Assignment #1: 虚拟机，Shell & 大语言模型

Updated 1730 GMT+8 Feb 20, 2025

2025 spring, Complied by 任宇桐 物理学院

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 没有提供源码：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 没有提供截图：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：

引入math，使用gcd最小公倍数进行模拟十字相乘和约分即可。

代码：

```
import math

class Fraction:
    def __init__(self, top, bottom):
        self.num = top
        self.den = bottom
    def __add__(self, another):
        orig_num = self.num*another.den+another.num*self.den
        orig_den = self.den*another.den
        common = math.gcd(orig_num, orig_den)
        top = orig_num // common
        bottom = orig_den // common
        return f'{top}/{bottom}'

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

代码运行截图 （至少包含有"Accepted"）

状态：Accepted

源代码

```
import math

class Fraction:
    def __init__(self, top, bottom):
        self.num = top
        self.den = bottom
    def __add__(self, another):
        orig_num = self.num*another.den+another.num*self.den
        orig_den = self.den*another.den
        common = math.gcd(orig_num, orig_den)
        top = orig_num // common
        bottom = orig_den // common
        return f'{top}/{bottom}'

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

基本信息

#：	48105490
题目：	27653
提交人：	24n2400011498
内存：	3624kB
时间：	26ms
语言：	Python3
提交时间：	2025-01-14 10:37:47

1760.袋子里最少数目的球

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

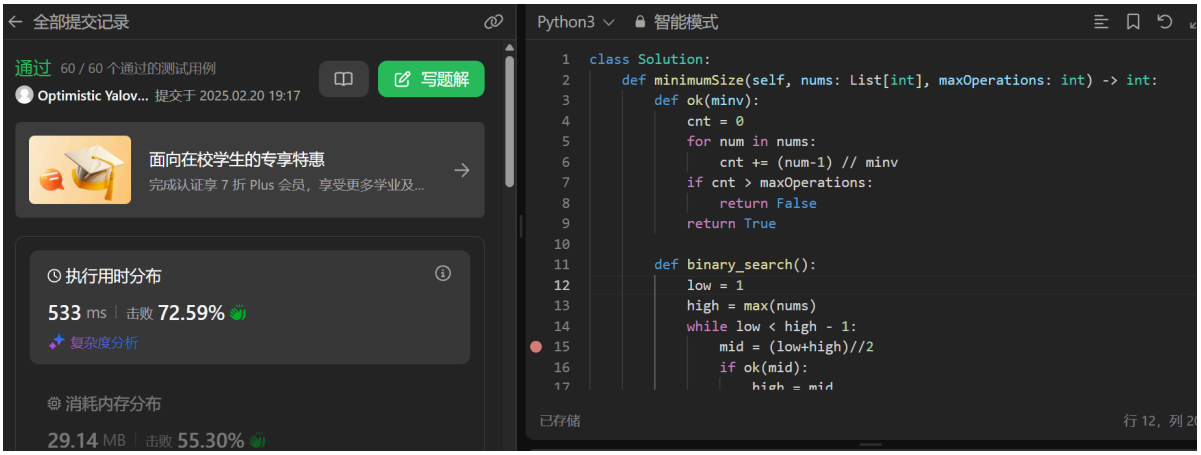
思路：

单调搜索问题，首先需要想到二分查找优化枚举的效率。同时需要处理好小细节，如判断需要分几次，以及最值的边界特判。

代码：

```
class Solution:
    def minimumSize(self, nums: List[int], maxOperations: int) -> int:
        def ok(minv):
            cnt = 0
            for num in nums:
                cnt += (num-1) // minv
            if cnt > maxOperations:
                return False
            return True

        def binary_search():
            low = 1
            high = max(nums)
            while low < high - 1:
                mid = (low+high)//2
                if ok(mid):
                    high = mid
                else:
                    low = mid
            if ok(low):
                return low
            return high

        return binary_search()
```

代码运行截图 （至少包含有"Accepted"）



04135: 月度开销

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

思路：

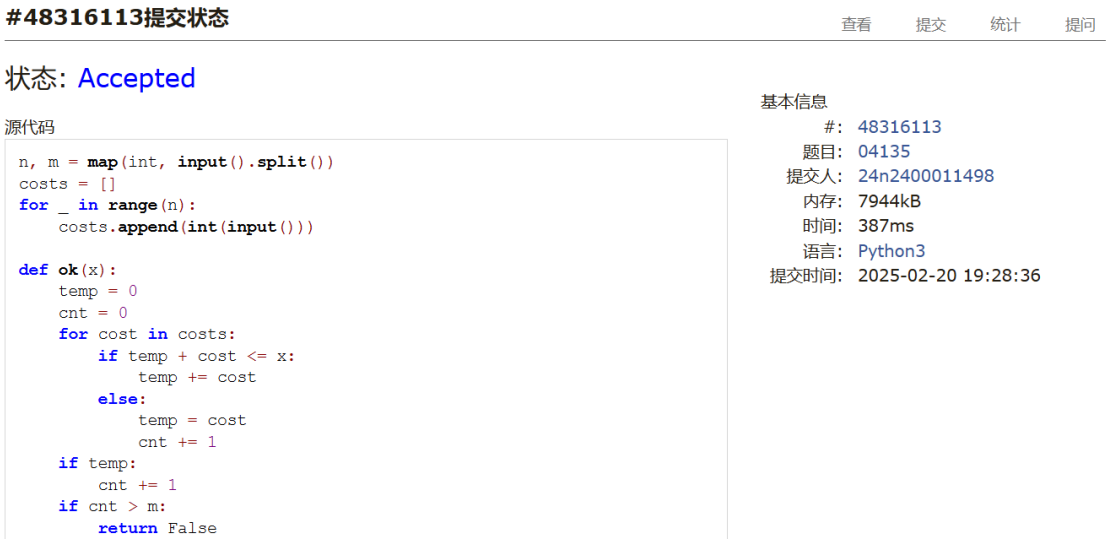
容易注意到需要使用二分查找得到答案，同样需要注意判断计数条件。

代码：

```
n, m = map(int, input().split())
costs = []
for _ in range(n):
    costs.append(int(input()))

def ok(x):
    temp = 0
    cnt = 0
    for cost in costs:
        if temp + cost <= x:
            temp += cost
        else:
            temp = cost
            cnt += 1
    if temp:
        cnt += 1
    if cnt > m:
        return False
    else:
        return True

def binary_search():
    low = max(costs)
    high = sum(costs)
    while low < high -1:
        mid = (low+high)//2
        if ok(mid):
            high = mid
```

```
        else:
            low = mid
    if ok(low):
        return low
    else:
        return high

print(binary_search())
```

代码运行截图 （至少包含有"Accepted"）



#48316113提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
n, m = map(int, input().split())
costs = []
for _ in range(n):
    costs.append(int(input()))

def ok(x):
    temp = 0
    cnt = 0
    for cost in costs:
        if temp + cost <= x:
            temp += cost
        else:
            temp = cost
            cnt += 1
    if temp:
        cnt += 1
    if cnt > m:
        return False
```

基本信息

#: 48316113
题目: 04135
提交人: 24n2400011498
内存: 7944kB
时间: 387ms
语言: Python3
提交时间: 2025-02-20 19:28:36

27300: 模型整理

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

思路:

思路比较直接，注意多条件排序即可。

代码:

```
n = int(input())
pairs = {}
for _ in range(n):
    s = input().split('-')
    if s[0] in pairs:
        pairs[s[0]].append(s[1])
    else:
        pairs[s[0]] = [s[1]]
for key in pairs:
    pairs[key].sort(key=lambda x: (-ord(x[-1]), float(x[:-1])))
for key in sorted(pairs.keys()):
    print(f"{key}: {', '.join(pairs[key])}")
```

代码运行截图（至少包含有"Accepted"）

#46265526提交状态

查看 提交 统计 提问

状态：Accepted

源代码

```
n = int(input())
pairs = {}
for _ in range(n):
    s = input().split('-')
    if s[0] in pairs:
        pairs[s[0]].append(s[1])
    else:
        pairs[s[0]] = [s[1]]
for key in pairs:
    pairs[key].sort(key=lambda x: (-ord(x[-1]), float(x[:-1])))
for key in sorted(pairs.keys()):
    print(f"{key}: {', '.join(pairs[key])}")
```

基本信息

#: 46265526
题目: 27300
提交人: 24n2400011498
内存: 3612kB
时间: 24ms
语言: Python3
提交时间: 2024-09-29 11:31:49

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。

尝试在本地运行deepseek7B模型，能解决不少简单问题，但是笔记本电脑似乎GPU不大行，运行速度非常慢。尝试了LM Studio+AnythingLLM构建知识库，小规模测试了一两个问题，感觉有点作用。

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。

https://github.com/stur007/2025spring_DataStructure_and_Algorithm_B/blob/main/LLM%20Notes.md

笔记第一章的内容附后。

LLM Notes

Chapter 1 Basic Concepts

Difference between LLM and traditional models

- NLP (natural language model)
- the wider range of application

the position of LLM

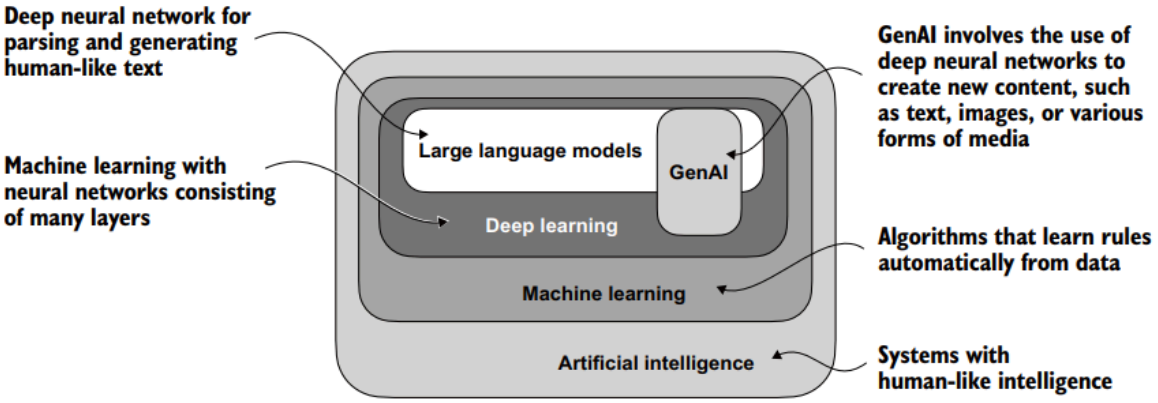


Figure 1.1 As this hierarchical depiction of the relationship between the different fields suggests, LLMs represent a specific application of deep learning techniques, using their ability to process and generate human-like text. Deep learning is a specialized branch of machine learning that focuses on using multilayer neural networks. Machine learning and deep learning are fields aimed at implementing algorithms that enable computers to learn from data and perform tasks that typically require human intelligence.

the most important feature of deep learning: without bring explicitly programmed and manual feature extraction

custom-built LLMs: performed better and more secure for company etc.

how to build custom-built LLMs

pretraining -> finetuning

instruction finetuning: input & output

classification finetuning: Yes & No

transformer

1. composition

   encoder: input text -> (numeric) **embedding** vectors

   decoder: numeric vectors -> output text
2. self-attention mechanism: 'combine' the relevant text

the tasks that LLMs can do

- filling the blanks (BERT) complete the sentences (GPT)
- zero-shot task (a completely new task)
- few-shot task (a few examples ahead)

token: the unit for a model to read

autoregressive model: incorporate previous outputs as inputs for predictions

the procedures of LLM construction

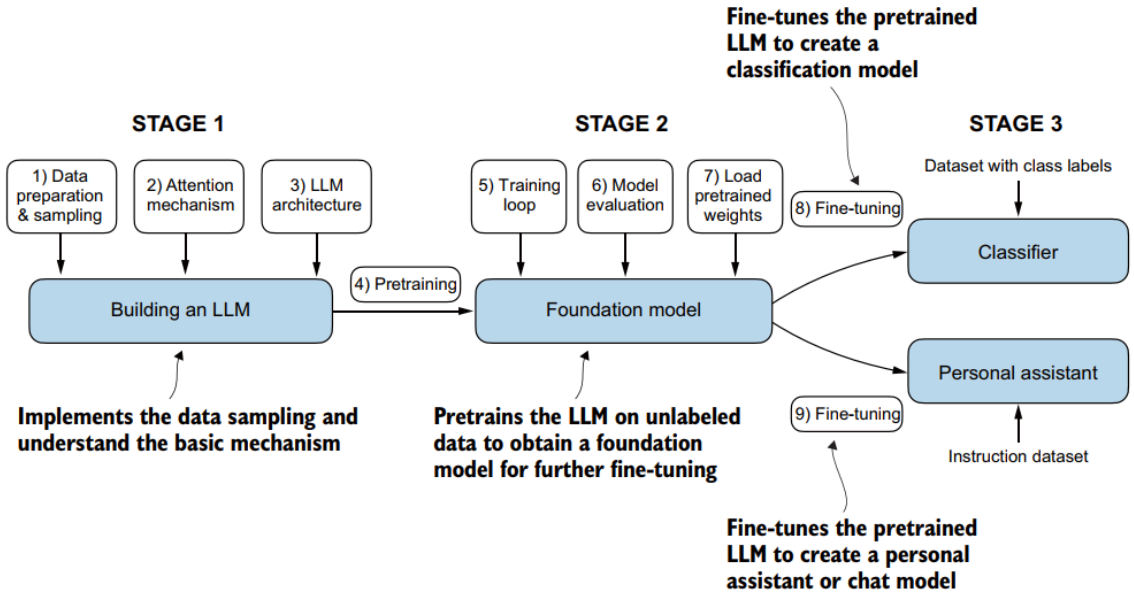


Figure 1.9 The three main stages of coding an LLM are implementing the LLM architecture and data preparation process (stage 1), pretraining an LLM to create a foundation model (stage 2), and fine-tuning the foundation model to become a personal assistant or text classifier (stage 3).

2. 学习总结和个人收获

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。

通过作业复习了二分查找的方案处理一维单调问题，也通过阅读书籍对LLM的基本概念有了一定的了解，但是本地部署测试题目还没有什么头绪，还没有正式开始动工，希望最近有实践可以试试。