
                # 当前的 mid 不符合，说明每个盒子的球太少了，要增加
                left=mid+1
            else:
                right=mid-1
        return left
```

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

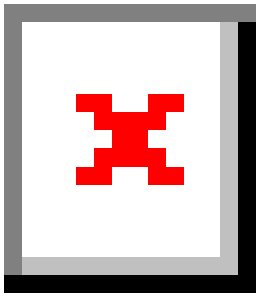
思路： 二分，最关键的地方是初始化 left 和 right，由于月份金额会累加，所以最小的最大金额肯定是数组的最大值（left=max(s)），而最大的最大金额肯定是把全部看成一个月，即 right=sum(s)。然后进行二分，mid 定义为“可能的最小的最大金额”，每次检查当前金额是否符合，如果符合，尝试减小 mid，看看能不能更小。其中 check 函数的 for 循环中先写 if 判断来更新比较简洁易懂。

代码：

```
def check(mid,s,m):
    curr=0
    cnt=1
    for i in s:
        if curr+i>mid:
            cnt+=1
            curr=i
        else:
            curr+=i
    return cnt<=m

def solve():
    n,m=map(int,input().split())
    s=[]
    for _ in range(n):
        s.append(int(input()))
    left=max(s)
    right=sum(s)
    while left<=right:
        mid=(left+right)//2
        if not check(mid,s,m):
            left=mid+1
        else:
```

```
            right=mid-1
    return left

print(solve())
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



27300: 模型整理

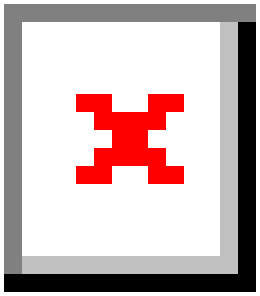
http://cs101.openjudge.cn/practice/27300/

思路： 使用 defaultdict 来获取每种模型对应的参数，然后全部存到列表里面，先对模型名称进行一次排序，然后在输出每个模型的各个参数时，使用 lambda 函数计算带单位的数值大小来进行第二次排序，再输出。

代码：

```
map={'M':10**6, 'B':10**9}
from collections import defaultdict
n=int(input())
d=defaultdict(list)
for _ in range(n):
    name,num=input().split('-')
    d[name].append(num)

s=[]
for k,v in d.items():
    s.append((k,v))
# print(s)
s.sort(key=lambda x:x[0])
for k,v in s:
    print(k+': ',end='')
    v.sort(key=lambda x:float(x[:-1])*map[x[-1]])
    print(', '.join(v))
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>



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。

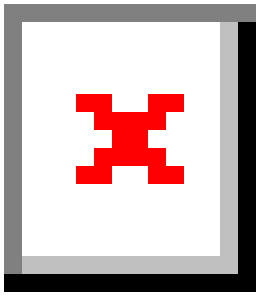
下载了 LM studio，也把这个推荐给了室友和高中同学，我的电脑只能跑 14b 的，每次都要推理很长时间。感觉 shell 的命令行的命令还需要进一步学习熟练熟练。

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

作者：Sebastian Raschka

花了时间读了一下英文原著，用时很长，第一次啃英文原文书，但是确实对 LLM 有了一定的认识。

①文中是以 GPT3 作为示例来展开 LLM 的讲解的，这也是我第一次知道“GPT”的全称，原来是 Generative Pre-Training。我上学期的一门专业课的期末考试出现过这道题：写出 ChatGPT 的英文全称。。



②与第一条差不多，文中提到了 Attention is all you need，这句广为流传的（甚至带有戏谑性的？）话居然是 2017 年提出 Tranformer 架构的论文题目！我也去专门找了一下阅读前面的一小部分，对于这本书中关于 encoder-decoder 部分有更为深入的阐述和描述。

③有人称 LLM 为“大预言模型”看来不是假的（哈哈哈），原来确实是通过对数据的预处理和预训练，来使得模型具有 预测文本将会出现的下一个单词 的能力，这是 Pretraining 的结果。接下来进行 Fine-tune，来使得大模型具备某一领域的更高能力。

④GPT 采用的架构是不包含 encoder 的，仅仅是包含 decoder，而它最原始的设计目的也仅仅是 “next-word prediciton”（查看 Attention is all you need 的摘要部分提到了 GPT 在翻译中的极其优秀的能力），然而它却也可以进行 translation tasks （原文这里说 This capability was initially unexpected to researchers，连研究员都没想到能这么强），作者给出的解释是，模型暴露在大量语料的环境下，自然而然地具备了这种能力。有点细思极恐..?

⑤关于 gpt 的预测下一个词语出现的方式，一是结合用户 input 的不完整句子，根据已有的单词进行推测；二是将已经推测出的单词也转换成新的 input，再继续进行预测。

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>

每日选做还在跟进！刚开始的题目有好多都是计概的题目，不过有些也很有挑战性，比如力扣的 2906.构造乘积矩阵，前缀和的运用真的很神奇（这种题目一看我就知道会超时但是又想不到什么好的写法，只能看题解），还有后面的力扣 2502.设计内存分配器，虽然是 implementation 但是找思路感觉对我来说有点困难，还要多多练习！