



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2024 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
赵帅博	2021111008	13546595163
吕雨钊	2021112636	2659136316

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	2
3.1	设计思路与算法流程图	2
3.2	数据结构设计	4
3.3	算法时间复杂度分析	4
4	实验与测试	5
4.1	读取文本文件并展示有向图	5
4.2	查询桥接词	7
4.3	根据桥接词生成新文本	8
4.4	计算最短路径	8
4.5	随机游走	11
5	编程语言与开发环境	11
6	结对编程	12
6.1	分组依据	12
6.2	角色切换与任务分工	12
6.3	工作照片	12
6.4	工作日志	13
7	Git 操作过程	13
7.1	实验场景(1): 仓库创建与提交	13
7.2	实验场景(2): 分支管理	15
8	在 IDE 中使用 Git Plugin	17
9	小结	18

1 实验要求

练习结对编程 (pair programming)，体验敏捷开发中的两人合作；

- 两人一组，自由组合；
- 使用一台计算机，共同编码，完成实验要求；
- 在工作期间，两人的角色至少切换 6 次；
- 选择一门支持面向对象的编程语言，推荐 Java

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；
- 本部分实验由个人单独完成。

2 待求解问题描述

实现从文本文件中读取数据并根据要求生成图结构，输出该图结构，并在其上进行一系列计算操作，实时展示各操作的结果。

1) 要求 1、2：读入文本，生成有向图并展示

输入数据：包含有纯英文文本数据的文本文件。

输出数据：由输入数据根据相邻关系建立的可视化有向图。

约束条件：对于若干行的文本数据，仅识别字母字符（A-Za-z）构成的单词，忽略其他字符，并将全部文本连接为一个字符串；可以调用外部绘图库或绘图工具 API 自动生成有向图，但不能采用手工方式绘图。

2) 要求 3：查询桥接词

输出数据：两个英文单词 word1, word2。

输出数据：根据有向图计算得到的对应桥接词。

约束条件：桥接词 word3 的标准为图中存在两条边 word1→word3, word3→word2；输入单词在图中不存在时需要给出提示信息；不存在对应的桥接词时需要给出提示信息；存在多个桥接词时需要全部输出；要求不可以使用外部算法库（以下要求 4-6 相同）。

3) 要求 4：根据桥接词生成新文本

输入数据：一行文本 text。

输出数据：根据有向图将 text 插入桥接词得到的新文本。

约束条件：计算 text 中两两相邻的单词的桥接词，将桥接词插入 text 的两个单词之间构造新文本。如果两个单词无桥接词，则不插入；如果两个单词之间存在多个桥接词，则随机选择一个插入。

4) 要求 5：计算两个单词之间的最短路径

输入数据：两个英文单词 word1, word2。

输出数据：根据有向图计算两个单词间的最短路径和路径长，同时在可视化图上标注对应路径。

约束条件：最短路径指路径上所有边权值之和最小；如果有多条最短路径，只需要展示一条即可；若两个单词间不可达则输出提示信息；若只输入一个单词 word1，则计算 word1 到图中任一结点的最短路径并全部输出。

5) 要求 6: 随机游走

输入数据：无

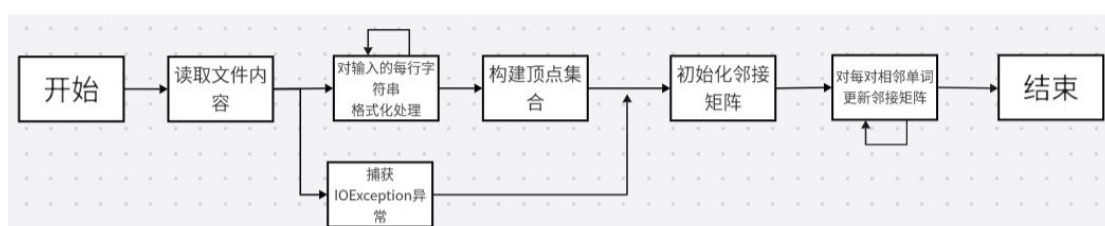
输出数据：根据有向图计算得到的随机游走路径

约束条件：在有向图中随机选择开始节点，沿出边进行随机遍历，记录经过的所有节点和边，出现第一条重复边或进入无出边的节点时停止；需要实现在遍历过程中可随时停止遍历；输出结果需要以文件形式写入磁盘。

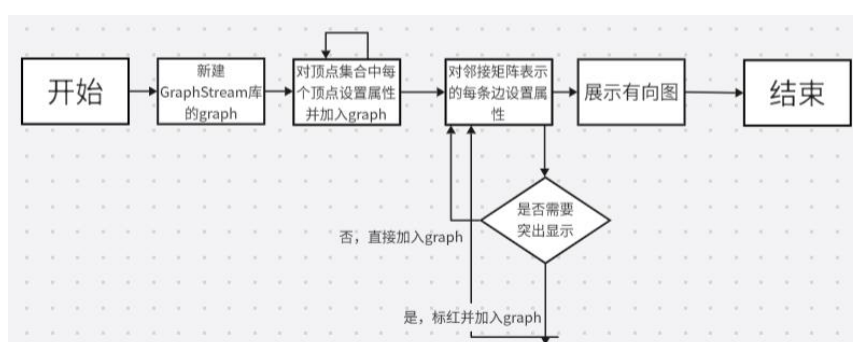
3 算法与数据结构设计

3.1 设计思路与算法流程图

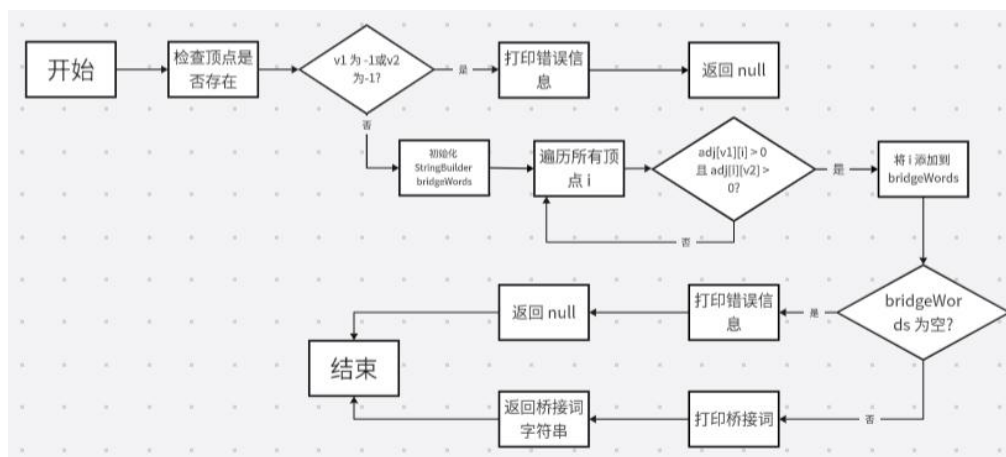
1) 读入文本，生成有向图



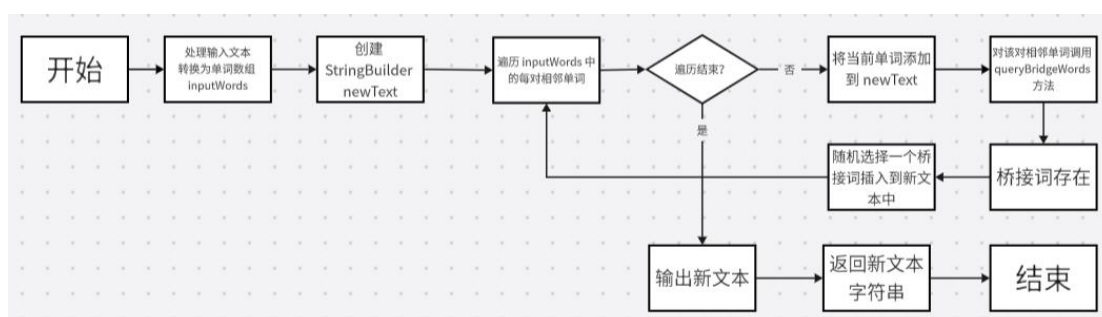
2) 展示有向图



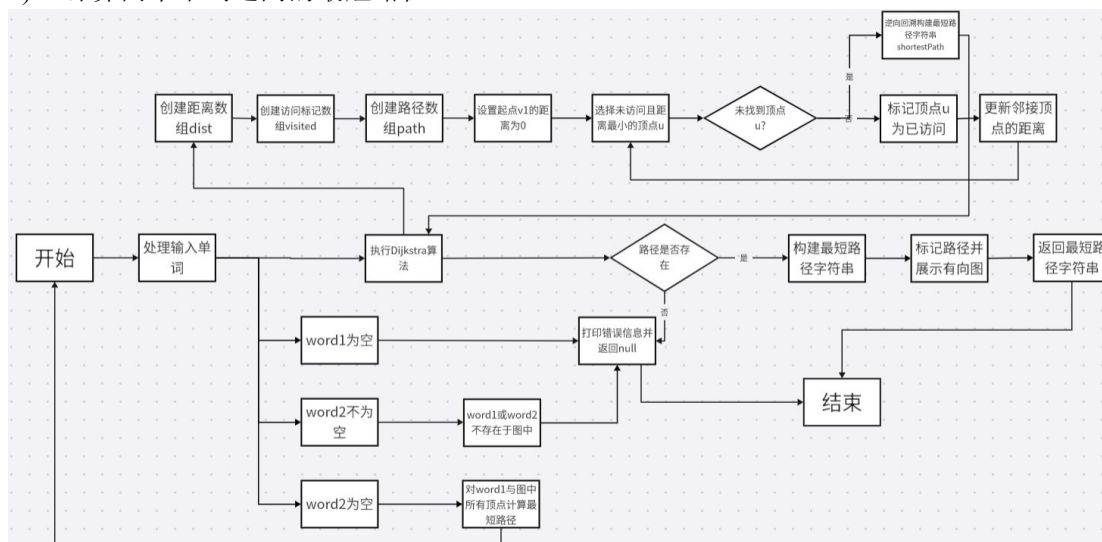
3) 查询桥接词



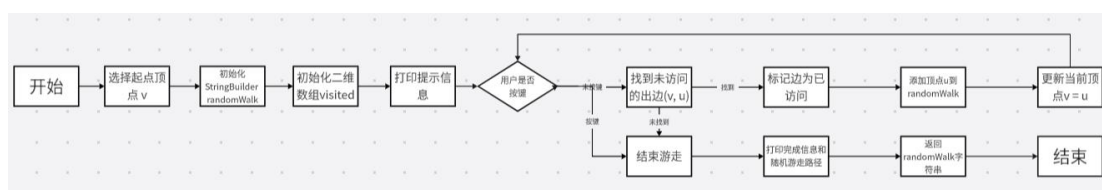
4) 根据桥接词生成新文本



5) 计算两个单词之间的最短路径



6) 随机游走



3.2 数据结构设计

有向图数据结构:

顶点集使用集合数据结构, 便于对输入文本中的单词进行去重;

邻接矩阵使用 `int` 类型数组, 可存放边的权值

```
private Set<String> vertices; // 顶点集合
private int[][] adj; // 邻接矩阵
```

计算最短路径、随机游走所用的标记数组:

标记数组与邻接矩阵大小一致

```
boolean[][] visited;
```

计算最短路径所用的路径记录数组:

路径记录数组与顶点数组大小一致, 记录所有顶点到起始点最短路径的下一个路径点

```
int[] path;
```

3.3 算法时间复杂度分析

设输入文本共有 T 个字符, 处理后文本有 N 个单词, 其中不重复的有 V 个, 即有向图顶点数:

1) 读入文本, 生成有向图

- 读取文件和处理每行文本的时间复杂度为 $O(T)$ 。
- 初始化邻接矩阵的时间复杂度为 $O(V^2)$ 。
- 构建文本和更新邻接矩阵的时间复杂度为 $O(N)$, 且 $N \approx T/W$

因此整体时间复杂度可以近似表示为: $O(T+V^2)$

在最坏情况下, 如果顶点数 V 很大, 时间复杂度可能会被邻接矩阵的初始化和处理所支配, 达到 $O(V^2)$ 。否则, 时间复杂度主要取决于输入文本的长度 T 。

2) 展示有向图

主要时间复杂度来自于添加节点和边的过程。添加节点的时间复杂度为 $O(V)$, 而添加边的时间复杂度为 $O(V^2)$ 。因此, 整个方法的时间复杂度为: $O(V^2)$ 。

3) 查询桥接词

获取顶点索引: $v1 = \text{getVIndex}(\text{word1})$ 和 $v2 = \text{getVIndex}(\text{word2})$ 的时间复杂度为 $O(V)$; 查找桥接词的过程中需要遍历所有顶点, 时间复杂度为 $O(V)$ 。因此, 整个方法的时间复杂度为: $O(V)$ 。

4) 根据桥接词生成新文本

设输入文本字符数为 L ，总单词数为 M 。

预处理输入文本的时间复杂度为 $O(L)$ 。遍历输入文本的每对相邻单词，调用时间复杂度为 $O(V)$ 的查询桥接词的方法，总时间复杂度约为 $O(M * V)$ 。 L 与 M 线性相关，因此总时间复杂度为 $O(L * V)$ 。

5) 计算两个单词之间的最短路径

获取顶点索引的时间复杂度为 $O(V)$ ；每个数组的初始化时间复杂度为 $O(V)$ ；Dijkstra 算法外层循环运行 V 次，每次查找最小距离节点的时间复杂度为 $O(V)$ ，内层循环更新距离，时间复杂度为 $O(V)$ ，总时间复杂度为 $O(V^2)$ ；从目标节点回溯到起点，构建路径的时间复杂度为 $O(V)$ ；对于图中每个顶点，最坏情况下需要检查所有其他顶点的距离，因此总时间复杂度为 $O(V^2)$ 。

6) 随机游走

初始化 `visited` 数组时间复杂度为 $O(V^2)$ 。随机游走时，最坏情况下会遍历所有边，导致每次找邻接节点的时间复杂度为 $O(V)$ 。假设边数为 E ，则 `while` 循环时间复杂度= $O(E \times V)$ 总时间复杂度= $O(V^2)+O(E \times V)$ 。

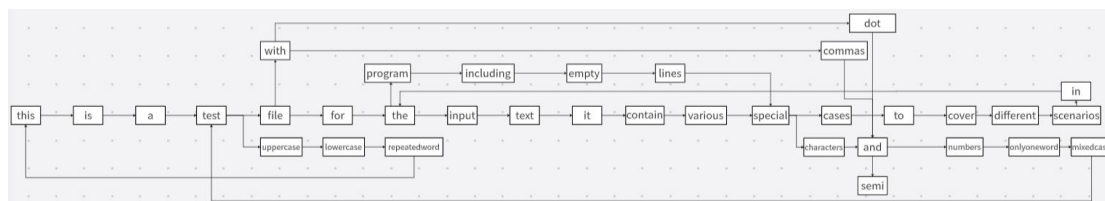
4 实验与测试

4.1 读取文本文件并展示有向图

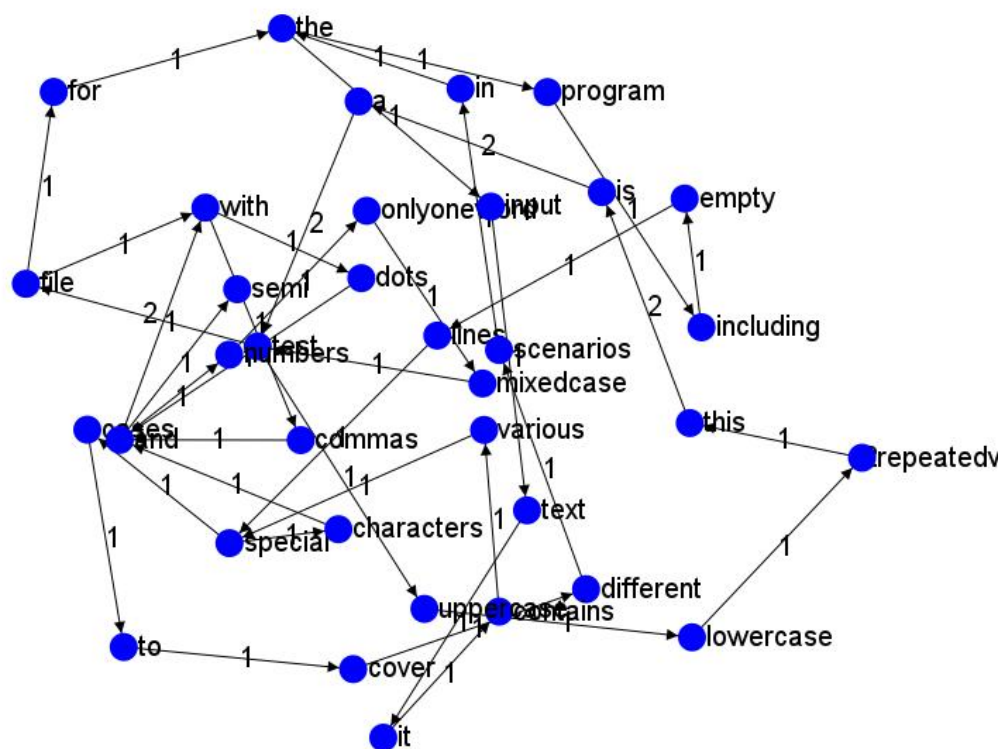
文本文件中包含的内容：

```
This is a test file for the input text.  
It contains various special cases to cover different scenarios in the program.  
Including: empty lines,  
, special characters! and numbers,  
OnlyOneWord  
MIXEDcaSe TesT  
UPPERCASE  
lowercase  
RepeatedWord RepeatedWord RepeatedWord  
This,is,a,test,file,with,commas,and,with.dots.and;semi
```

期望生成的图（手工计算得到）：



程序实际生成的图：



二者是否一致：一致

给出实际运行得到结果的界面截图。


```
1
Please input two words:
with
and
The bridge words from "with" to "and" are: commas dots

1
Please input two words:
empty
cover
No bridge words from "empty" to "cover"!
```

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	说明
1	The including lines	The new text is: the program including empty lines	The new text is: the program including empty lines	正确	输入文本中不止一对相邻单词存在桥接词
2	it special cases	The new text is: it special cases	The new text is: it special cases	正确	无桥接词
3	the text contains it and text contains another	The new text is: the input text it contains it and text it contains another	The new text is: the input text it contains it and text it contains another	正确	输入文本存在重复的存在桥接词的相邻单词对

给出实际运行得到结果的界面截图。

```
2
Please input the text:
The including lines
The new text is: the program including empty lines

2
Please input the text:
it special cases
The new text is: it special cases

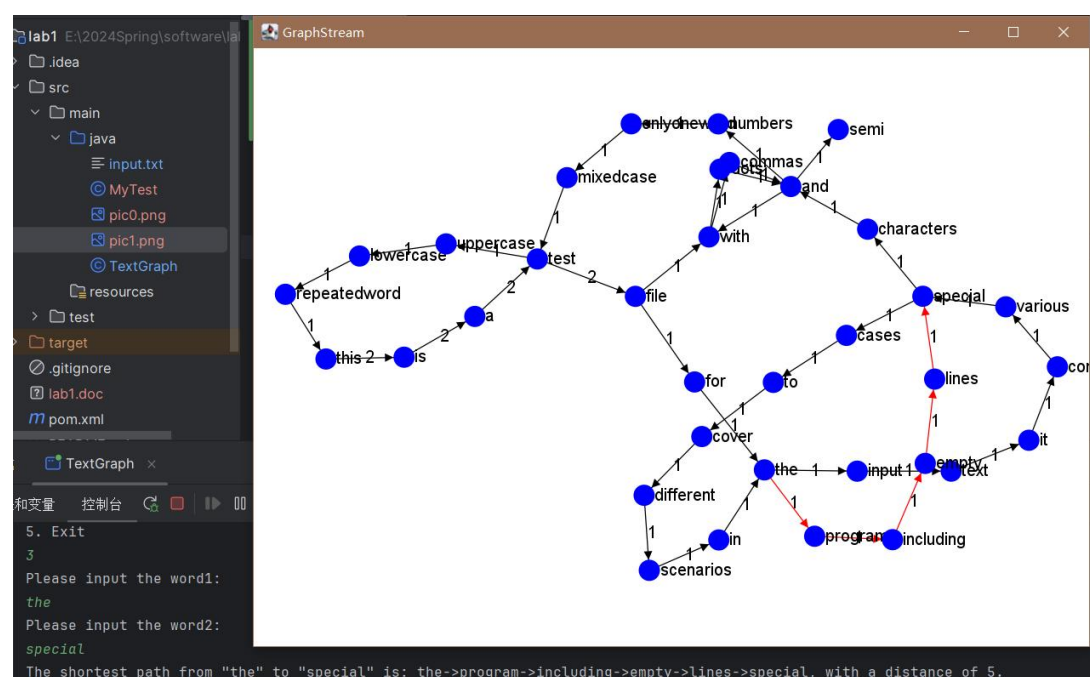
2
Please input the text:
the text contains it and text contains another
The new text is: the input text it contains it and text it contains another
```

4.4 计算最短路径

序号	输入（两	期望输出	实际输出	运行是否正确	说明
----	------	------	------	--------	----

	个单词、或一个单词)				
1	the、special	The shortest path from "the" to "special" is: the->program->including->empty->lines->special, with a distance of 5.	The shortest path from "the" to "special" is: the->program->including->empty->lines->special, with a distance of 5.	正确	两个单词间不止一条路径
2	semi、and	No path from "semi" to "and"!	No path from "semi" to "and"!	正确	不可达
3	test、/	The shortest path from "test" to other words are: The shortest path from "test" to "uppercase" is: test->uppercase, with a distance of 1. The shortest path from "test" to "commas" is: test->file->with->commas, with a distance of 4. The shortest path from "test" to "for" is: test->file->for, with a distance of 3. ... (test 到所有可达顶点的最短路径)	The shortest path from "test" to other words are: The shortest path from "test" to "uppercase" is: test->uppercase, with a distance of 1. The shortest path from "test" to "commas" is: test->file->with->commas, with a distance of 4. The shortest path from "test" to "for" is: test->file->for, with a distance of 3. ... (test 到所有可达顶点的最短路径)	正确	只输入一个单词

给出实际运行得到结果的界面截图。



```
3
Please input the word1:
semi
Please input the word2:
and
No path from "semi" to "and"!
```

```
Please input the word1:
test
Please input the word2:

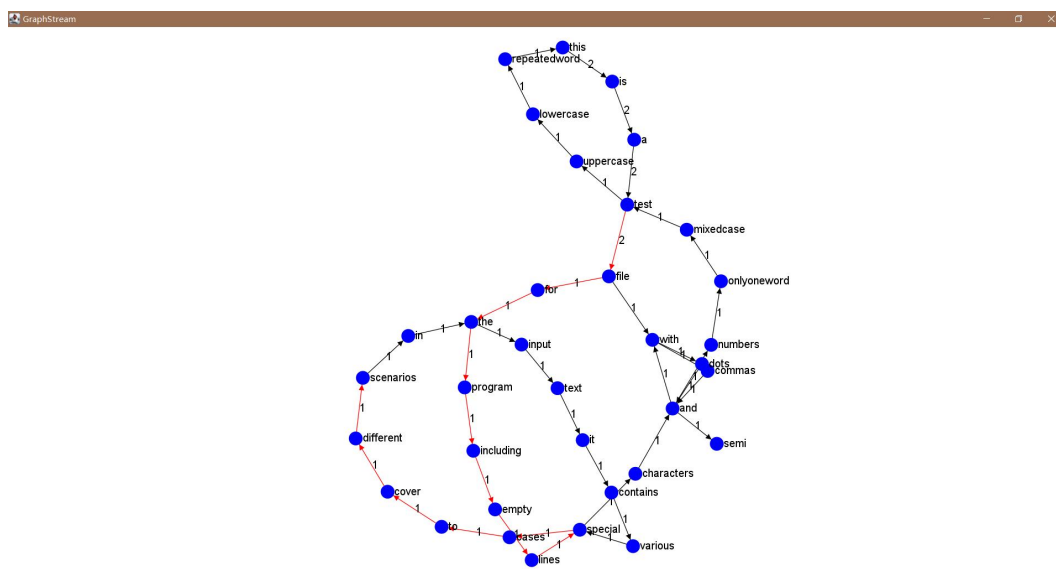
/
=====
The shortest path from "test" to other words are:

The shortest path from "test" to "uppercase" is: test->uppercase, with a distance of 1.
The shortest path from "test" to "commas" is: test->file->with->commas, with a distance of 4.
The shortest path from "test" to "for" is: test->file->for, with a distance of 3.
The shortest path from "test" to "numbers" is: test->file->with->commas->and->numbers, with a distance of 6.
The shortest path from "test" to "scenarios" is: test->file->for->the->program->including->empty->lines->special->cases->to->cover->different->scenarios, with a distance of 12.
The shortest path from "test" to "program" is: test->file->for->the->program, with a distance of 5.
The shortest path from "test" to "empty" is: test->file->for->the->program->including->empty, with a distance of 7.
The shortest path from "test" to "cover" is: test->file->for->the->program->including->empty->lines->special->cases->to->cover, with a distance of 10.
The shortest path from "test" to "onlyoneword" is: test->file->with->commas->and->numbers->onlyoneword, with a distance of 7.
The shortest path from "test" to "characters" is: test->file->for->the->program->including->empty->lines->special->characters, with a distance of 10.
The shortest path from "test" to "file" is: test->file, with a distance of 2.
The shortest path from "test" to "and" is: test->file->with->commas->and, with a distance of 5.
The shortest path from "test" to "mixedcase" is: test->file->with->commas->and->numbers->onlyoneword->mixedcase, with a distance of 8.
The shortest path from "test" to "text" is: test->file->for->the->input->text, with a distance of 6.
The shortest path from "test" to "semi" is: test->file->with->commas->and->semi, with a distance of 6.
The shortest path from "test" to "different" is: test->file->for->the->program->including->empty->lines->special->cases->to->cover->different, with a distance of 12.
The shortest path from "test" to "lines" is: test->file->for->the->program->including->empty->lines, with a distance of 8.
The shortest path from "test" to "a" is: test->uppercase->lowercase->repeatedword->this->is->a, with a distance of 8.
The shortest path from "test" to "space" is: test->file->for->the->program->including->empty->lines->special->cases->to->cover->different, with a distance of 12.
```

(截取部分)

- pic3.png
- pic4.png
- pic5.png
- pic6.png
- pic7.png
- pic8.png
- pic9.png
- pic10.png
- pic11.png
- pic12.png
- pic13.png
- pic14.png

(所有路径均进行了可视化)



(取其中一例)

4.5 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	说明
1	The final random walk is: various special characters and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including empty lines special cases to cover different scenarios in the input text it contains various	正确	程序自动运行至停止
2	The final random walk is: and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including empty lines special characters and semi	正确	程序自动运行至停止
3	The final random walk is: input text it contains various special characters	正确	键入 enter 停止遍历

给出实际运行得到结果的界面截图。

```
4
=====
Press any key to stop.
The random walk is:   various special characters and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including empty l
Done!
The final random walk is:  various special characters and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including emp
=====

4
=====
Press any key to stop.
The random walk is:   and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including empty lines special characters an
Done!
The final random walk is:  and numbers onlyoneword mixedcase test uppercase lowercase repeatedword this is a test file for the program including empty lines special character
=====

4
=====
Press any key to stop.
The random walk is:   input text it contains various special characters

Done!
The final random walk is:  input text it contains various special characters
=====
```

5 编程语言与开发环境

Java DK 版本：Amazon Corretto 17.0.9
IDE 版本：IntelliJ IDEA 2023.3.4
图可视化库：GraphStream 1.3

6 结对编程

6.1 分组依据

性格方面，我们一个细心沉稳，专注认真；一个思想跳脱，思维发散。

能力方面，我们一个善于分析问题，注重细节，能够深入思考并且有耐心地解决复杂的任务，在编程方面可能更加注重程序的稳定性和可靠性，善于进行逻辑推理与程序构建。另一个更富有创造力和想象力，思维敏捷，善于从不同角度思考问题，能够提出新颖的观点和解决方案，擅长发散性思维和创意性的编程方法，能够在算法设计时带来新鲜的思维和灵感。

另外，我们在同一寝室，可以随时交流想法，并随时在讨论的过程中将想法付诸实践。

6.2 角色切换与任务分工

日期	时间(HH:MM -- HH:MM)	“驾驶员”	“领航员”	本段时间的任务
5.19	13: 30 - 13: 50	赵帅博	吕雨钊	功能需求 1 的基本实现
5.19	14: 00 - 14: 30	吕雨钊	赵帅博	功能需求 2 的基本实现
5.19	14: 45 - 15: 15	赵帅博	吕雨钊	功能需求 3 的基本实现
5.19	15: 15 - 15: 50	吕雨钊	赵帅博	功能需求 4 的基本实现
5.22	10: 30 - 11: 30	赵帅博	吕雨钊	功能需求 5 的基本实现
5.22	14: 00 - 14: 30	吕雨钊	赵帅博	功能需求 6 的基本实现
5.22	14: 30 - 15: 30	赵帅博	吕雨钊	代码优化与程序调试
5.22	15: 35 - 16: 10	吕雨钊	赵帅博	程序测试

6.3 工作照片





6.4 工作日志

日期/时间	问题描述	最终解决方法	两人如何通过交流找到解决方法
5.19 13: 30 - 14: 30	不熟悉 Java 语言的一些基本语法, 包括文件读写等	在网络上搜寻样例进行学习	通过网络上的程序样例与讲解共同探讨并学习, 并互相辅助地完成代码的编写
5.19 13: 30 - 14: 30	有向图可视化库的选择与如何使用	选定了 GraphStream 库, 并学习使用	分别上网搜索相关库, 并将搜索结果进行交流与讨论, 最终确定
5.19 14: 45 - 15: 50	功能的实现过程中可能存在错误计算	使用与指导书相同的测试用例一边编写程序一边调试	对程序运行的结果与指导手册样例的比对的差异进行讨论, 检查代码可能存在的问题
5.22 14: 30 - 15: 30	代码的组织有些混乱, 不易调试	对代码结构进行了优化	共同反复查看代码中可能存在的问题, 讨论解决方法
5.22 15: 35 - 16: 10	测试样例可能不够全面	使用更多各种情况的测试用例进行了测试	在讨论的过程中想出尽可能覆盖程序各种输入情况的测试用例

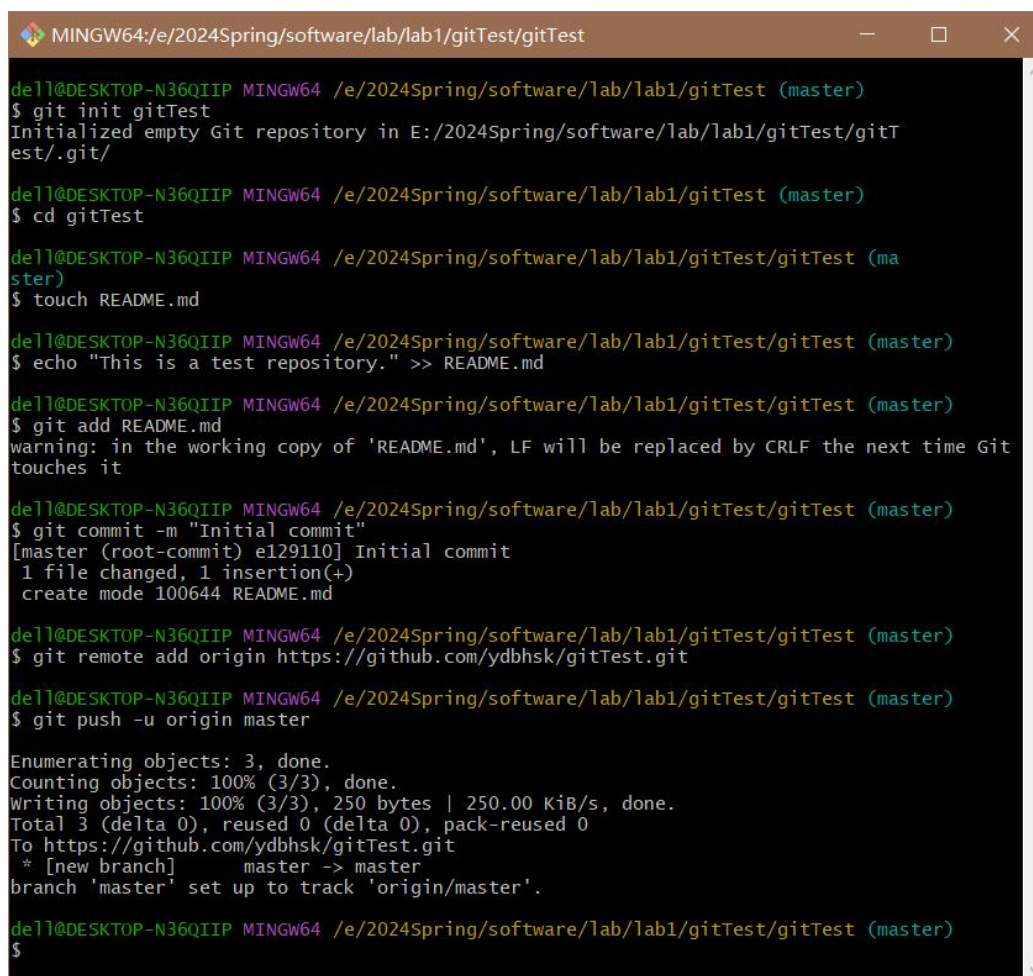
7 Git 操作过程

7.1 实验场景(1): 仓库创建与提交

- 1) 创建新的仓库: `git init my_repo`

- 2) 进入新仓库目录: `cd my_repo`
- 3) 创建一个新文件并添加内容:
`touch README.md`
`echo "This is a test repository." >> README.md`
- 4) 添加文件到暂存区 `git add README.md`
- 5) 提交文件到仓库 `git commit -m "Initial commit"`
- 6) 在 Github 上创建一个新的仓库, 并将本地仓库与 Github 远程仓库关联:
- 7) 在 Github 上创建一个新的仓库, 并获取仓库的 URL。
- 8) 将本地仓库与 Github 远程仓库关联: `git remote add origin <repository_url>`
- 9) 推送本地提交到 Github 远程仓库 `git push -u origin master`

执行界面的截图（命令输入界面和结果界面）：



```
MINGW64:/e/2024Spring/software/lab/lab1/gitTest/gitTest
de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest (master)
$ git init gitTest
Initialized empty Git repository in E:/2024Spring/software/lab/lab1/gitTest/gitTest/.git/

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest (master)
$ cd gitTest

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ touch README.md

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ echo "This is a test repository." >> README.md

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git commit -m "Initial commit"
[master (root-commit) e129110] Initial commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

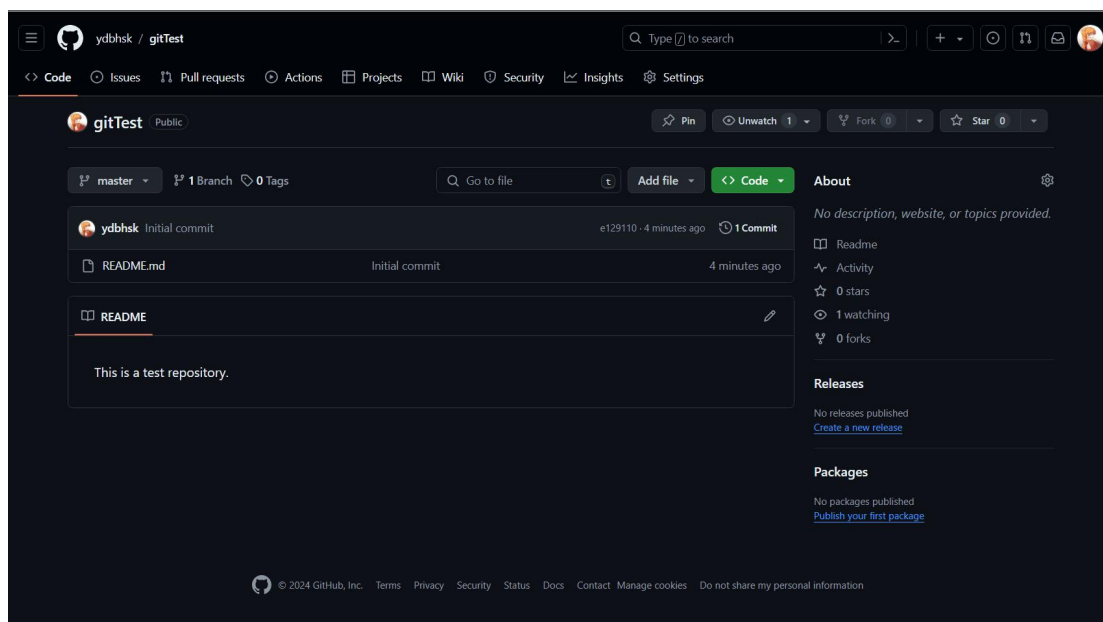
de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git remote add origin https://github.com/ydbhsk/gitTest.git

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git push -u origin master

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 250 bytes | 250.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ydbhsk/gitTest.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

de11@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$
```

Github 仓库界面



7.2 实验场景(2): 分支管理

- 1) 创建新分支并切换到该分支 `git checkout -b feature_branch`
 - 2) 在新分支上进行修改: 在新分支上进行一些修改和提交操作, 例如修改 `README.md` 文件并添加一些新内容
 - 3) 添加修改后的文件到暂存区 `git add README.md`
 - 4) 提交修改到新分支 `git commit -m "Add new content to README.md"`
 - 5) 切换回主分支 `git checkout master`
 - 6) 合并新分支到主分支 `git merge feature_branch`
 - 7) 删除新分支 `git branch -d feature_branch`
- 执行界面的截图(命令输入界面和结果界面):

```
MINGW64:/e/2024Spring/software/lab/lab1/gitTest/gitTest
dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git checkout -b feature_branch
Switched to a new branch 'feature_branch'

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git add README.md

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git add newbranch.txt

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git commit -m "Add new file"
[feature_branch b4f223b] Add new file
1 file changed, 1 insertion(+)
create mode 100644 newbranch.txt

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git commit -m "Add new content to README.md"
[feature_branch 5c8ecbc] Add new content to README.md
1 file changed, 2 insertions(+)

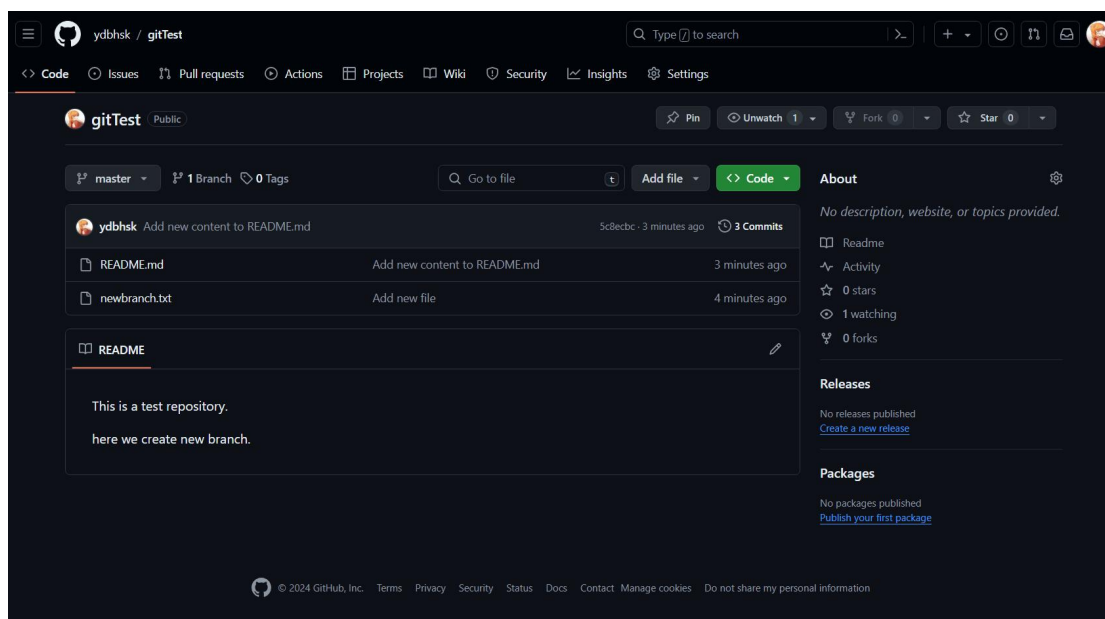
dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (feature_branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git merge feature_branch
Updating e129110..5c8ecbc
Fast-forward
 README.md      | 2 ++
 newbranch.txt  | 1 +
 2 files changed, 3 insertions(+)
 create mode 100644 newbranch.txt

dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git branch -d feature_branch
Deleted branch feature_branch (was 5c8ecbc).

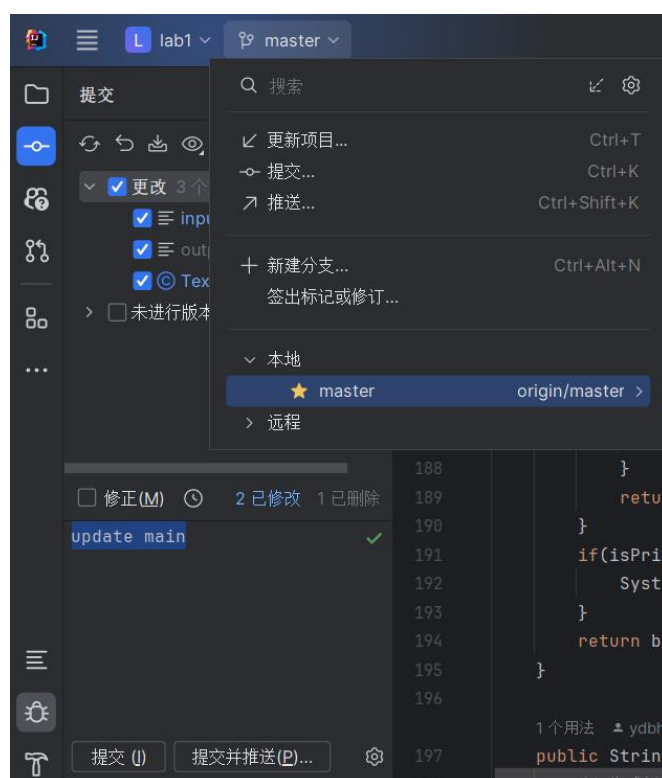
dell@DESKTOP-N36QIIP MINGW64 /e/2024Spring/software/lab/lab1/gitTest/gitTest (master)
$ git push -u origin master
```

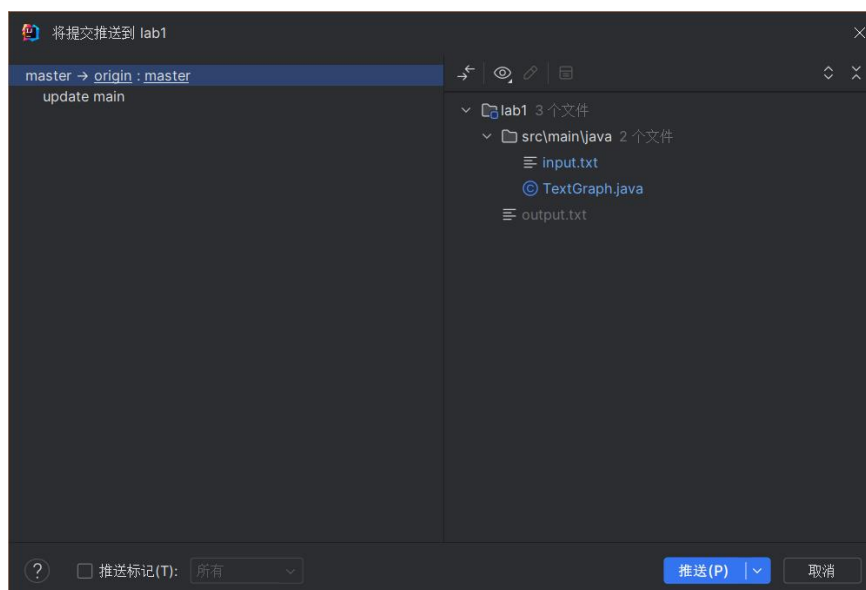
将修改推送到 Github，仓库界面如下图，成功修改



8 在 IDE 中使用 Git Plugin

在 IDEA 中使用版本控制工具，如下图，对于存在更改的文件可输入提交信息，进行提交，并推送至 Github 远程仓库对应分支上





9 小结

在本次实验中，通过结对编程和学习 Git，我深刻理解了团队协作的价值和原则。我学会了与队友有效沟通、分享想法和解决问题。在学习 Git 的过程中，我掌握了分支管理、版本控制和协同开发的技能，学会了如何避免代码冲突、合并分支以及恢复历史版本。我也意识到了代码提交和代码审查的重要性，以及如何利用 Git 工具进行团队合作和项目管理。同时，我的团队合作，团队项目处理和编程能力也得到了提升。

本实验 Github 库链接：[ydbhsk/HIT-SoftwareLab-2024Spring \(github.com\)](https://github.com/ydbhsk/HIT-SoftwareLab-2024Spring)