



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2025 年春季学期
计算学部《软件工程》课程

Lab 1 实验报告

姓名	班级/学号	联系方式
王禹昕	2203201/ 2022112760	17287006839

目 录

1	实验要求.....	1
2	待求解问题描述.....	1
3	算法与数据结构设计.....	1
3.1	设计思路与算法流程图.....	1
3.2	数据结构设计.....	2
3.3	算法时间复杂度分析.....	3
3.4	算法代码的生成.....	4
4	实验与测试.....	4
4.1	读取文本文件并展示有向图.....	5
4.2	查询桥接词.....	6
4.3	根据桥接词生成新文本.....	6
4.4	计算最短路径.....	7
4.5	计算 PageRank 值.....	8
4.6	随机游走.....	8
5	编程语言与开发环境.....	9
6	Git 操作过程	9
6.1	实验场景(1): 仓库创建与提交	9
6.2	实验场景(2): 分支管理	11
7	在 IDE 中使用 Git Plugin	13
8	小结.....	14

[文档全部完成之后，请在上述区域点击右键，选择“更新域”，在打开的对话框中选择“更新整个目录”]

1 实验要求

实验内容 1: 基于大模型的编程

- 熟悉面向对象的编程;
- 掌握利用大模型辅助编程的方式;
- 实验同大模型的“结对编程”。

实验内容 2: Git 实战

- 熟练掌握 Git 的基本指令和分支管理指令;
- 掌握 Git 支持软件配置管理的核心机理;
- 在实践项目中使用 Git /Github 管理自己的项目源代码。

2 待解决问题描述

试验要求从文本文件中读取英文文本数据,生成有向图并进行多种操作。具体描述如下:

- 输入数据: 一个包含英文文本的文件,文本中的换行符、标点符号均视为空格,非字母字符被忽略。
- 输出数据: 根据输入文本生成的有向图,以及在图上进行操作后的结果,如桥接词查询结果、新文本生成结果、最短路径计算结果、PageRank 值计算结果和随机游走结果等。为方便操作和用户友好,使用图形化操作界面。
- 约束条件: PR 计算(d 取值 0.85),其中出度为 0 的节点需要将 PR 值均分给其他节点。

3 算法与数据结构设计

3.1 设计思路与算法流程图

程序整体流程如下:

用户选择或输入文本文件路径。

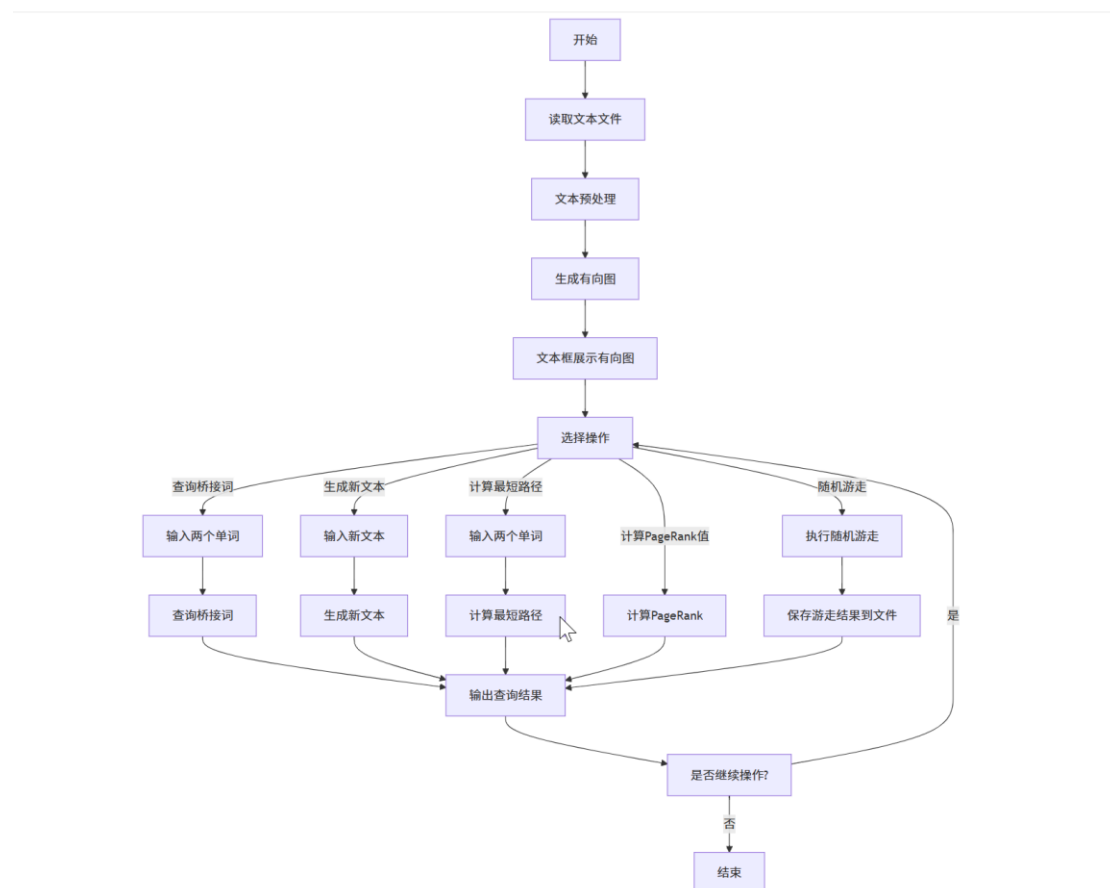
程序读取文本文件,预处理文本(去除标点、转小写等)并生成有向图。

通过图形化界面或命令行展示有向图。

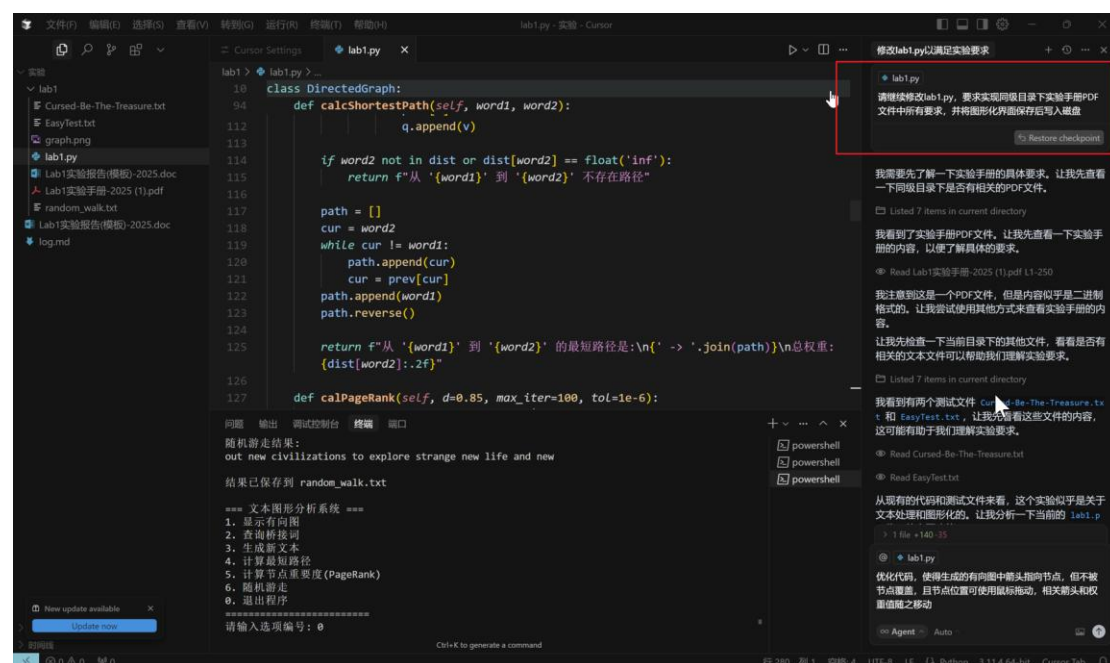
提供多种功能操作供用户选择,如查询桥接词、生成新文本、计算最短路径、计算 PageRank 值和随机游走等。

输出相应操作结果。

流程图:



采用大模型辅助，直接输入实验指导文档，使 Cursor 直接按照文档内要求完成代码，之后不断调整优化。如下图所示：



3.2 数据结构设计

有向图数据结构：

使用**嵌套字典**（defaultdict）存储有向图，支持多重边计数。具体定义如下：

graph: 外层字典，键为节点（单词），值为另一个字典。

内层字典：键为从当前节点指向的节点，值为边的权重（相邻出现次数）。

例如，对于文本 "To explore strange new worlds, To seek out new life and new civilizations", 生成的有向图数据结构为：

```
{
  'to': {'explore': 1, 'seek': 1},
  'explore': {'strange': 1},
  'strange': {'new': 1},
  'new': {'worlds': 1, 'life': 1, 'and': 1},
  'worlds': {},
  'seek': {'out': 1},
  'out': {'new': 1},
  'life': {'and': 1},
  'and': {'new': 1},
  'civilizations': {}
}
```

其他数据结构：

使用 **deque** 实现广度优先搜索（BFS）以计算最短路径。

使用 **dict** 存储 PageRank 值，键为节点，值为对应的 PR 值。

3.3 算法时间复杂度分析

文本预处理：时间复杂度为 $O(n)$ ，其中 n 为文本中字符数量。需遍历整个文本进行字符检查和处理。

有向图生成：时间复杂度为 $O(m)$ ，其中 m 为文本中单词数量。需遍历处理后的单词列表，建立单词之间的边关系。

桥接词查询：时间复杂度为 $O(k)$ ，其中 k 为图中边的数量。需遍历指定节点的出边，查找满足条件的桥接词。

新文本生成：时间复杂度为 $O(l)$ ，其中 l 为输入文本的单词数量。需遍历输入文本单词，查询桥接词并生成新文本。

最短路径计算（BFS）：时间复杂度为 $O(n + m)$ ，其中 n 为节点数， m 为边数。需遍历所有节点和边进行 BFS 搜索功能。

PageRank 计算：时间复杂度为 $O(n \times \text{iter})$ ，其中 iter 为迭代次数。需多次遍历所有节点进行 PR 值更新。

随机游走：时间复杂度为 $O(w)$ ，其中 w 为游走路径长度。需根据游走规则逐步生成路径。

3.4 算法代码的生成

要求在显示有向图图形化界面的同时通过自定义的格式在 CLI（命令行界面）上进行展示生成的有向图，要求格式清晰，易于理解。

如：A→B A→C

B→D

```
=====
请输入选项编号：1
正在生成并显示有向图...

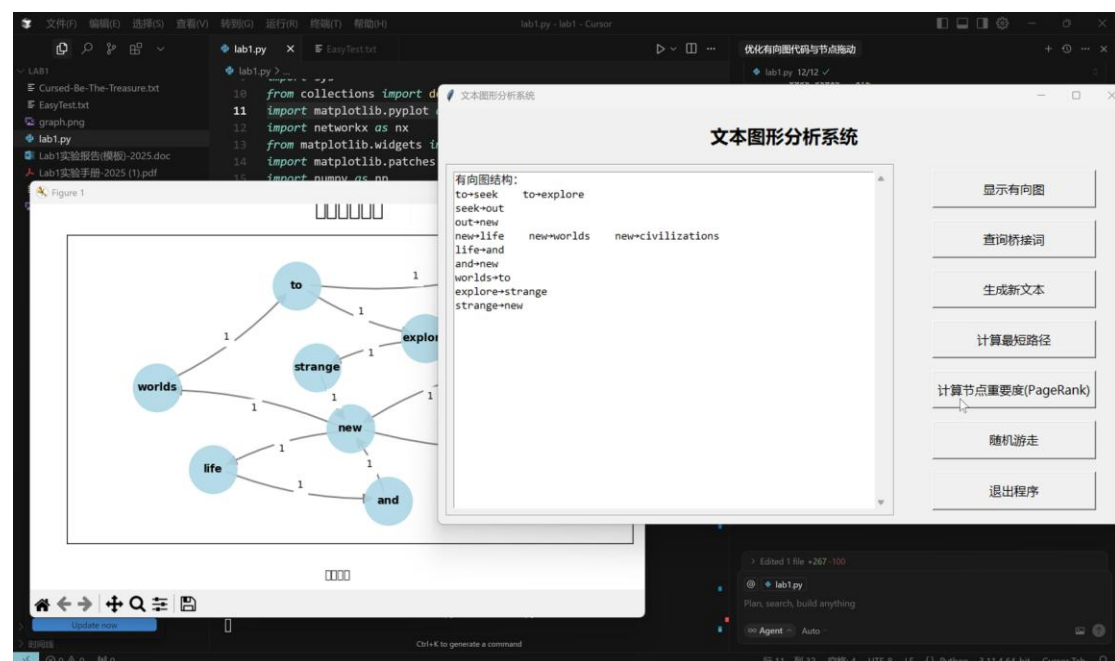
有向图结构（CLI展示）：
to→seek    to→explore
seek→out
out→new
new→life    new→worlds    new→civilizations
life→and
and→new
worlds→to
explore→strange
strange→new
civilizations→to
```

要求将整个命令行操作界面变成图形化操作界面，一个按钮代表一个功能，与命令行界面对应功能不变。

优化图形化操作界面布局和有向图图形展示界面布局。

将按钮放在右侧（图形化操作界面），大小自适应界面。

可使用鼠标滚轮放大或缩小有向图。



4 实验与测试

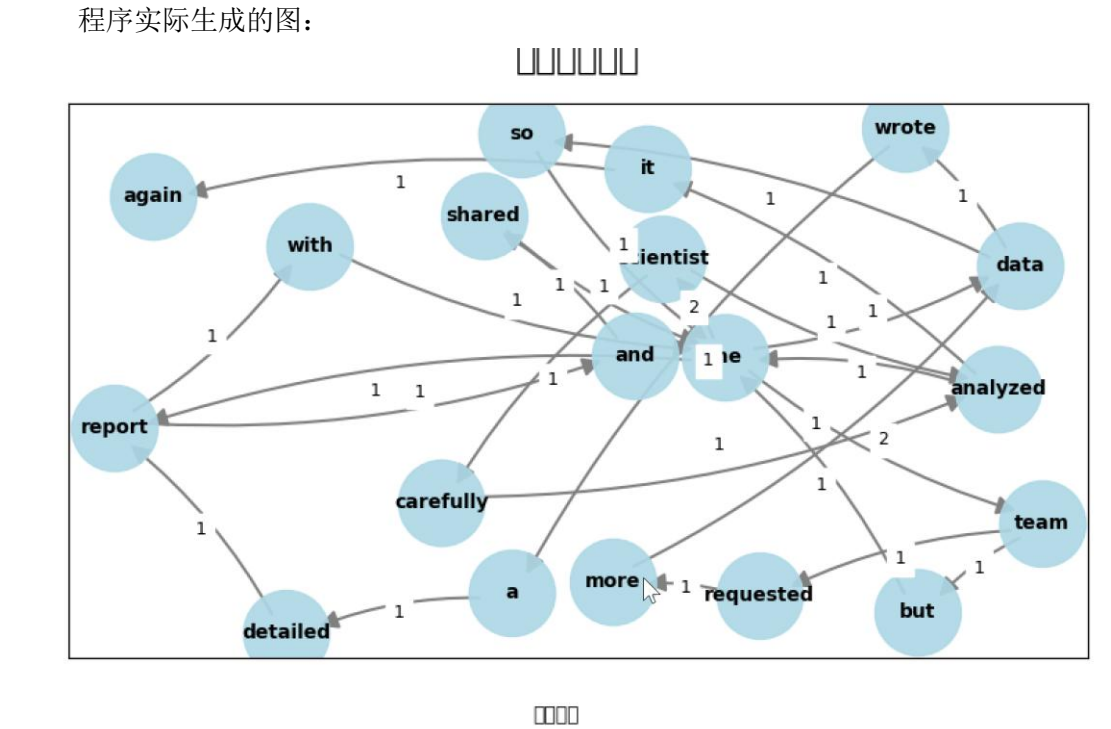
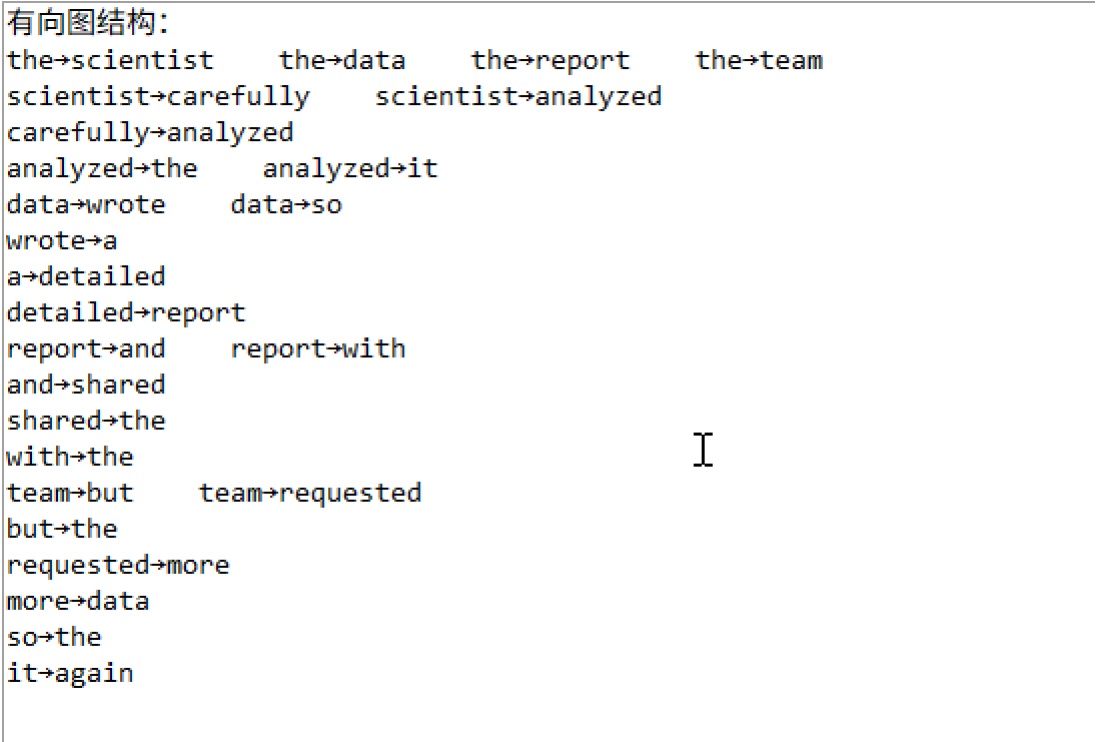
利用提供的 Easy Test.txt 文件进行测试。

4.1 读取文本文件并展示有向图

文本文件中包含的内容：

The scientist carefully analyzed the data, wrote a detailed report, and shared the report with the team, but the team requested more data, so the scientist analyzed it again.

期望生成的图（手工计算得到）：



4.2 查询桥接词

序号	输入（2 个单词）	期望输出	实际输出	运行是否正确
1	scientist, analyzed	从 'scientist' 到 'analyzed' 的桥接词是: carefully	从 'scientist' 到 'analyzed' 的桥接词是: carefully	是
2	analyzed, data	从 'analyzed' 到 'data' 的桥接词是: the	从 'analyzed' 到 'data' 的桥接词是: the	是
3	team, requested	在 'team' 和 'requested' 之间没有桥接词	在 'team' 和 'requested' 之间没有桥接词	是

给出实际运行得到结果的界面截图。



4.3 根据桥接词生成新文本

序号	输入（一行文本）	期望输出	实际输出	运行是否正确
1	The scientist wrote a report	the scientist wrote a detailed report	the scientist wrote a detailed report	是
2	team requested more data	team requested more data	team requested more data	是
3	scientist analyzed it	scientist carefully analyzed it	scientist carefully analyzed it	是

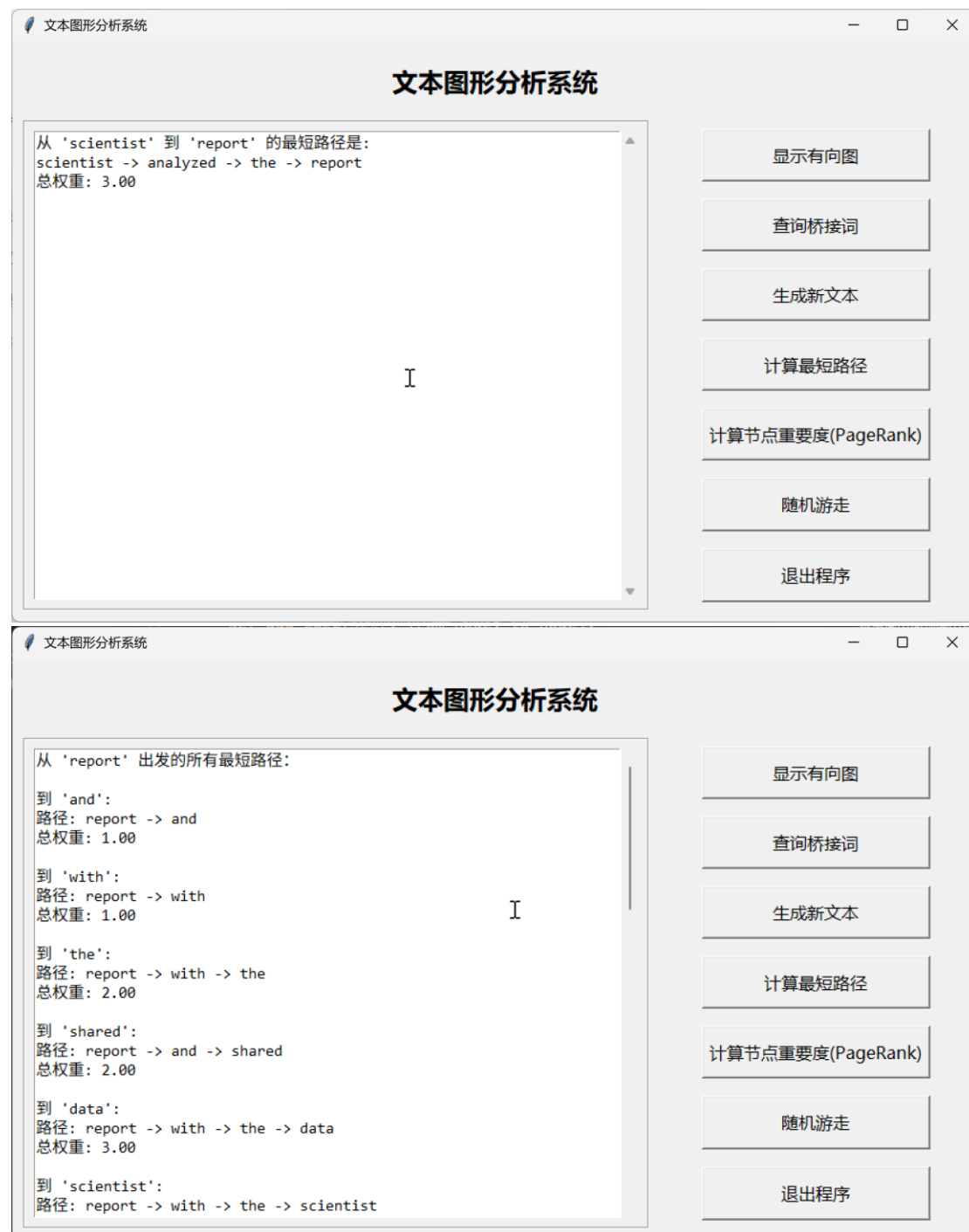
给出实际运行得到结果的界面截图。



4.4 计算最短路径

序号	输入（两个单词、 或一个单词）	期望输出	实际输出	运行是否正确
1	scientist, report	从 'scientist' 到 'report' 的最短路径是: scientist -> analyzed -> the -> report 总权重: 3.00	从 'scientist' 到 'report' 的最短路径是: scientist -> analyzed -> the -> report 总权重: 3.00	是
2	team, analyzed	从 'team' 到 'analyzed' 的最短路径是: team -> but -> the -> scientist -> analyzed 总权重: 5.00	从 'team' 到 'analyzed' 的最短路径是: team -> but -> the -> scientist -> analyzed 总权重: 5.00	是
3	report	从 'report' 出发的所有最短路径: 到 'and': 路径: report -> and 总权重: 1.00 到 'with': 路径: report -> with 总权重: 1.00 ……略	从 'report' 出发的所有最短路径: 到 'and': 路径: report -> and 总权重: 1.00 到 'with': 路径: report -> with 总权重: 1.00 ……略	是

给出实际运行得到结果的界面截图。



4.5 计算 PageRank 值

序号	单词	期望输出	实际输出	运行是否正确
1	the	0.1707	0.1707	是
2	report	0.0727	0.0727	是
3	data	0.0670	0.0670	是

4.6 随机游走

该功能无输入，让你的程序执行多次，分别记录结果。

序号	实际输出	程序运行是否正确
1	team requested more data so the team	是
2	scientist carefully analyzed the data so the report and shared the	是
3	the data so the	是

给出实际运行得到结果的界面截图。



5 编程语言与开发环境

Python3 版本、IDE（cursor 或者 vscode）；
采用的大模型：cursor。

6 Git 操作过程

6.1 实验场景(1): 仓库创建与提交

R0: 查看工作区、暂存区、git 仓库的状态

```
git status
```

R1: 本地初始化一个 git 仓库

```
git init
```

R2: 将项目文件加入 git 管理并提交

```
git add .
```

```
git commit -m "Initial commit"
```

R3: 修改文件后查看状态和修改内容

使用文本编辑器修改文件

```
git status
```

git diff

R4: 重新提交修改

git add .

git commit -m "Updated files"

R5: 再次修改文件并提交

使用文本编辑器再次修改文件

git add .

git commit -m "Another update"

R6: 撤销最后一次提交

git reset --soft HEAD^

R7: 查看提交记录

git log

R8: 创建远程仓库并关联本地仓库

在 GitHub 上创建仓库后，复制仓库 URL

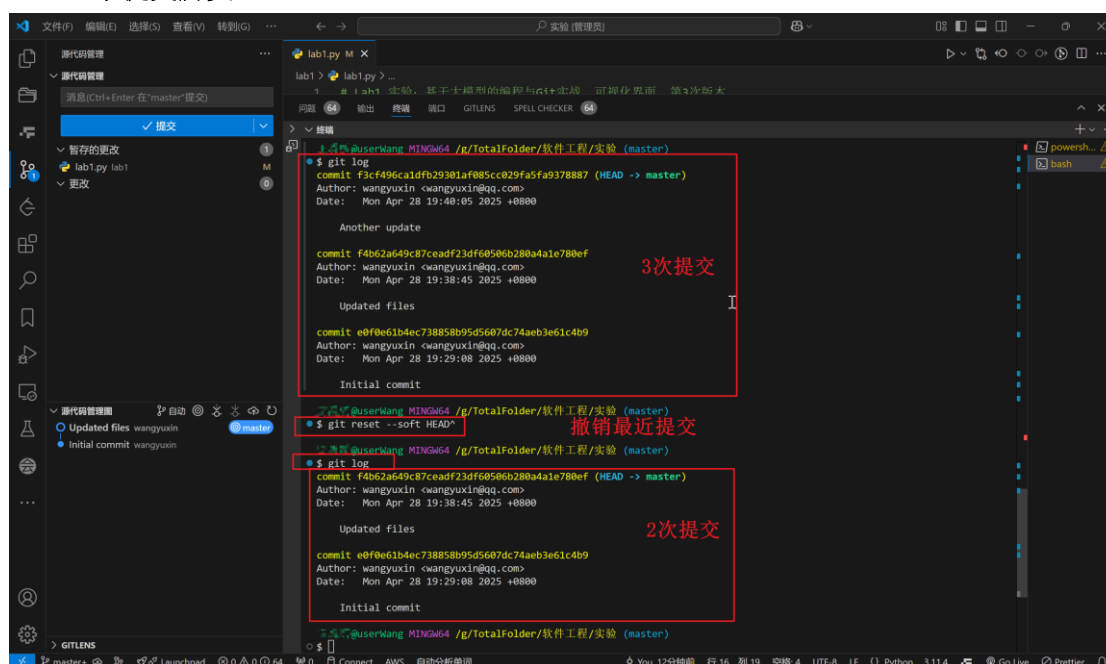
git remote add origin <_url>

R9: 推送本地仓库到 GitHub

git push -u origin master

图示如下:

显示提交历史:



```
lab1py M X
lab1 > git log
commit f3cf496caldfb29301af085cc029fa5fa9378887 (HEAD -> master)
Author: wangyuxin <wangyuxin@qq.com>
Date: Mon Apr 28 19:48:05 2025 +0800

    Another update

commit f4b62a649c87cead723df6058b280a41e780ef
Author: wangyuxin <wangyuxin@qq.com>
Date: Mon Apr 28 19:38:45 2025 +0800

    Updated files

commit e0f0e61b4ec738858b95d5607dc74aeb3e61c4b9
Author: wangyuxin <wangyuxin@qq.com>
Date: Mon Apr 28 19:29:08 2025 +0800

    Initial commit

$ git reset --soft HEAD^
$ git log
commit f4b62a649c87cead723df6058b280a41e780ef (HEAD -> master)
Author: wangyuxin <wangyuxin@qq.com>
Date: Mon Apr 28 19:38:45 2025 +0800

    Updated files

commit e0f0e61b4ec738858b95d5607dc74aeb3e61c4b9
Author: wangyuxin <wangyuxin@qq.com>
Date: Mon Apr 28 19:29:08 2025 +0800

    Initial commit
```

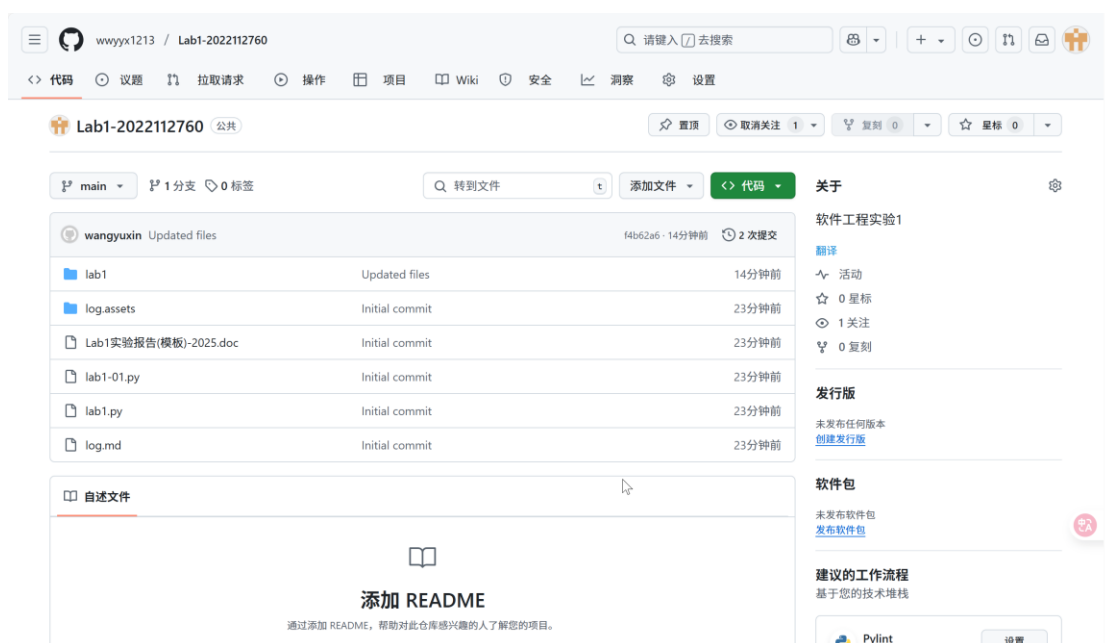
创建分支并将代码 push 到 GitHub:

```
@userWang MINGW64 /g/TotalFolder/软件工程/实验 (master)
$ git remote add origin https://github.com/wwyyx1213/Lab1-2022112760.git

@guserWang MINGW64 /g/TotalFolder/软件工程/实验 (master)
$ git branch -M main

@guserWang MINGW64 /g/TotalFolder/软件工程/实验 (main)
$ git push -u origin main
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 20 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (27/27), 3.59 MiB | 1.30 MiB/s, done.
Total 27 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/wwyyx1213/Lab1-2022112760.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

GitHub 界面:



6.2 实验场景(2): 分支管理

R1: 查看和切换分支

`git branch`

`git checkout master`

R2: 创建新分支 B1 和 B2

`git branch B1`

`git branch B2`

R3: 在 B2 基础上创建 C4 分支

`git checkout B2`

`git checkout -b C4`

R4: 在 C4 上修改文件并提交

使用文本编辑器修改文件

`git add .`

`git commit -m "Changes on C4"`

R5: 在 B1 上修改相同文件并提交

`git checkout B1`

使用文本编辑器进行不同修改

```
git add .
```

```
git commit -m "Changes on B1"
```

R6: 合并 C4 到 B1 并解决冲突

```
git merge C4
```

如果有冲突，手动解决后

```
git add .
```

```
git commit -m "Merged C4 into B1"
```

R7: 在 B2 上修改文件并提交

```
git checkout B2
```

使用文本编辑器修改文件

```
git add .
```

```
git commit -m "Changes on B2"
```

R8: 查看合并状态

```
git branch --merged
```

```
git branch --no-merged
```

R9: 删除已合并分支并合并未合并分支到新分支（学号命名）

```
git branch -d C4
```

```
git checkout -b <_id>
```

```
git merge B2
```

R10: 推送新分支到 GitHub

```
git push origin <_id>
```

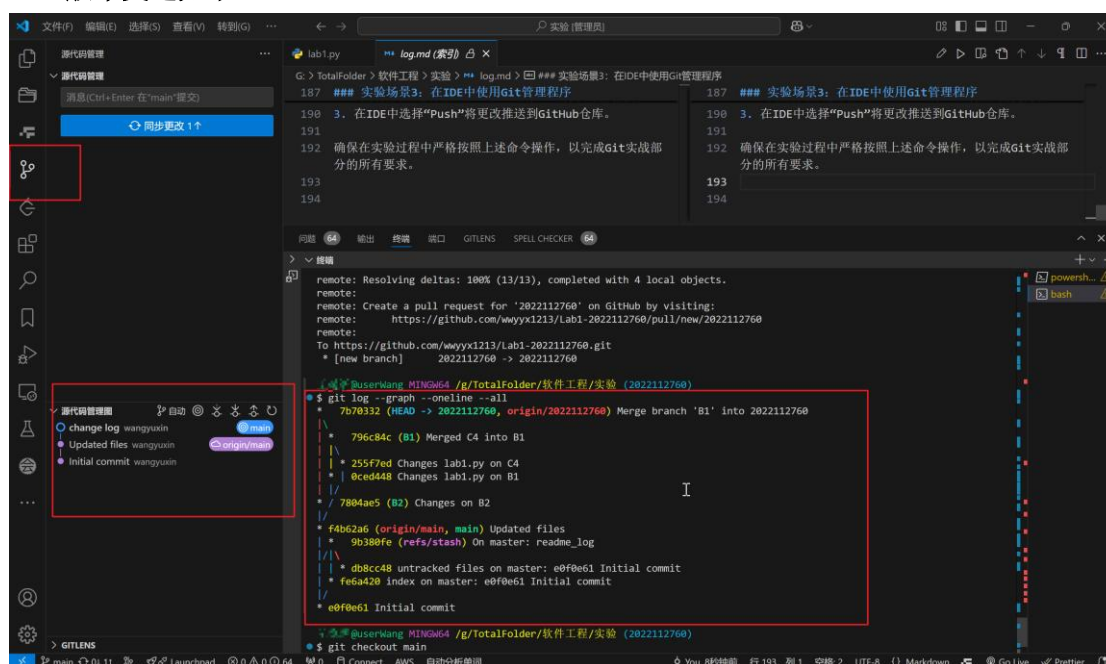
R11: 查看版本变迁树

```
git log --graph --oneline --all
```

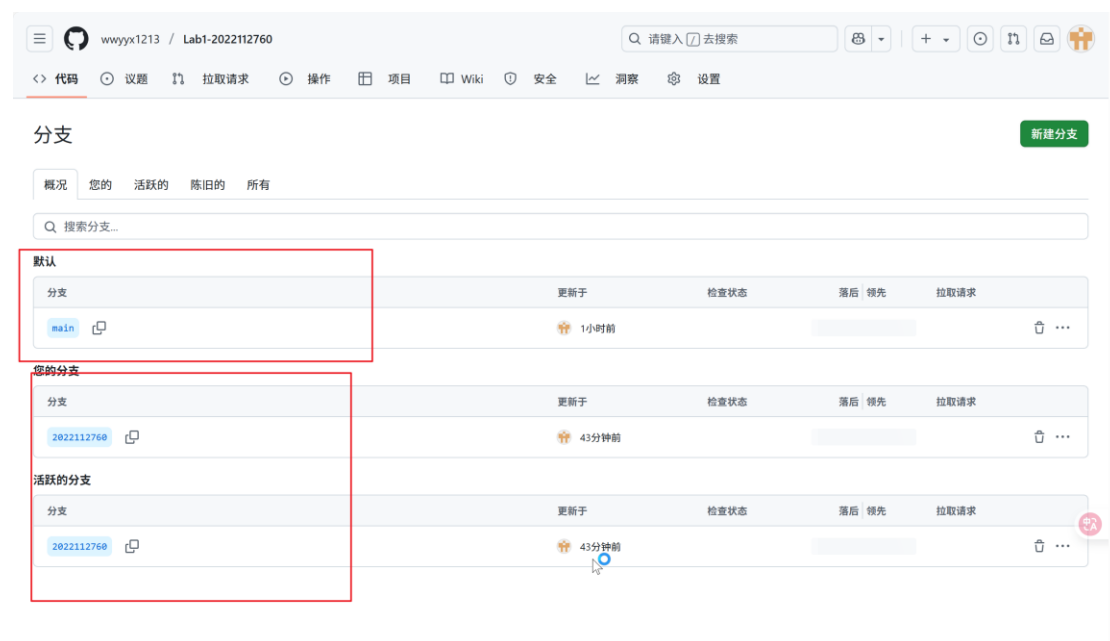
R12: 在 GitHub 上查看仓库状态

打开浏览器访问 GitHub 仓库页面

版本变迁如下:



GitHub 界面:



7 在 IDE 中使用 Git Plugin

提交代码到 GitHub 的命令:

git init

git add README.md

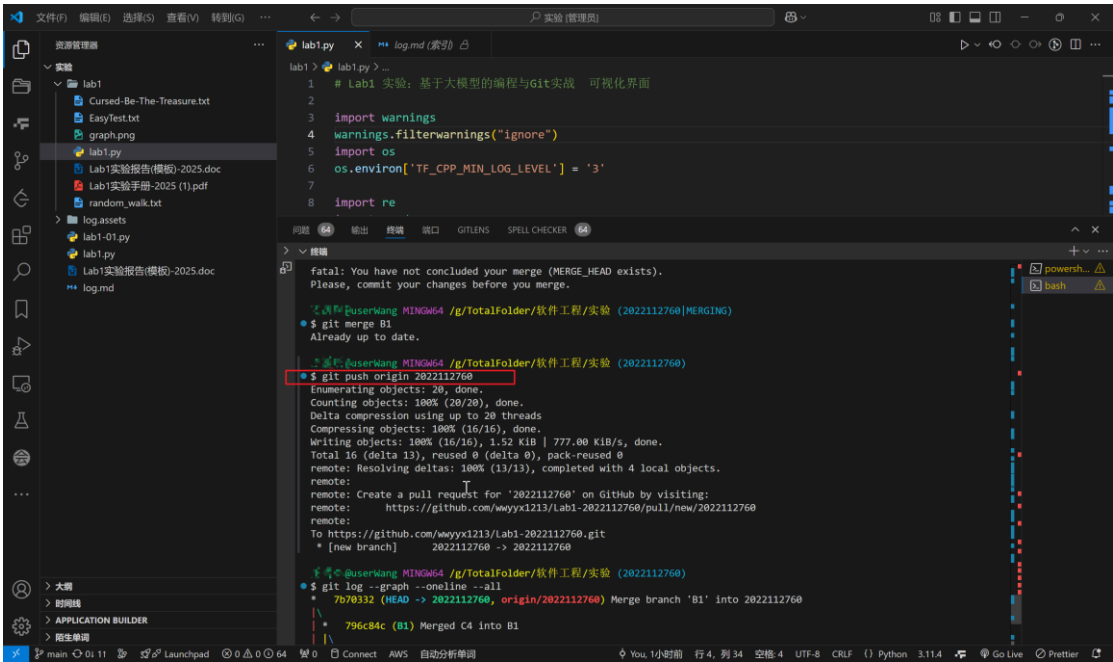
git commit -m "first commit"

git branch -M main

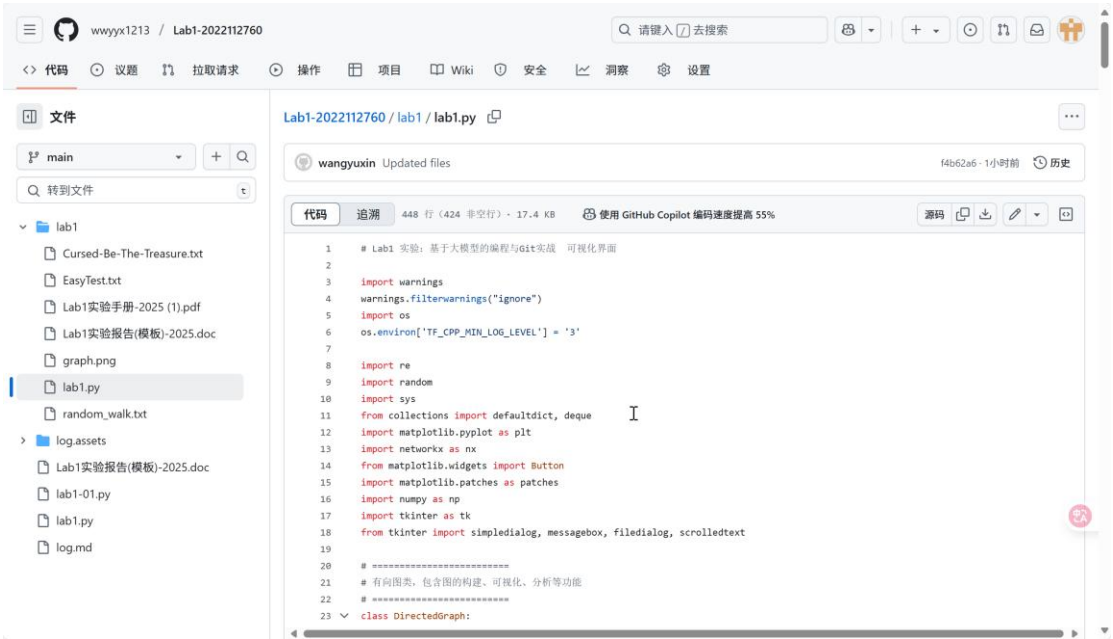
git remote add origin https://github.com/URL

git push -u origin main

IDE 使用 **vscode**, 如下所示:



GitHub 界面:



8 小结

通过本次实验，深刻体悟到大模型辅助编程的优势与劣势：

- 优势：大模型能够快速提供思路和代码示例，帮助理解复杂概念和算法实现，加快开发进度。例如在设计有向图数据结构和 PageRank 算法时，参考大模型给出的思路，能够更高效地构思解决方案。
- 劣势：大模型生成的代码可能不够精准，需要进一步分析和修改。同时，过度依赖大模型可能会削弱自主思考和深入理解问题的能力。在实验过程中，部分代码经大模型生成后，仍需人工调整以适应实际需求和优化性能。

本次实验综合运用了面向对象编程、图算法、Git 版本控制等知识，提升了编程实践能

力。通过完成各项功能需求，深入理解了文本处理、图操作和算法实现等关键技术点，为后续的团队组队完成软件开发实践奠定了坚实基础。