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

目 录

1 实验要求

创建一个应用程序，旨在实现以下功能：程序需能够读取来自文本文件的数据，并依据这些数据构建一个图结构。构建完毕后，系统应展示该图结构，并支持在该图上执行一连串的计算操作，期间要即时反馈每一步操作的结果。此应用程序具备灵活性，既可以通过命令行界面运行，也可通过图形用户界面（GUI）来操作，确保在两种使用模式下，所有核心功能均得到有效支持与实现。练习结对编程，体验敏捷开发中的两人合作。

2 待求解问题描述

待求解问题为一个程序，实现文本有向图生成分析，使之能实现文本的有向图生成展示，桥接词查询，新文本生成，最短路径计算和随机游走的功能。

功能的描述如下表所示：

功能名称	输入数据	输出数据	约束条件
有向图生成展示	输入文本	有向图结构	满足文本需求
桥接词查询	两个待分析单词	单词的桥接词或空	输入需满足格式要求
新文本生成	文本	新生成的文本	输入需满足格式要求
最短路径计算	两个待分析单词	单词的路径	输入需满足格式要求
随机游走	文本输出位置	随机游走文本	输入需满足格式要求

3 算法与数据结构设计

3.1 设计思路与算法流程图

1 生成图：从文本文件中逐行读取数据，对每行进行清洗，只保留字母并转换为小写。将每行分割成单词。对于每个单词，将其作为图的节点。如果当前单词和前一个单词已经存在于图中，则在这两个单词之间添加一条边。如果这条边已存在，则增加边的权重（表示两个单词相邻的频次）。最后打印图。

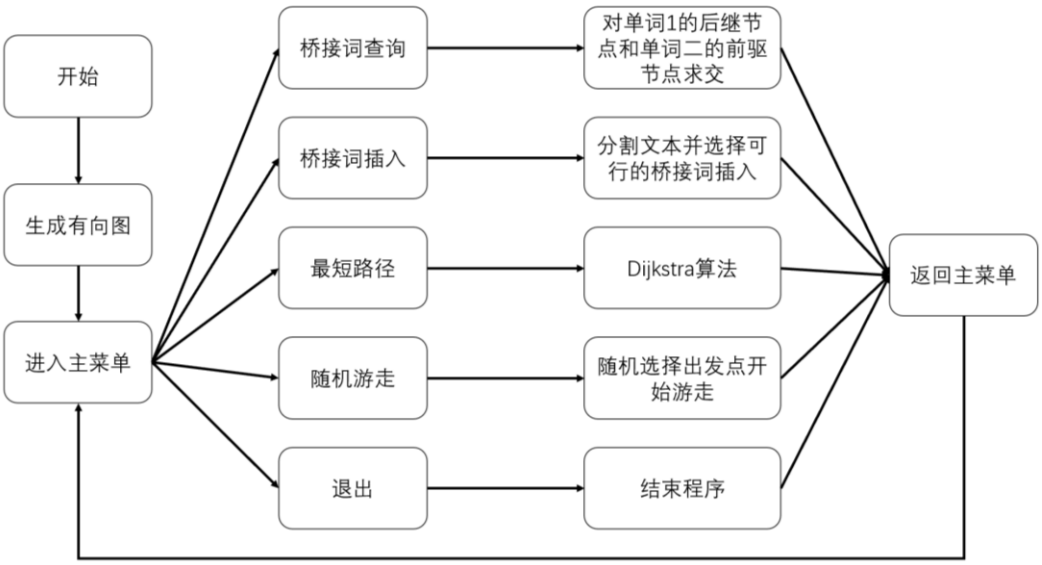
2 桥接词查询：用户输入两个单词。检查单词是否存在于图中。查询第一个单词的所有邻接点，检查这些邻接点是否与第二个单词相邻。返回所有满足条件的邻接点作为桥接词。

3 新文本生成：用户输入一段文本。对文本进行分词。对于文本中的每对相邻单词，尝试插入一个或多个桥接词，增加文本的连贯性或丰富性。

4 最短路径计算：用户输入两个单词。应用 Dijkstra 算法。返回路径及其长度。

5 随机游走：选择一个随机的起始单词。从当前单词的邻接点中随机选择一个，作为下一个单词。重复步骤直到达到所需的文本长度或用户指定停止或满足要求。

算法流程图如下：



3.2 数据结构设计

有向图数据结构：Map<String, Map<String, Integer>>
其为一个嵌套的哈希表，格式为：外层的 Map 的键是单词，值是另一个 Map。
内层的 Map 的键是邻接单词，值是这两个单词相邻的次数（权重）。

3.3 算法时间复杂度分析

有向图展示生成。依次扫描文本，并对于每一对词在图中添加一条边，故总体的时间复杂度为 O(N)。其中 N 为文本长度。

桥接词查询。对所有节点进行遍历来寻找两个词，时间复杂度 O(N)。然后取两个目标词的后继节点和前驱节点集合，并对两个集合求交。其复杂度与相邻节点数量有关，时间复杂度为 O(N)。故总体时间复杂度为 O(N)。

根据桥接词生成新文本。需要对每一对相邻的词尝试进行桥接词查询并插入。词对的数量在 O(N)量级，桥接词查询的时间复杂度为 O(N)，故总体时间复杂度为 O(N^2)。

最短路径算法采用的是 dijkstra 算法，时间复杂度为 O(NlogN)。

随机游走。随机游走取初始节点和选取下一个节点的时间复杂度均为 O(1)。于是随机游走的时间复杂度为 O(K),其中 K 为期望游走的次数。

4 实验与测试

实验测试采用一段包含英文和符号的文本构成，测试内容包括所有实现的项目。

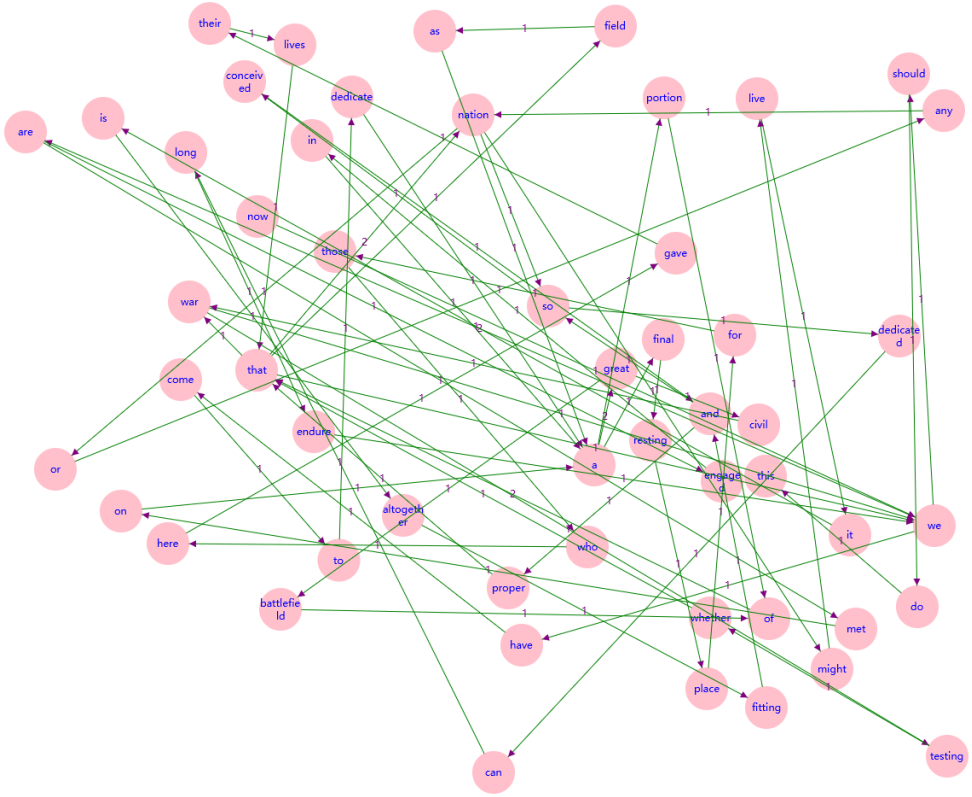
4.1 读取文本文件并展示有向图

文本文件中包含的内容：
Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are @ met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

期望生成的图（手工计算得到）：
下面的结果中，a->b(c)表示有一条从 a 指向 b 的边，并且其权重为 c（文本中 ab 相邻出现 c 次）。
nation -> or(1), might(1), so(1)
do -> this(1)
that -> that(1), field(1), nation(2), war(1), we(1)
whether -> that(1)
should -> do(1)
those -> who(1)
dedicated -> can(1)
lives -> that(1)
in -> a(1)
might -> live(1)
testing -> whether(1)
come -> to(1)
is -> altogether(1)
it -> is(1)
as -> a(1)
field -> as(1), of(1)
final -> resting(1)
gave -> their(1)
endure -> we(1)
who -> here(1)
engaged -> in(1)
here -> gave(1)
conceived -> and(1)
portion -> of(1)
for -> those(1)
their -> lives(1)
proper -> that(1)

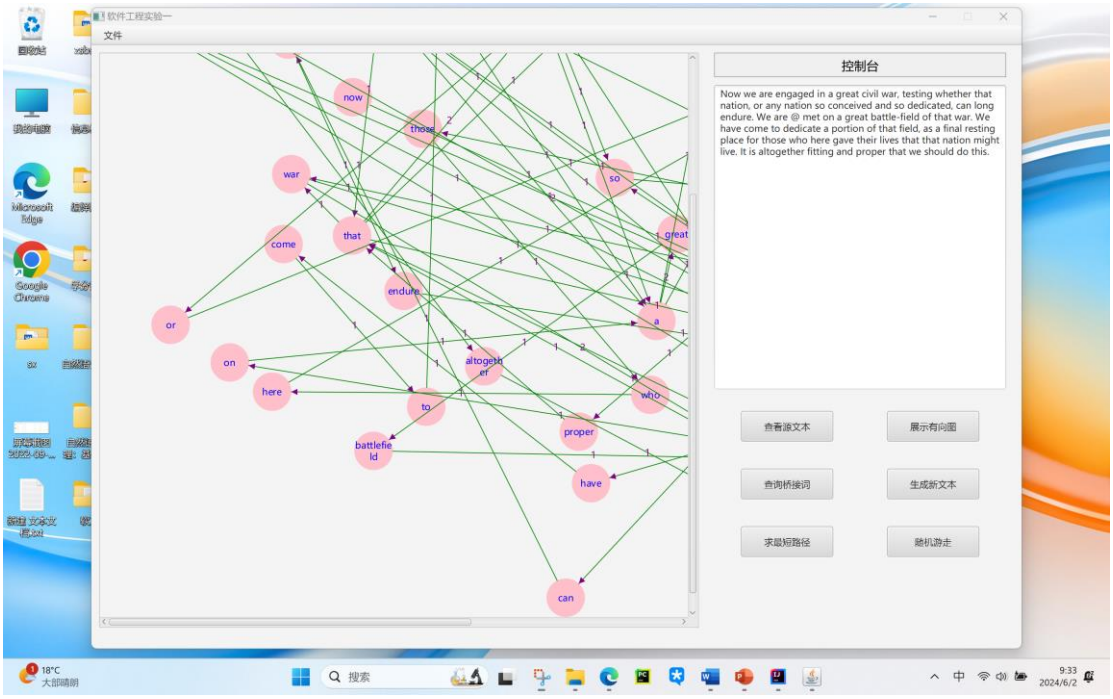
we -> are(2), have(1), should(1)
long -> endure(1)
can -> long(1)
are -> engaged(1), met(1)
and -> proper(1), so(1)
now -> we(1)
civil -> war(1)
of -> that(2)
have -> come(1)
place -> for(1)
so -> dedicated(1), conceived(1)
met -> on(1)
live -> it(1)
on -> a(1)
a -> portion(1), final(1), great(2)
or -> any(1)
resting -> place(1)
war -> testing(1), we(1)
great -> battle(1), civil(1)
any -> nation(1)
battle -> field(1)
dedicate -> a(1)
altogether -> fitting(1)
to -> dedicate(1)
fitting -> and(1)

程序实际生成的图：



二者是否一致：
两种完全一致。

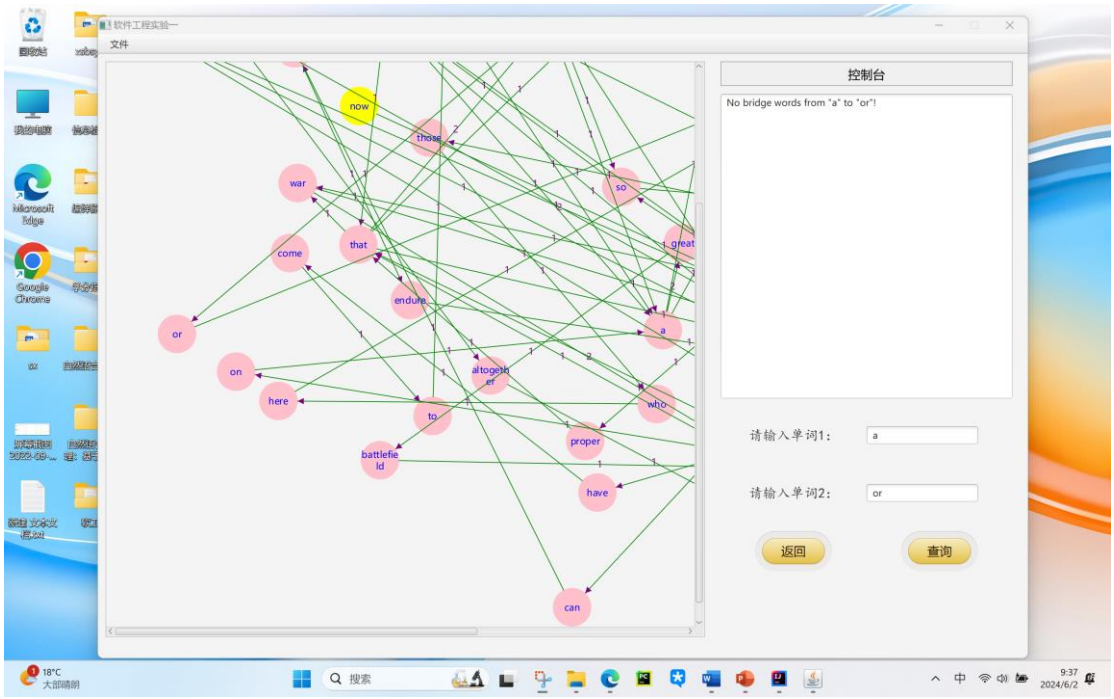
给出实际运行得到结果的界面截图。

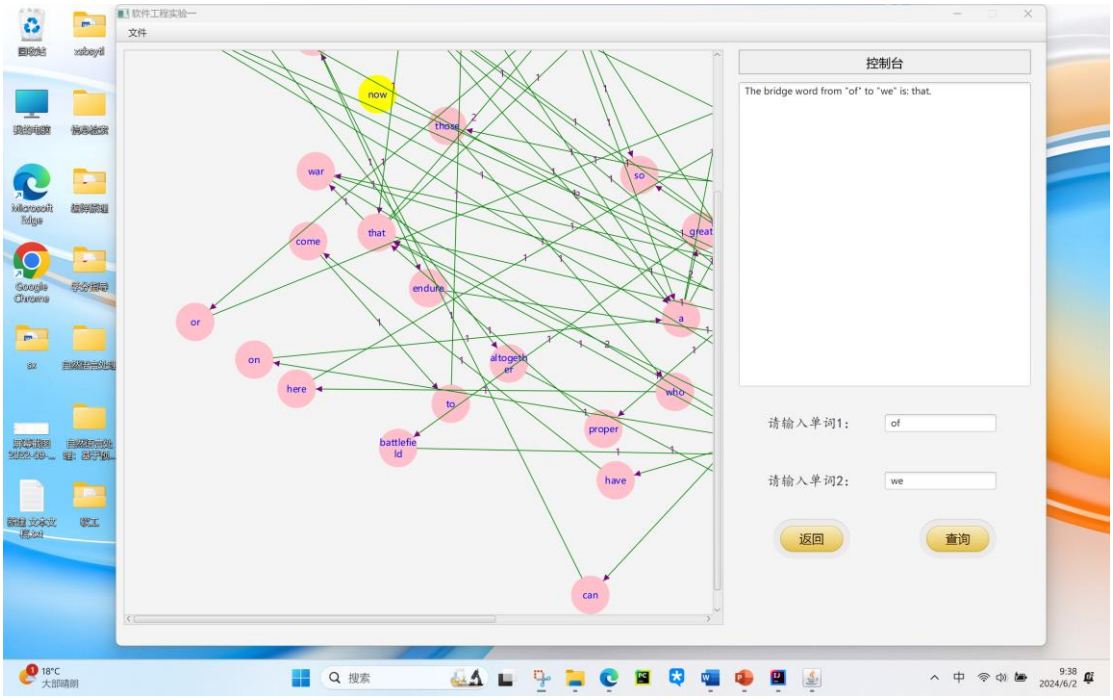


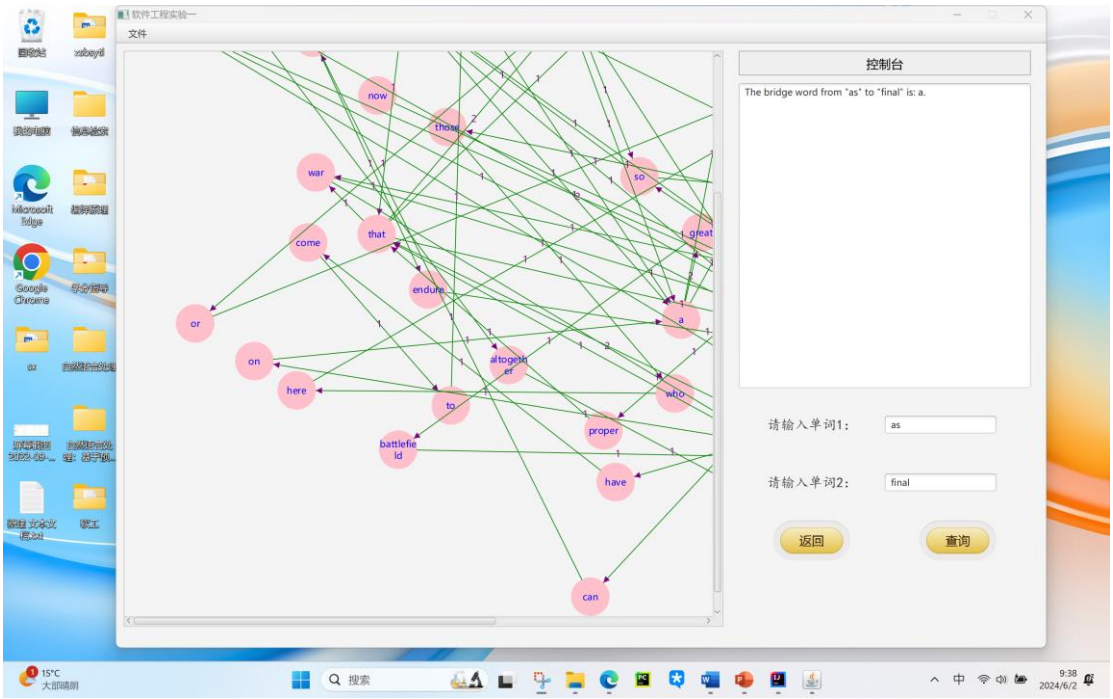
4.2 查询桥接词

序号	输入（2 个单词）	期望输出	实际输出	运行是否正确
1	a or	No bridge words from "a" to "or"!	No bridge words from "a" to "or"!	正确
2	of we	The bridge word from "of" to "we" is: that.	The bridge word from "of" to "we" is: that.	正确
3	as final	The bridge word from "as" to "final" is: a.	The bridge word from "as" to "final" is: a.	正确

给出实际运行得到结果的界面截图。



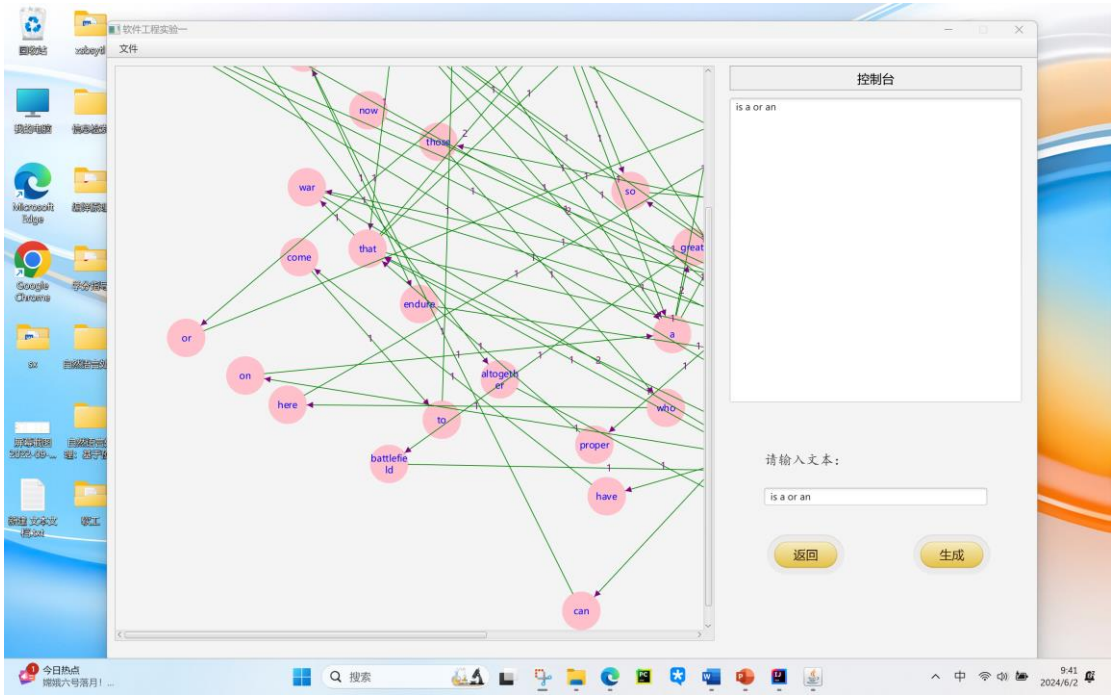


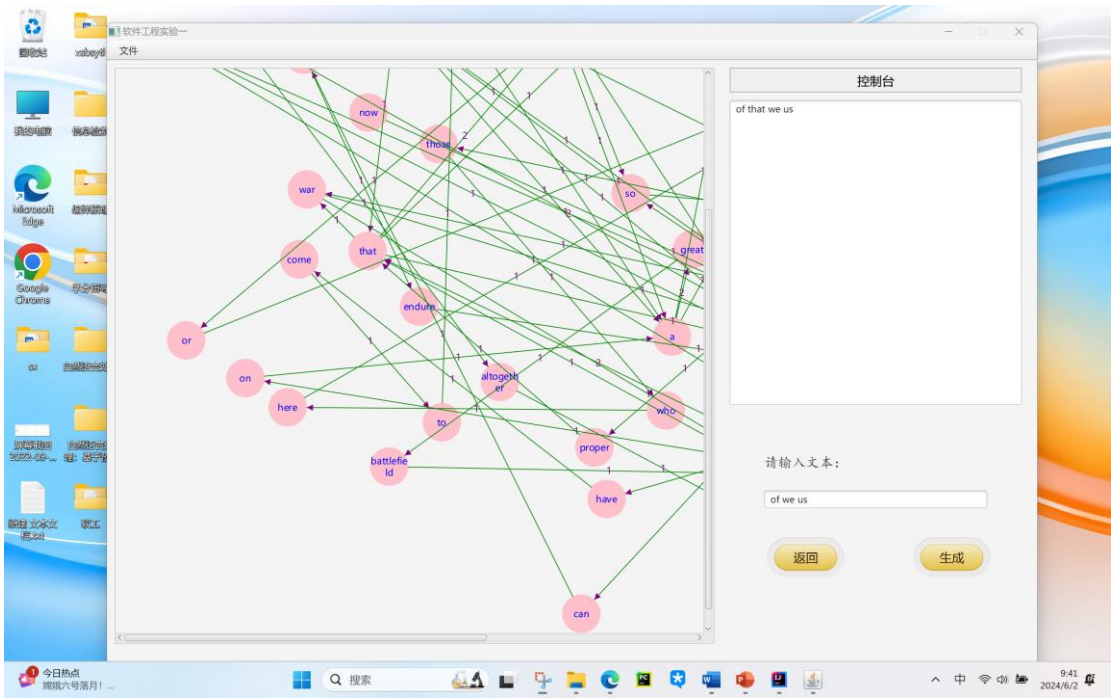


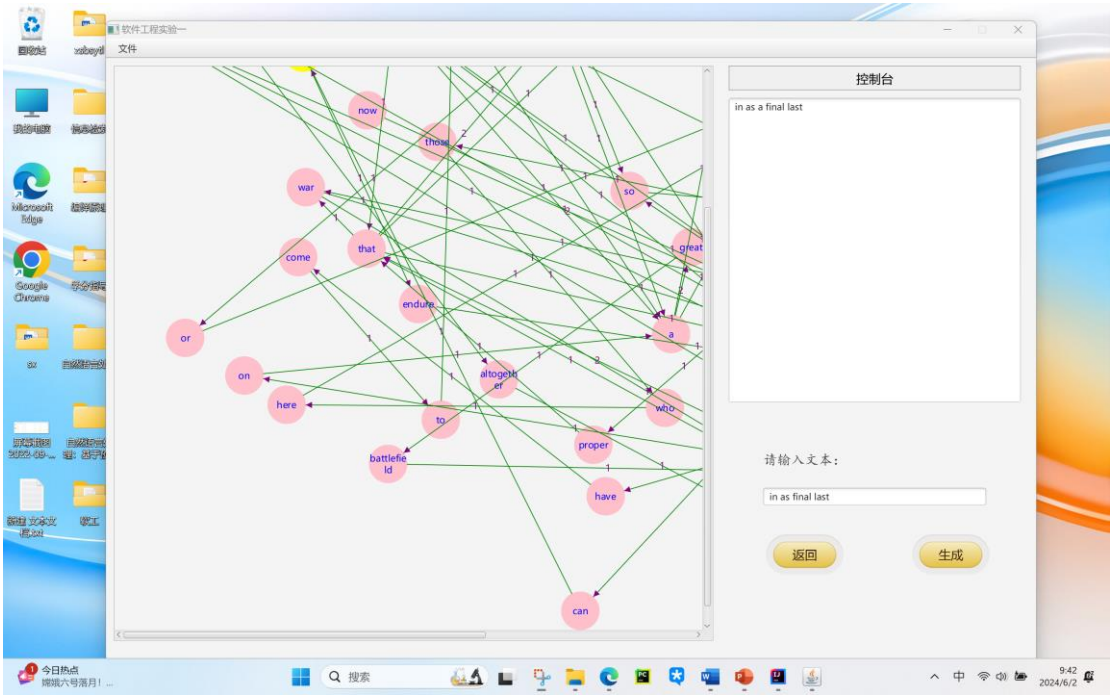
4.3 根据桥接词生成新文本

序号	输入（一行文本）	期望输出	实际输出	运行是否正确
1	is a or an	is a or an	is a or an	正确
2	of we us	of that we us	of that we us	正确
3	in as final last	in as a final last	in as a final last	正确

给出实际运行得到结果的界面截图。


软件工程实验一
文件
控制台
is a or an
请输入文本：
is a or an
返回
生成
now
those
war
so
great
come
that
endure
or
a
on
altogether
here
who
to
proper
battlefield
have
can

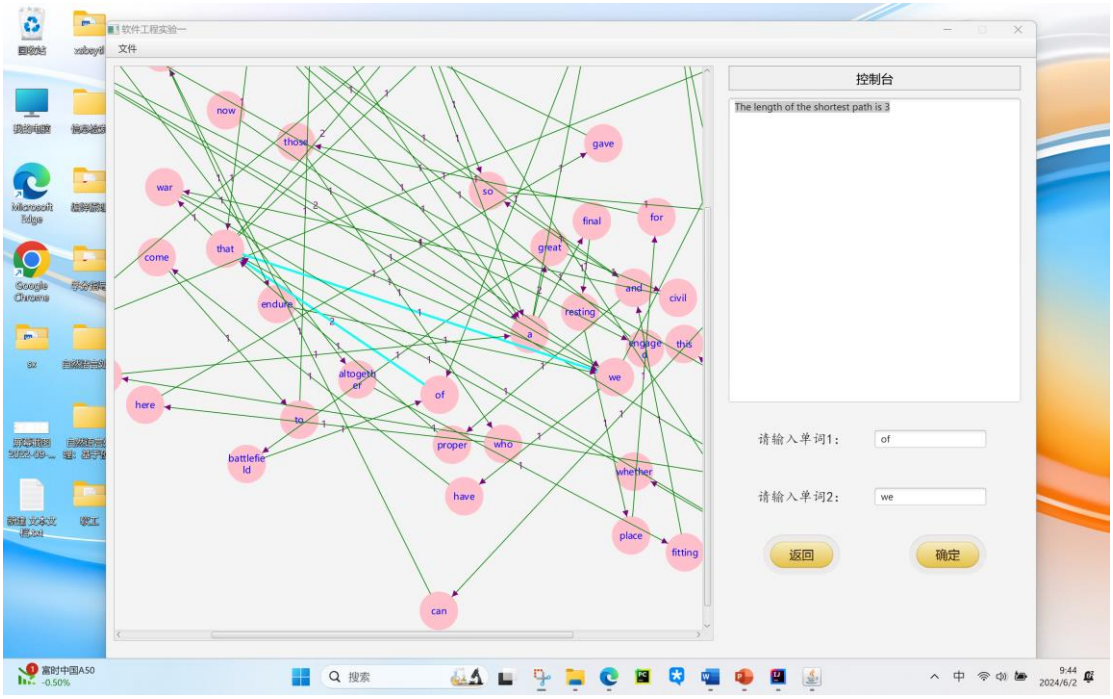

软件工程实验一
文件
控制台
of that we us
请输入文本：
of we us
返回
生成
now
those
war
so
great
come
that
endure
or
a
on
altogether
here
who
to
proper
battlefield
have
can

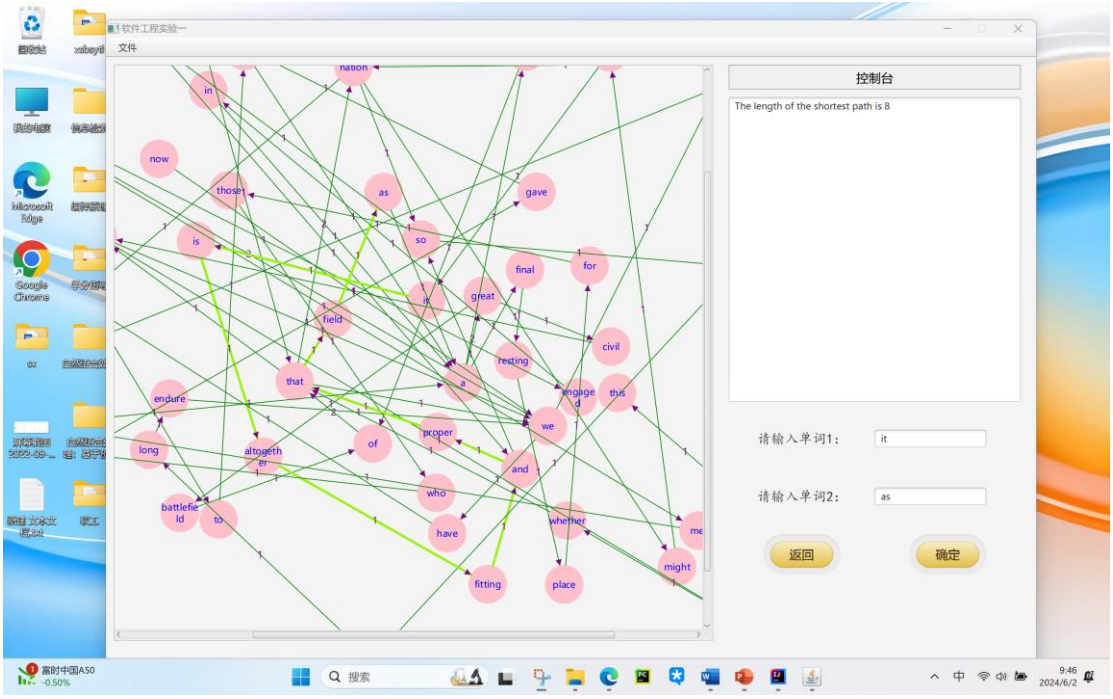


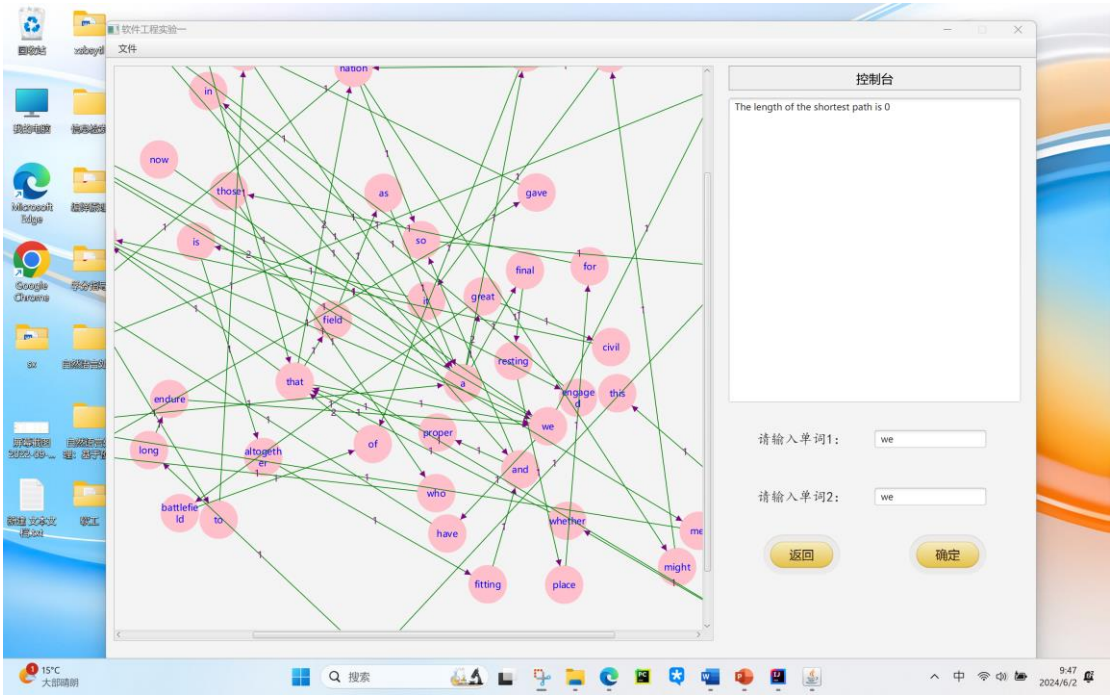
4.4 计算最短路径

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	of we	The length of the shortest path is 3 of -> that -> we	The length of the shortest path is 3 of -> that -> we	正确
2	it as	The length of the shortest path is 8 it -> is -> altogether -> fitting -> and -> proper -> that -> field -> as	The length of the shortest path is 8 it -> is -> altogether -> fitting -> and -> proper -> that -> field -> as	正确
3	we we	The length of the shortest path is 0	The length of the shortest path is 0	正确

给出实际运行得到结果的界面截图。


软件工程实验一
文件
控制台
The length of the shortest path is 3
请输入单词1：
of
请输入单词2：
we
返回
确定
now
those
war
gave
so
final
for
that
come
great
and
civil
endure
resting
a
engage
d
this
altogeth
er
we
here
of
to
proper
who
battlefie
ld
whether
have
place
fitting
can


软件工程实验一
文件
控制台
The length of the shortest path is 8
请输入单词1：
it
请输入单词2：
as
返回
确定
nation
in
now
those
as
gave
is
so
final
for
great
it
field
civil
resting
that
a
endure
engage
d
this
we
long
altogeth
er
of
proper
and
who
battlefie
ld
to
have
whether
might
fitting
place

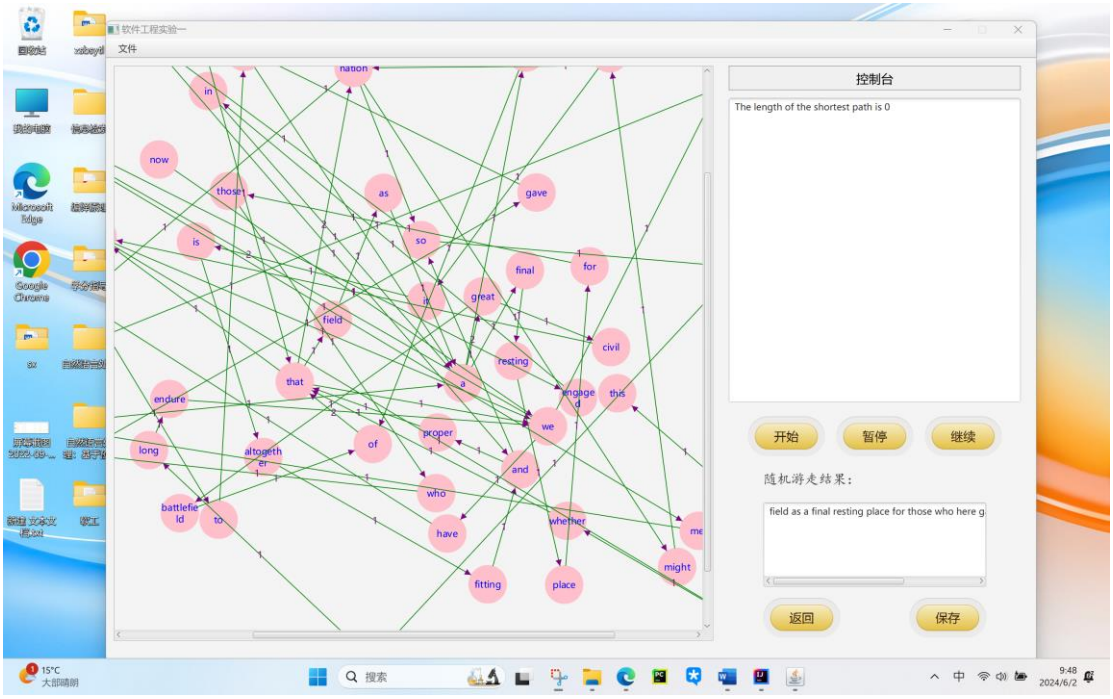


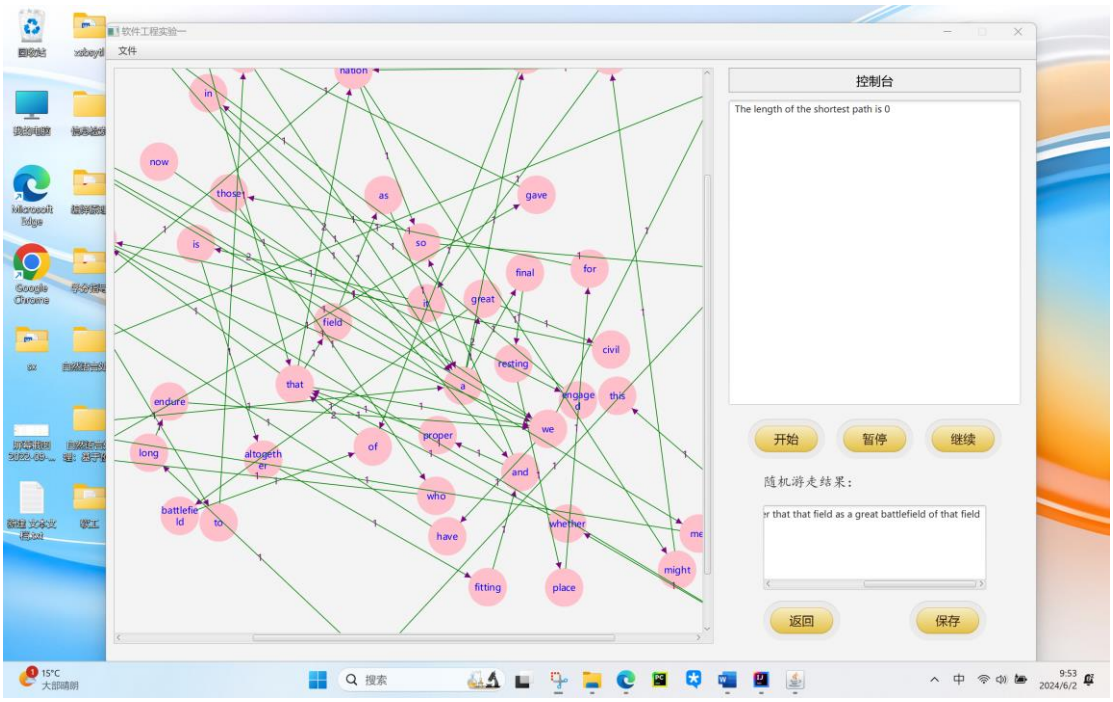
4.5 随机游走

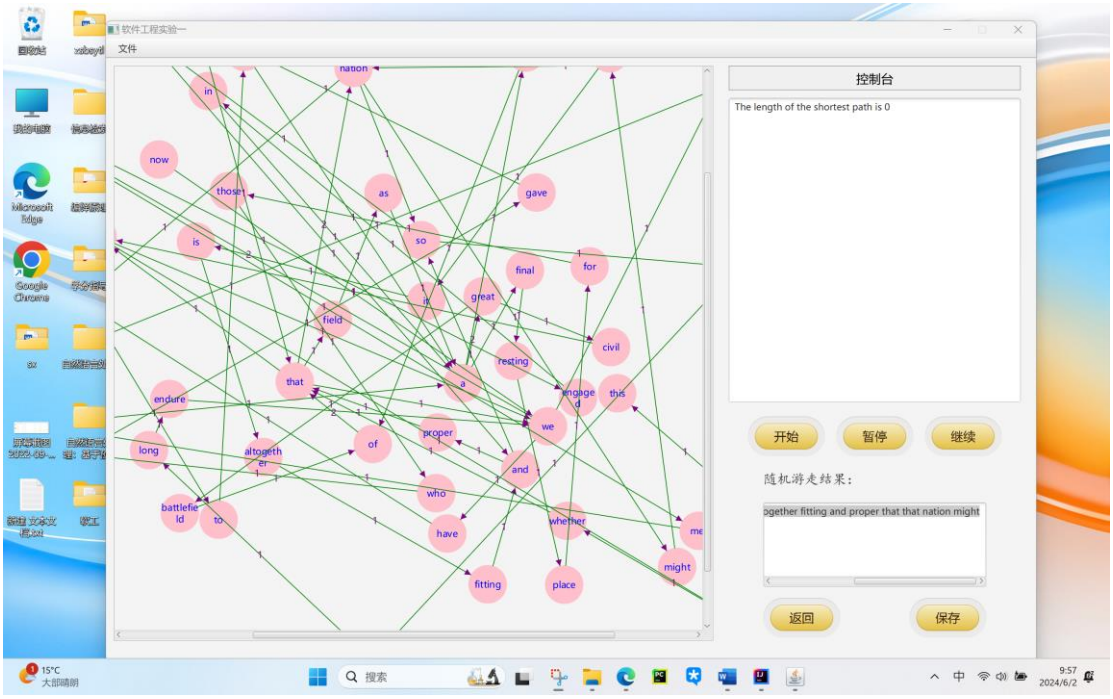
该功能无输入，让你的程序执行多次，分别记录结果。

序号	实际输出	程序运行是否正确
1	field as a final resting place for those who here gave their lives that field as	正确
2	gave their lives that war testing whether that that field as a great battlefield of that field	正确
3	or any nation might live it is altogether fitting and proper that that nation might	正确

给出实际运行得到结果的界面截图。


软件工程实验一
文件
控制台
The length of the shortest path is 0
开始
暂停
继续
随机游走结果：
field as a final resting place for those who here g
返回
保存
nation
in
now
those
as
gave
is
so
final
for
great
it
field
civil
resting
that
a
endure
engage
d
this
we
long
altogeth
er
of
proper
and
who
battlefie
ld
to
have
whether
might
fitting
place


软件工程实验一
文件
控制台
The length of the shortest path is 0
开始
暂停
继续
随机游走结果：
r that that field as a great battlefield of that field
返回
保存
nation
in
now
those
as
gave
is
so
final
for
great
it
field
civil
resting
that
a
endure
engage
d
this
we
long
altogeth
er
of
proper
and
who
battlefie
ld
to
have
whether
might
fitting
place



5 编程语言与开发环境

Java DK 版本：jdk-21、IDE 版本：IntelliJ IDEA 2024.1.1

6 结对编程

6.1 分组依据

郑文翔同学性格阳光开朗，擅长需求的查询与整理，同时较为擅长程序的测试。刘起瑞同学工作较为专注，更擅长实现功能。两位同学可以实现互补。

6.2 角色切换与任务分工

该表格可自行增加更多的行。

日期	时间(HH:MM -- HH:MM)	“驾驶员”	“领航员”	本段时间的任务
5.16	8：00-8：45	郑文翔	刘起瑞	完成 ui
5.16	8：45-9：15	刘起瑞	郑文翔	实现读取文本文件与展示有向图功能
5.16	9：15-9：45	郑文翔	刘起瑞	实现查询桥接词功能
5.23	8：00-8：45	刘起瑞	郑文翔	实现根据桥接词生成新文本功能

5.23	8：45-9：15	郑文翔	刘起瑞	实现计算最短路径功能
5.23	9：15-9：45	刘起瑞	郑文翔	实现随机游走功能

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
5.16	Ui 不理想	查询相关文档，进行修改。	一起查询相关文档
5.23	部分桥接词查询不到	修改代码，调整函数执行顺序。	一起 debug

7 Git 操作过程

7.1 实验场景(1)：仓库创建与提交

各个操作命令：
R0:查看 git 状态：git status
R1:初始化 git 仓库，将项目源文件加入 git 仓库：git init　　git add.
R2:提交所有已添加文件：git commit -m "Initial commit"
R3:查看哪些文件修改以及修改内容:git status git diff
R4:重新提交：git add .　git commit -m " Second commit"
R5:重新提交：git add .　git commit -m " Third commit"
R6:撤销最后一次提交：git reset --soft HEAD~1
R7:查询项目全部提交记录：git log
R8:在 GitHub 上创建远程仓库，在本地建立连接：
`git remote add origin https://github.com/HITLittleZheng/Lab1-2021113211.git`
R9:推送到 GitHub 仓库：`git push -u origin master`

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git status
warning: could not open directory 'System Volume Information/': Permission denie
d
On branch master

No commits yet
```

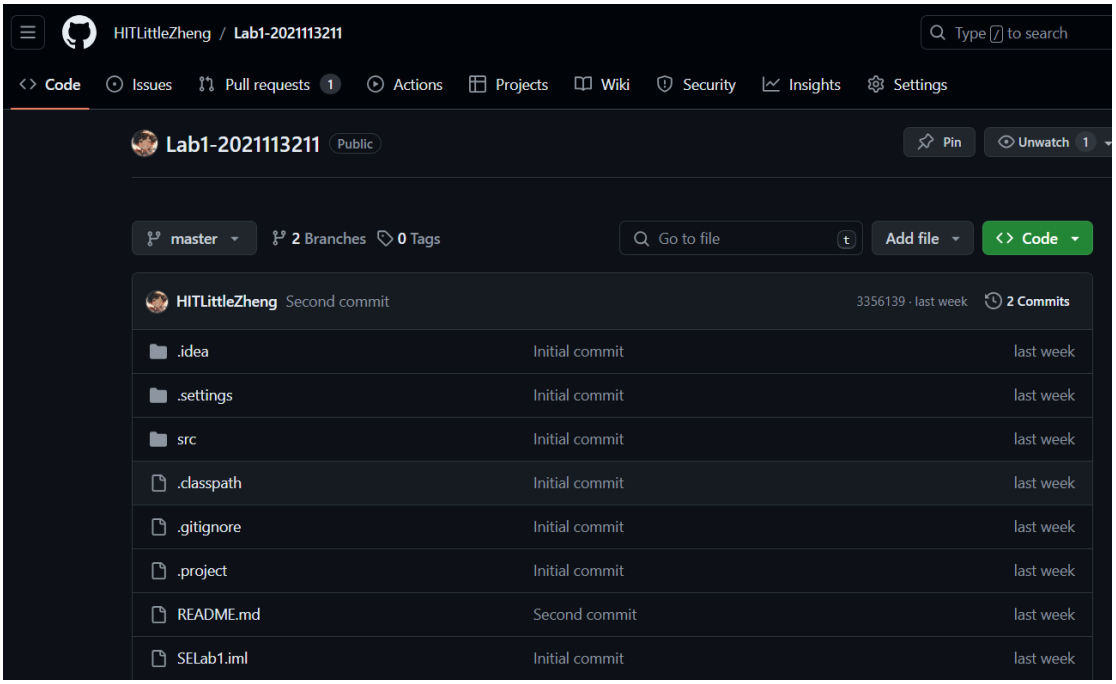
```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git init
Initialized empty Git repository in D:/JavaProject/lab1/.git/

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the ne
xt time Git touches it
warning: in the working copy of 'src/basis/RandomWalker.java', LF will be replac
ed by CRLF the next time Git touches it
warning: in the working copy of 'src/ui/Arrow.java', LF will be replaced by CRLF
 the next time Git touches it
warning: in the working copy of 'src/ui/BaseWindow.css', LF will be replaced by
CRLF the next time Git touches it
warning: in the working copy of 'src/ui/BaseWindow.fxml', LF will be replaced by
 CRLF the next time Git touches it
warning: in the working copy of 'src/ui/GeneratePane.fxml', LF will be replaced
by CRLF the next time Git touches it
warning: in the working copy of 'src/ui/ImageWindow.fxml', LF will be replaced b
y CRLF the next time Git touches it
warning: in the working copy of 'src/ui/PathPane.fxml', LF will be replaced by C
RLF the next time Git touches it
warning: in the working copy of 'src/ui/PointBox.java', LF will be replaced by C
RLF the next time Git touches it
warning: in the working copy of 'src/ui/QueryPane.fxml', LF will be replaced by
CRLF the next time Git touches it
warning: in the working copy of 'src/ui/SubPane.css', LF will be replaced by CRL
F the next time Git touches it
warning: in the working copy of 'src/ui/WalkPane.fxml', LF will be replaced by C
RLF the next time Git touches it
warning: in the working copy of 'test1.txt', LF will be replaced by CRLF the nex
t time Git touches it
```

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git commit -m "Initial commit"
[master (root-commit) b32a637] Initial commit
 31 files changed, 1932 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .gitignore
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/.name
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 README.md
 create mode 100644 SELab1.iml
 create mode 100644 lab1.jar
 create mode 100644 lab1.png
 create mode 100644 src/basis/DirectedGraph.java
 create mode 100644 src/basis/GraphProcessor.java
 create mode 100644 src/basis/RandomWalker.java
 create mode 100644 src/basis/Vertex.java
 create mode 100644 src/ui/Arrow.java
 create mode 100644 src/ui/BaseWindow.css
 create mode 100644 src/ui/BaseWindow.fxml
 create mode 100644 src/ui/BaseWindowController.java
 create mode 100644 src/ui/GeneratePane.fxml
 create mode 100644 src/ui/ImageWindow.fxml
 create mode 100644 src/ui/MainApplication.java
 create mode 100644 src/ui/PathPane.fxml
 create mode 100644 src/ui/PointBox.java
 create mode 100644 src/ui/QueryPane.fxml
 create mode 100644 src/ui/SubPane.css
 create mode 100644 src/ui/WalkPane.fxml
 create mode 100644 test.txt
 create mode 100644 test1.txt
```

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ ls
README.md  SELab1.iml  bin/  lab1.jar  lab1.png  src/  test.txt  test1.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ vi README.md

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git diff
diff --git a/README.md b/README.md
index d5785a3..9d472ba 100644
--- a/README.md
+++ b/README.md
@@ -1,4 +1,4 @@
-# 代码说明
+# 代码的说明
 > 哈尔滨工业大学《软件工程》2017年秋季实验1代码

 # 效果展示
```

```
86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git add .

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git commit -m "Second commit"
[master 3356139] Second commit
 1 file changed, 1 insertion(+), 1 deletion(-)

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ vi README.md

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git add .

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git commit -m "Third commit"
[master 6a910dd] Third commit
 1 file changed, 1 insertion(+), 1 deletion(-)

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git reset --soft HEAD~1

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git log
commit 33561397eb94830b2abfe2b810cf3fb96b4e0f02 (HEAD -> master)
Author: HITLittleZheng <3216234948@qq.com>
Date:   Thu May 23 08:33:45 2024 +0800

    Second commit

commit b32a637087a05943787672d5bea12c1982390bd2
Author: HITLittleZheng <3216234948@qq.com>
Date:   Thu May 23 08:29:06 2024 +0800

    Initial commit
```

```
86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git remote add origin https://github.com/HITLittleZheng/Lab1-2021113211.git
error: remote origin already exists.

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git push -u origin master
Enumerating objects: 41, done.
Counting objects: 100% (41/41), done.
Delta compression using up to 16 threads
Compressing objects: 100% (39/39), done.
Writing objects: 100% (41/41), 109.78 KiB | 12.20 MiB/s, done.
Total 41 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/HITLittleZheng/Lab1-2021113211.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```



7.2 实验场景(2)：分支管理

各个操作命令：

R1:获得本地 Lab1 仓库的全部分支，切换至分支 master：git branch git checkout master

R2:在 master 基础上建立两个分支 B1、B2：

```
git checkout B1
git checkout B2
```

R3:在 B2 分支基础上创建一个新分支 C4:git checkout -b C4

R4:在 C4 上，对 2 个文件进行修改并提交：

```
git add REAME.md test.txt
git commit -m "修改 C4 上的文件"
```

R5:在 B1 分支上对同样的 2 个文件做不同修改并提交：

```
git checkout B1
git add REAME.md test.txt
git commit -m "修改 B1 上的文件"
```

R6:将 C4 合并到 B1 分支，若有冲突，手工消解：

```
git merge C4
git add REAME.md test.txt
git commit -m "解决合并冲突"
```

R7:在 B2 分支上对 2 个文件做修改并提交：

```
git checkout B2
git add REAME.md test.txt
git commit -m "修改 B2 上的文件"
```

R8:查看目前哪些分支已经合并、哪些分支尚未合并：

```
git branch --merged
```

git branch --no-merged

R9:将已经合并的分支删除，将尚未合并的分支合并到一个新分支上， 分支名字为你的学号：

git branch -d C4

git checkout -b 2021113211

git merge B2

R10：将本地以你的学号命名的分支推送到 GitHub 上自己的仓库内

git push origin 2021113211

R11：查看完整的版本变迁树

git log --graph --all --decorate

R12：在 Github 上以 web 页面的方式查看你的 Lab1 仓库的当前状态

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch
* master

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git checkout master
Already on 'master'
M       README.md
Your branch is up to date with 'origin/master'.

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch B1

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch B2

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (master)
$ git checkout B2
Switched to branch 'B2'
M       README.md

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B2)
$ git branch C4

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B2)
$ git checkout C4
Switched to branch 'C4'
M       README.md
```

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ ls
README.md  SELab1.iml  bin/  lab1.jar  lab1.png  src/  test.txt  test1.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ vi README.md

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ vi test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ git add README.md test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ git commit -m "修改C4上的文件"
[C4 6cb3c8f] 修改C4上的文件
 2 files changed, 2 insertions(+), 2 deletions(-)
```

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (C4)
$ git checkout B1
Switched to branch 'B1'

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ vi README.md

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ vi test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ git add README.md test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ git commit -m "修改B1上的文件"
[B1 1a5d8bd] 修改B1上的文件
 2 files changed, 2 insertions(+), 2 deletions(-)
```

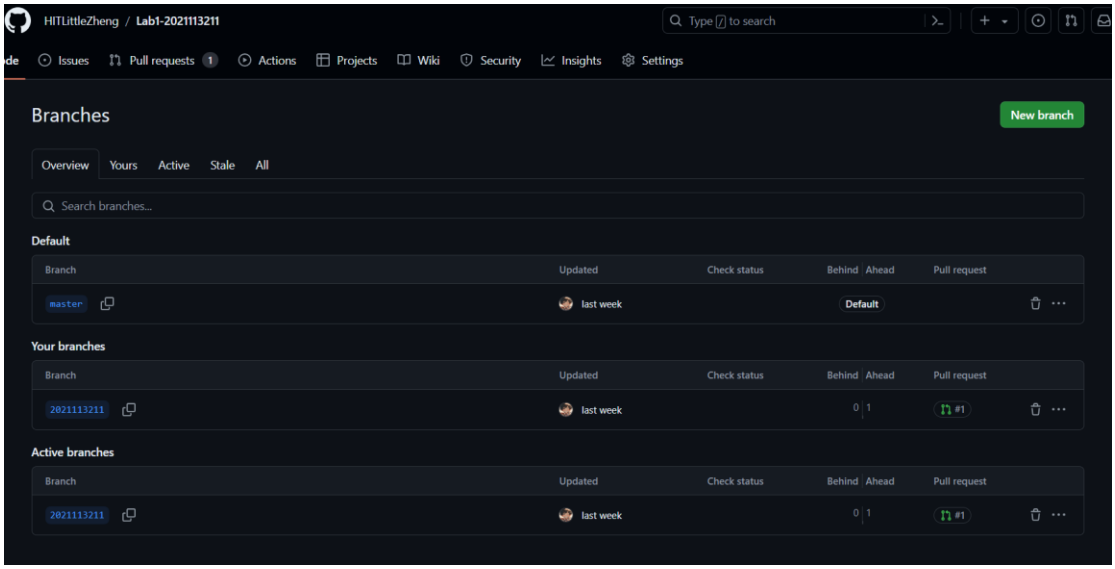
```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ git merge C4
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1|MERGING)
$ git checkout B2
error: you need to resolve your current index first
README.md: needs merge
test.txt: needs merge

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1|MERGING)
$ vi README.md

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1|MERGING)
$ vi test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1|MERGING)
$ git add README.md test.txt

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1|MERGING)
$ git commit -m "解决合并冲突"
[B1 d36fdcb] 解决合并冲突

86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (B1)
$ git merge C4
Already up to date.
```

```
Switched to branch 'B1'

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (B1)
$
git branch --merged
* B1
  C4
  master

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (B1)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch --no-merged
  B1
  B2
  C4

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$  git branch --merged
* master

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch -d B1
error: the branch 'B1' is not fully merged.
If you are sure you want to delete it, run 'git branch -D B1'

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git branch -D B1
Deleted branch B1 (was d36fdcb).

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$
```

```
86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (master)
$ git checkout -b  2021113211
Switched to a new branch '2021113211'

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (2021113211)
$ git merage B2
git: 'merage' is not a git command. See 'git --help'.

The most similar command is
        merge

86132@DESKTOP-AH40CSG MINGW64 /d/JavaProject/lab1 (2021113211)
$ git merge B2
Updating 3356139..23383f8
Fast-forward
 README.md | 2 +-
 test.txt  | 2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)
```

```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (2021113211)
$ git push -u origin 2021113211
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 440 bytes | 440.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for '2021113211' on GitHub by visiting:
remote:      https://github.com/HITLittleZheng/Lab1-2021113211/pull/new/20211132
11
remote:
To https://github.com/HITLittleZheng/Lab1-2021113211.git
 * [new branch]      2021113211 -> 2021113211
branch '2021113211' set up to track 'origin/2021113211'.
```

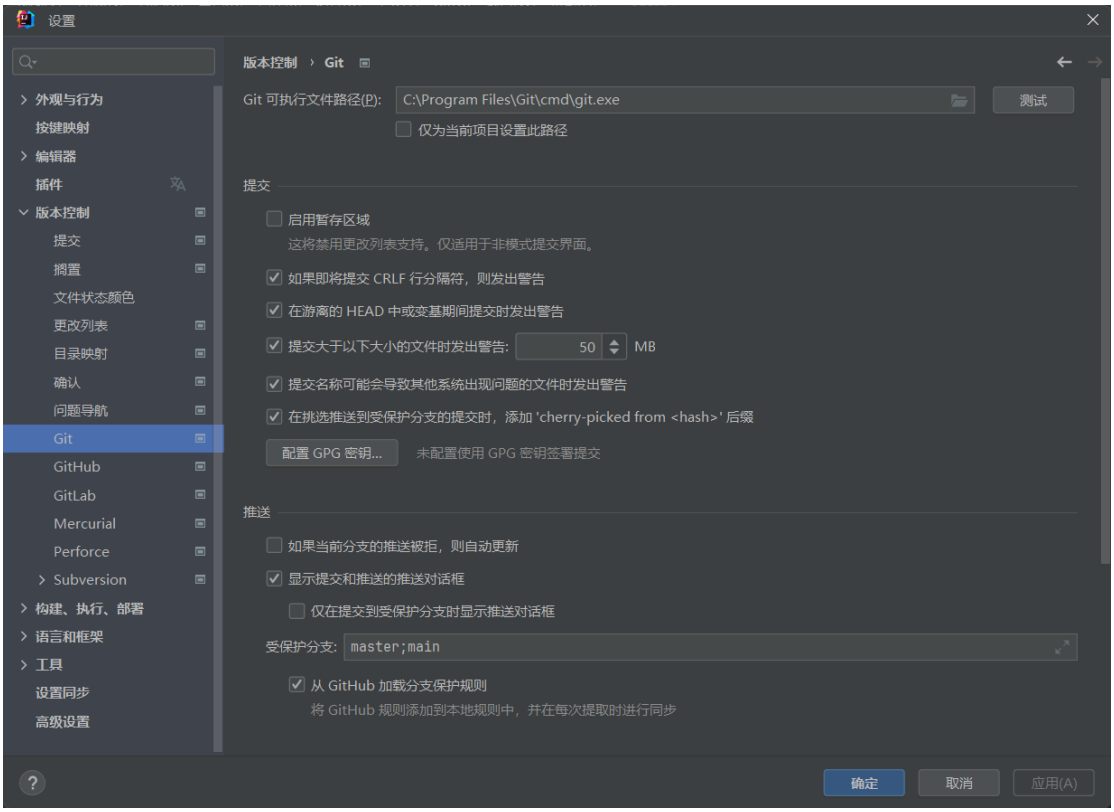
```
86132@DESKTOP-AH4OCSG MINGW64 /d/JavaProject/lab1 (2021113211)
$ git log --graph --all --decorate
* commit 23383f894c1203c371009d70dd76781c6e22ac95 (HEAD -> 2021113211, origin/20
21113211, B2)
| Author: HITLittleZheng <3216234948@qq.com>
| Date:   Thu May 23 09:30:00 2024 +0800
|
|     修改B2上的文件
|
| * commit 6cb3c8fbcce5adc71023425e6934c914826fcc33 (C4)
|/  Author: HITLittleZheng <3216234948@qq.com>
|   Date:   Thu May 23 09:19:04 2024 +0800
|
|       修改C4上的文件
|
* commit 33561397eb94830b2abfe2b810cf3fb96b4e0f02 (origin/master, master)
| Author: HITLittleZheng <3216234948@qq.com>
| Date:   Thu May 23 08:33:45 2024 +0800
|
|     Second commit
|
* commit b32a637087a05943787672d5bea12c1982390bd2
  Author: HITLittleZheng <3216234948@qq.com>
  Date:   Thu May 23 08:29:06 2024 +0800

:...skipping...
* commit 23383f894c1203c371009d70dd76781c6e22ac95 (HEAD -> 2021113211, origin/2021113211, B2)
| Author: HITLittleZheng <3216234948@qq.com>
| Date:   Thu May 23 09:30:00 2024 +0800
|
|     修改B2上的文件
|
| * commit 6cb3c8fbcce5adc71023425e6934c914826fcc33 (C4)
|/  Author: HITLittleZheng <3216234948@qq.com>
|   Date:   Thu May 23 09:19:04 2024 +0800
|
|       修改C4上的文件
|
* commit 33561397eb94830b2abfe2b810cf3fb96b4e0f02 (origin/master, master)
| Author: HITLittleZheng <3216234948@qq.com>
| Date:   Thu May 23 08:33:45 2024 +0800
|
|     Second commit
|
* commit b32a637087a05943787672d5bea12c1982390bd2
  Author: HITLittleZheng <3216234948@qq.com>
  Date:   Thu May 23 08:29:06 2024 +0800

      Initial commit
```



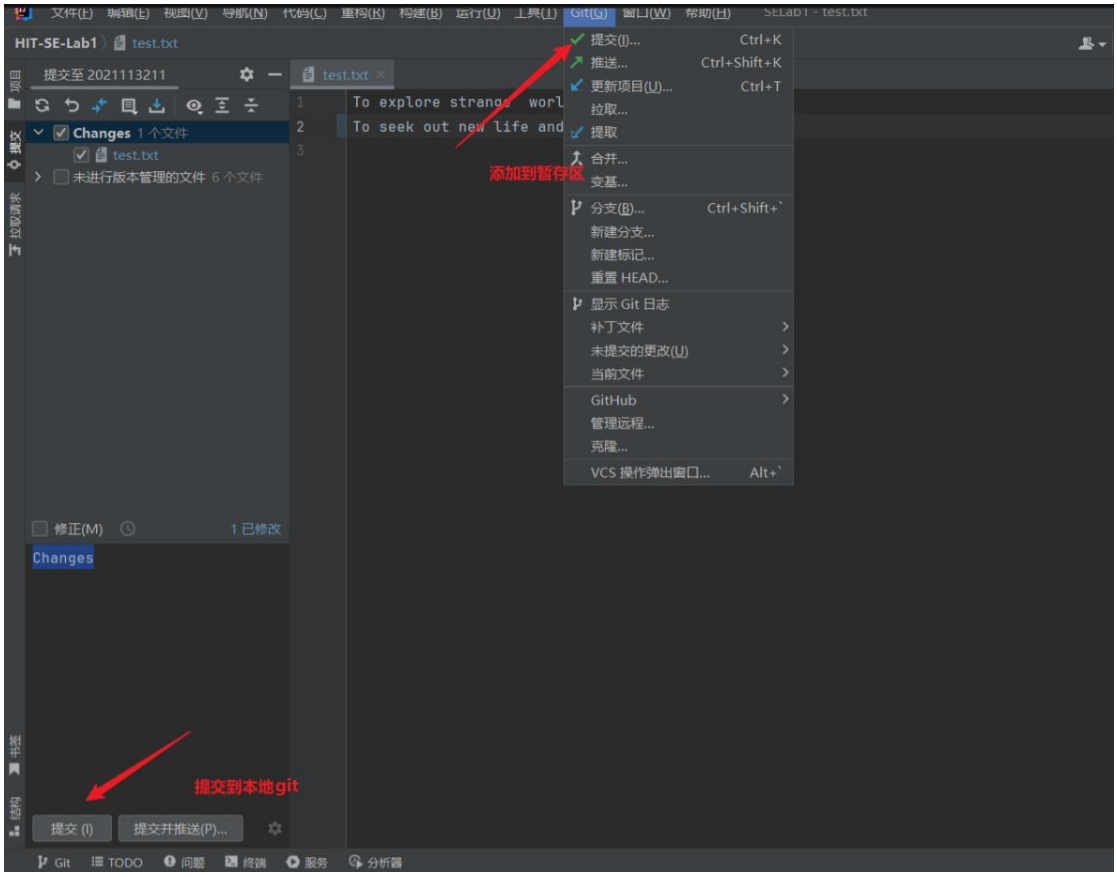
8 在 IDE 中使用 Git Plugin

我使用的 IDE 是 IntelliJ IDEA 2024.1.1：

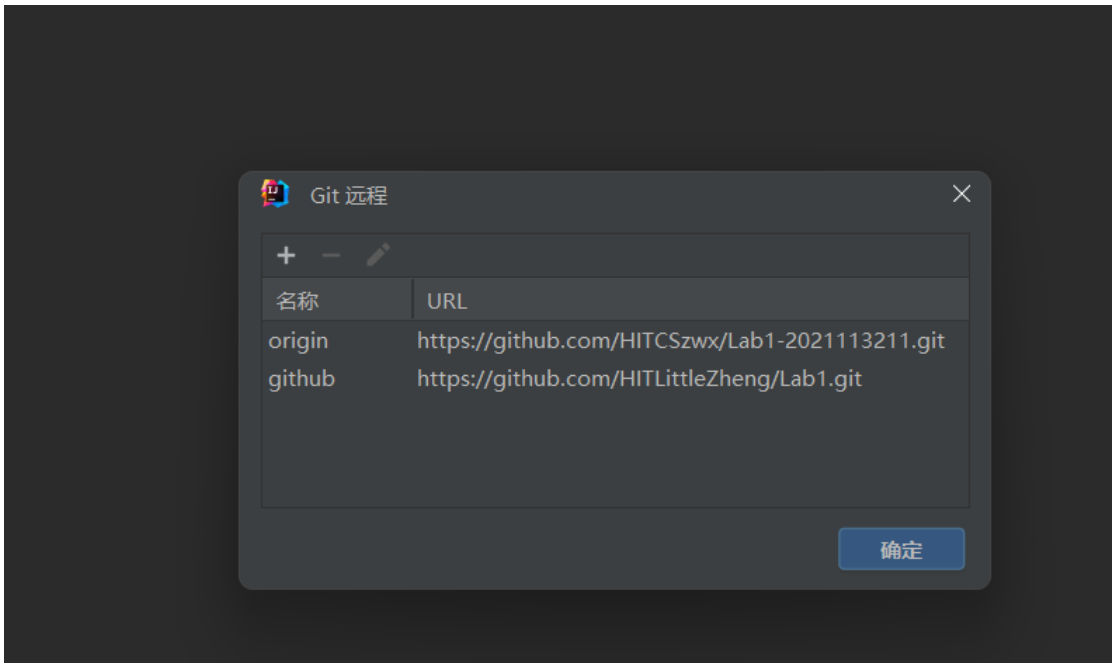
1.将 lab1 纳入 Git 管理，设置好 git 配置。通过 VCS 中将文件目录建成 git 仓库即可：

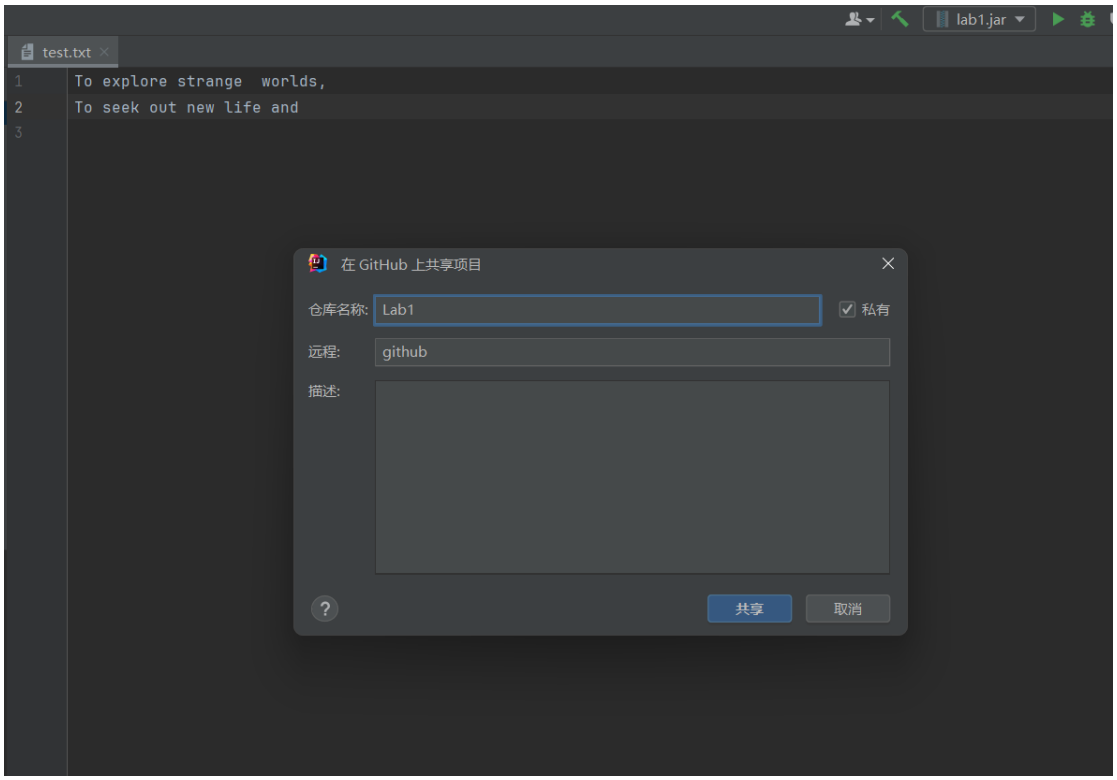


2.将 lab1 修改后提交到本地 git：修改代码后，点击 IDEA 中 git 的对号按钮，将文件添加到暂存区，之后点击左侧出现的 commit，将暂存区文件进行提交：



3.将 lab1 修改后提交到 github 仓库





9 小结

体验总结：在这次实验中，我首次尝试了与同伴进行结对编程，深刻领会到敏捷开发模式下双人协作的不可或缺性。这让我认识到，在软件开发的浩瀚工程里，个人的贡献虽不可或缺，但团队的合力才是推动项目前进的强大力量。通过这一过程，我的 Java 编程技能得到了实质性的提升，更重要的是，我学会了如何在协作框架下更高效地达成目标，超越了个人单打独斗的局限。此外，我的团队合作、沟通交流及共同探索问题解决方案的能力也有了显著提高。实验期间，我还掌握了 git 这一版本控制工具的基础应用，深切体会到它为团队协作带来的便利，为将来在更多软件开发项目中有效利用 git 及其他高级工具奠定了坚实基础。

改进建议：为了进一步提升实验效率与成果质量，建议未来实验的需求说明能更加详尽精确。在我们实践过程中，部分任务需求的模糊性导致了解决方案的不确定性和多样性，清晰无误的需求表述将有助于减少误解，确保所有参与者都能朝着同一明确目标努力。