Lab 1：结对编程与Git实战

实验目标

- **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

结对编程部分

开发任务

- 开发一个程序，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。
- 开发的程序可以是命令行方式运行，也可以用图形化用户界面GUI的方式运行。无论何种方式，均应覆盖后续所有功能需求。

输入：文本文件

- 输入一个文本文件，其中包含用英文书写的文本数据；
- 文本分为多行，你的程序应默认将换行/回车符当作空格；
- 文本中的任何标点符号，也应当作空格处理；
- 文本中的非字母(A-Z, a-z)字符应被忽略。

- 例如：

To @ explore strange new worlds,

To seek out new life and new civilizations?

- 等价于 **to explore strange new worlds to seek out new life and new civilizations**

功能需求1：读入文本并生成有向图

- **程序首先让用户选择或输入文本文件的位置和文件名。也可以参数的形式，在启动程序时提供文件路径和文件名。**
- **程序读入文本数据，进行分析，将其转化为有向图：**
  - 有向图的节点为文本中包含的某个单词（不区分大小写）
  - 两个节点A,B之间存在一条边A→B，意味着在文本中至少有一处位置A和B相邻出现（即A和B之间有且仅有1或多个空格）。
  - A→B的权重w=文本中A和B相邻出现的次数，w>=1。

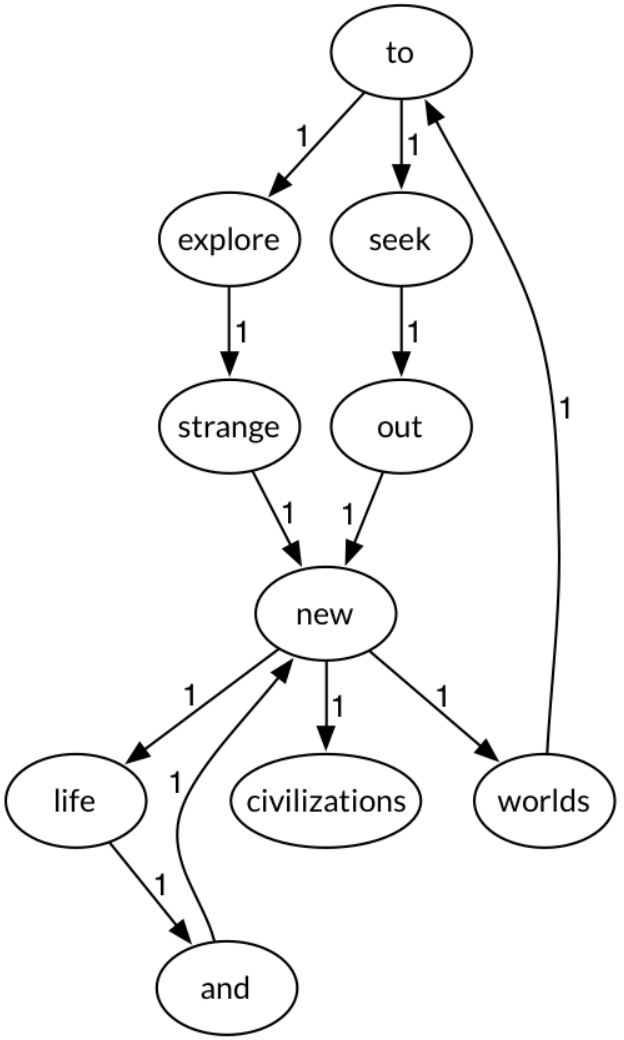
小例子

- 输入的文本文件：

To explore strange new worlds,

To seek out new life and new civilizations

- 生成的有向图：



功能需求2：展示有向图

- 展示生成的有向图。

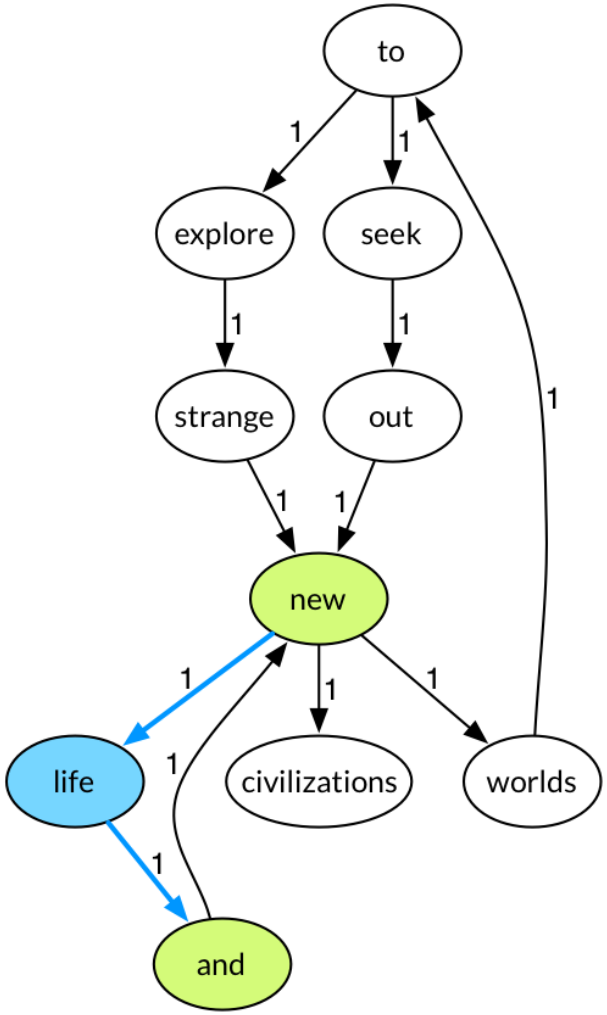
- 可选功能：将生成的有向图以图形文件形式保存到磁盘，可以调用外部绘图库或绘图工具API自动生成有向图，但不能采用手工方式绘图。

功能需求3：查询桥接词（bridge words）

- 在生成有向图之后，用户输入任意两个英文单词word1、word2，程序从图中查询它们的“桥接词”。
- word1、word2的桥接词word3：图中存在两条边word1→word3, word3→word2。
- 输入的word1或word2如果不在图中出现，则输出“No word1 or word2 in the graph!”
- 如果不存在桥接词，则输出“No bridge words from word1 to word2!”
- 如果存在一个或多个桥接词，则输出“The bridge words from word1 to word2 are: xxx, xxx, and xxx.”

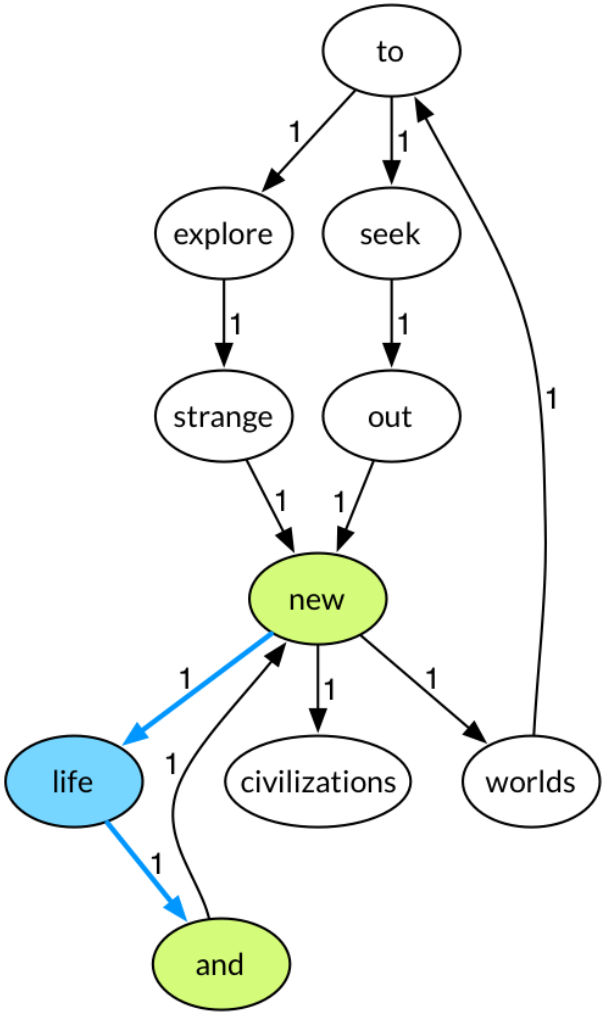
例子

$word_1$	$word_2$	Output
seek	to	No bridge words from ”seek” to ”to”!
to	explore	No bridge words from ”to” to ”explore”!
explore	new	The bridge words from ”explore” to ”new” is: strange
new	and	The bridge words from ”new” to ”and” is: life
and	exciting	No “exciting” in the graph!
exciting	synergies	No “exciting” and “synergies” in the graph!



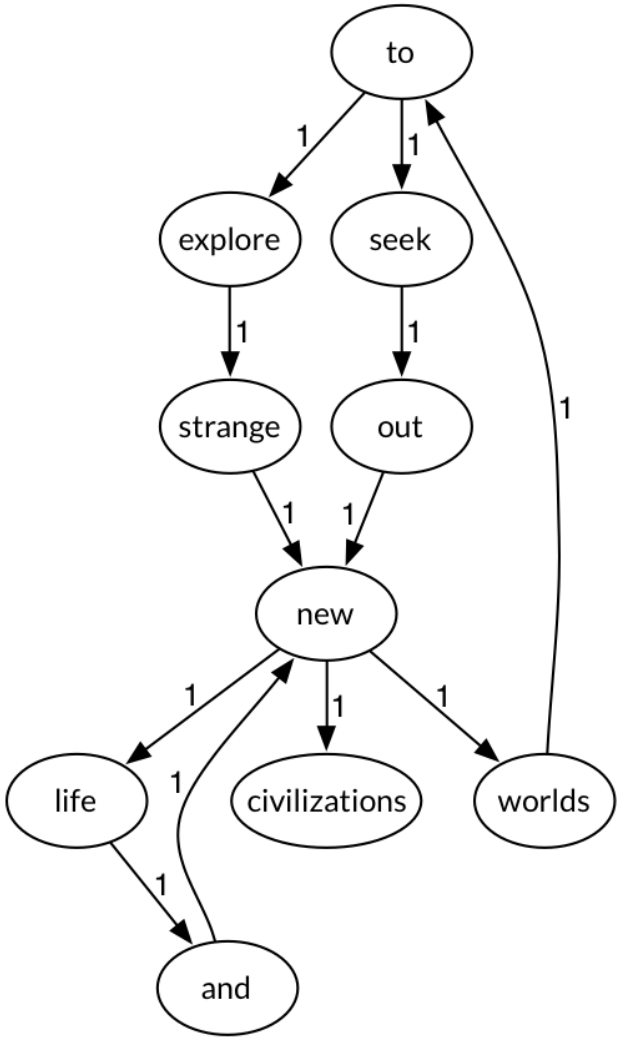
功能需求4：根据bridge word生成新文本

- **用户输入一行新文本，程序根据之前输入文件生成的图，计算该新文本中两两相邻的单词的bridge word，将bridge word插入新文本的两个单词之间，输出到屏幕上展示。**
  - 如果两个单词无bridge word，则保持不变，不插入任何单词；
  - 如果两个单词之间存在多个bridge words，则随机从中选择一个插入进去形成新文本。
- **例如用户输入：Seek to explore new and exciting synergies**
- **则输出结果为：Seek to explore strange new life and exciting synergies**



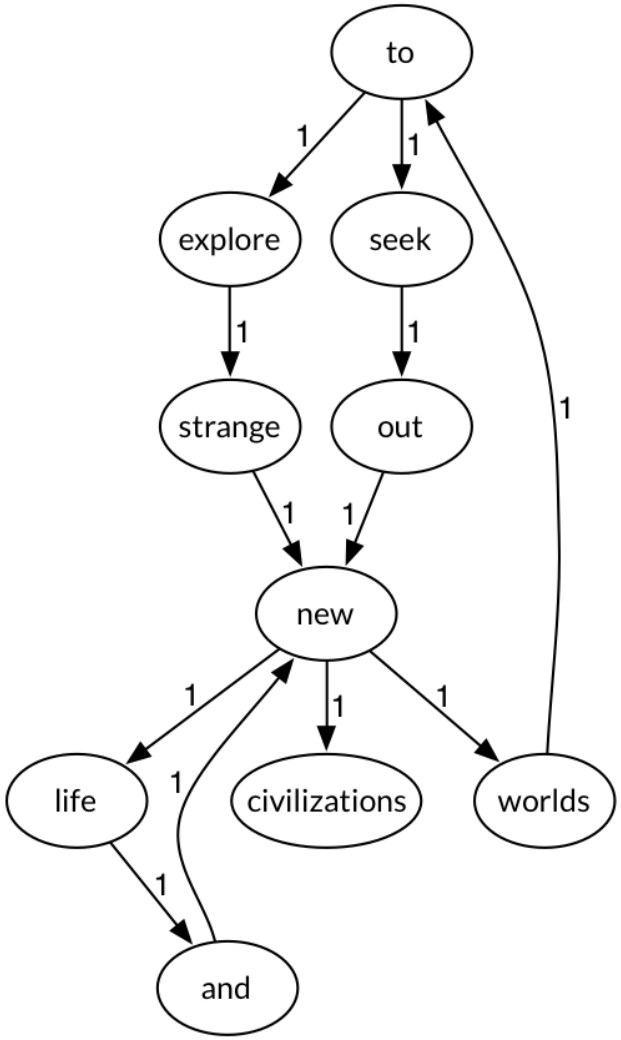
功能需求5：计算两个单词之间的最短路径

- 用户输入两个单词，程序计算它们之间在图中的最短路径（路径上所有边权值之和最小），以某种突出的方式将路径标注在原图并展示在屏幕上，同时展示路径的长度（所有边权值之和）。
  - 例如：输入to和and，则其最短路径为 to→explore→strange→new→life→and
- 如果有多条最短路径，只需要展示一条即可。
  - 可选：计算出所有的最短路径，并以不同的突出显示方式展示出来。
  - 例如to和and之间还有另一条路径：to→seek→out→new→life→and。
- 如果输入的两个单词“不可达”，则提示。
- 可选功能：如果用户只输入一个单词，则程序计算出该单词到图中其他任一单词的最短路径，并逐项展示出来。



功能需求6：随机游走

- 进入该功能时，程序随机的从图中选择一个节点，以此为起点沿出边进行随机遍历，记录经过的所有节点和边，直到出现第一条重复的边为止，或者进入的某个节点不存在出边为止。在遍历过程中，用户也可随时停止遍历。
- 将遍历的节点输出为文本，并以文件形式写入磁盘。
- 例如：
  - to seek out new life and new worlds to explore strange new civilizations
  - to explore strange new worlds to explore



实验要求

- **提交一个.java文件，其中至少包含以下函数：**
  - `main(…)`：主程序入口，接收用户输入文件，生成图，并允许用户选择后续各项功能；
  - `void showDirectedGraph(type G, …)`：展示有向图
  - `String queryBridgeWords(String word1, String word2)`：查询桥接词
  - `String generateNewText(String inputText)`：根据bridge word生成新文本
  - `String calcShortestPath(String word1, String word2)`：计算两个单词之间的最短路径
  - `String randomWalk()`：随机游走

实验要求

- **除了`main()`之外，上述其他函数应尽可能保持与用户输入/系统输出的独立性（所有输入输出均应在main函数中完成；如果采用GUI，则在GUI框架中完成）；**
- **不能改变函数的specification（参数列表/类型、返回值类型、函数名）；**
  - 例外1：函数`void showDirectedGraph(type G,…)`的输入参数`G`的类型`type`，由开发者自行定义；可根据需要增加其他参数。
  - 例外2：函数`main(String[] args)`的输入参数个数与具体含义由开发者自定义。
- **必要时可增加其他辅助函数，但须在实验报告中列清楚各函数的作用；**
- **避免使用任何第三方Java外部算法库完成上述功能。**

实验评判标准

- 结果的正确性
- 健壮性
- 算法执行时间
- 代码质量

- 结对编程过程中两人的配合度
- 遵循实验报告模板撰写，格式规范美观

- 对可选需求的支持程度（附加分）

Git实战

实验目标

- **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。

在本地机器上安装Git

- 下载Git客户端安装包https://git-scm.com/download，在本地机器安装
- 在Github/或其他基于Git的代码托管平台上申请个人账号，作为Git远程服务器：Github(http://www.github.com)
- 关于Git的学习手册：https://git-scm.com/book/en/v2（英文）、https://git-scm.com/book/zh/v2（中文）。可以使用https://learngitbranching.js.org 提供的在线环境进行Git命令练习。
- 使用命令行方式完成实验，避免图形界面下的操作。

实验场景(1)：仓库创建与提交

- **R0：在进行每次git操作之前，随时查看工作区、暂存区、git仓库的状态，确认项目里的各文件当前处于什么状态；**
- **R1：本地初始化一个git仓库，将自己在Lab1中所创建项目的全部源文件加入进去，纳入git管理；**
- **R2：提交；**
- **手工对提交的部分文件进行修改；**
- **R3：查看上次提交之后都有哪些文件修改、具体修改内容是什么；**
- **R4：重新提交；**
- **再次对部分文件进行修改；**
- **R5：重新提交**
- **R6：把最后一次提交撤销；**
- **R7：查询该项目的全部提交记录；**

实验场景(1)：推送到GitHub上

R8：在GitHub上创建名为“Lab1-学号”的仓库，并在本地仓库建立相应的远程仓库；

R9：将之前各步骤得到的本地仓库全部内容推送到GitHub的仓库中；

实验场景(2)：分支管理

- R1：获得本地Lab1仓库的全部分支，切换至分支master；
- R2：在master基础上建立两个分支B1、B2；
- R3：在B2分支基础上创建一个新分支C4；
- R4：在C4上，对2个文件进行修改并提交；
- R5：在B1分支上对同样的2个文件做不同修改并提交；
- R6：将C4合并到B1分支，若有冲突，手工消解；
- R7：在B2分支上对2个文件做修改并提交；
- R8：查看目前哪些分支已经合并、哪些分支尚未合并；
- 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；
- 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；
- R11：查看完整的版本变迁树；
- R12：在Github上以web页面的方式查看你的Lab1仓库的当前状态。

实验场景3：在IDE中使用Git管理程序

1. 在IDE中，将自己的Lab1纳入Git管理；
2. 对Lab1进行若干修改，对其进行本地仓库提交操作；
3. 将Lab1内容推送至个人GitHub的Lab1仓库。

提交与检查方式

- 提交日期：第13周周日晚(6月2日 23:55)
- 提交实验报告到头歌平台：
  - 实验报告：命名规则“学号-lab1-report.doc”
- 同组内的两人均要提交，文件命名不同，结对编程部分内容相同、Git实战部分内容不同。

- 检查方式：
  - 第11、12、13周实验课上，随时请实验教师和TA现场检查程序演示和代码，并现场打分；
  - 第13周提交实验报告后，TA对实验报告进行打分。
  - 如果在实验课上未能与TA现场检查，则以TA对实验报告的打分为准（基于文字的打分，可能不准确和不全面，请理解）。

结束