软件工程实验一 项目架构

一 需要完成的功能以及初步设计

1.1 功能需求

1. 读入文本并生成有向图
   1. 需要注意边权值：A→B的权重w=文本中A和B相邻出现的次数， w>=1
2. 展示有向图
   1. 无UI：对于每条边A→B，在控制台上输出” A -> B”，每条边占一行
   2. 有UI：使用GraphStream绘图并展示
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
   1. 无UI：对于最短路”A->B->C->D”，输出”A->B”、”B->C”、”C->D”，每个输出占一行
   2. 有UI：将最短路的边用不同的颜色突出显示
   3. 可选功能：只输入一个单词并计算到任意单词的最短路
      1. 无UI：对于每个单词，按照5.a的格式输出，不同单词之间用”\*\*\*\*\*\*\*\*\*\*”作为分割
      2. 有UI：到不同单词的最短路用不同的颜色进行区分
6. 随机游走
   1. 用户应能随时停止

1.2 初步设计

1.2.1 有向图的存储

采用邻接表的方式进行存储。同时，为每个单词分配一个index，设置单词到index和index到单词的两个映射，邻接表的节点表示用index。

1.2.2 文本到有向图

需要注意，换行、标点符号当作空格，非字母要忽略。Gpt给出的参考代码如下：



1.2.3 有向图的展示

* 无UI：遍历邻接表，查找对应index的单词，并成对输出。
* 有UI：使用GraphStream生成有向图，并展示。

1.2.4 桥接词查询

采用邻接表存储，所以当查询单词A和单词B的桥接词时

1. 查询A和B对应的index，IA和IB
2. 根据IA查找A的后继IA\_next
3. 对每个IA\_next，查询其后继是否有IB
4. 如果有则将IA\_next对应的单词加入结果集合中
5. 如果没有则返回2直到A的所有后继都被遍历完
6. 返回结果集合

1.2.5 根据桥接词生成新文本

1. 首先将用户输入的Str划分为单词列表，{word\_1,word\_2…word\_n}
2. 对于每个word\_i，根据查询桥接词的程序查找word\_i和word\_i+1的桥接词
3. 如果有则在word\_i和word\_i+1之间插入桥接词
4. 否则继续遍历
5. 将结果单词列表拼接成完整的字符串Str并返回

1.2.6 计算两个单词最短路

直接深搜，记录最短路index，在返回时将index转换为对应的单词返回。

选做：对每个目标单词调两个单词最短路，将结果合并为一个列表返回，列表中每个目标单词最短路的顺序按照其index从小到大排序。

1.2.7 随机游走

Graph类只提供设置起点以及单步随机游走的功能。由外层完成控制游走逻辑、结果保存逻辑以及用户自定义终止逻辑

二 基本类设计

2.1 fileIO

作为图的输入输出接口，完成从文本构建图以及将各个功能的输出接口

2.1.1 属性

String FilePath：输入文件的路径

2.1.2 Public方法

Boolean SetFilePath(String Path)：设置输入文件路径。

Boolean CreateGraph(MyGraph G)：从输入文件构建图G。

void OutPutEdges(Map<String, Map<String, Integer>> edges)：输出一系列带权值的边。用于图的展示。

void OutPutPath(ArrayList<String> path)：输出一条路径。用于路径展示。

Boolean OutPutBridgeWords(String word1,String word2,ArrayList<String> bridgeWords)：输出word1到word2的桥接词。

2.2 MyGraph

图类型，完成图的存储以及在图上的计算接口。

2.2.1 属性

ArrayList<ArrayList<Map<Integer, Integer>>> wordGraph：以邻接表的形式存储一张图。在这里，图节点用单词表示。为了表示方便，为每个单词赋予唯一的index，存储时用index表示一个单词。最里层的Map<Integer,Integer>表示一条边，以终点单词的index作为索引，边权值为对应的值。**可以优化的地方：这里最里层的Map只存储一条边，从一个单词出发的所有边用一个List存储。可以用一个Map来存储从一个单词出发的所有边。**

Map<Integer, String> ItoS：存储index到单词的映射关系。

Map<String, Integer> StoI：存储单词到index的映射关系。

String randomWalkNodeName：存储当前随机游走到的节点名称。

2.2.2 Public方法

void AddNode(String name)：为图添加一个节点叫做name。

void AddRelation(String source,String dest)：为图添加一个边，从source节点指向dest节点。

Map<String, Map<String, Integer>> getGraph()：获取图的所有边，带权值，用于fileIO的OutPutEdges方法，输出图结构。

Map<String, ArrayList<String>> getVGraph()：获取图的所有边，不带权值，用于VGraph的CreateGraph方法，创建可视化图。

ArrayList<String> searchBridgeWord(String sour, String dest, AtomicInteger isError)：寻找source和dest之间的桥接词，同时用isError指示是否出错以及出错的类型。

boolean inputCheck(String sour, String dest, AtomicInteger isError)：用于输入检查，通过设置isError来指明错误类型。

ArrayList<String> shortestPath(String source, String destination, AtomicInteger isError)：寻找source和dest之间的最短路，同时用isError指明是否出错以及错误类型。

String SetRandomWalkStartNode()：设置随机游走起点，并返回节点名称。

String RandomWalkOneStep(AtomicInteger isStop)：单步随机游走。返回游走到的节点名称，如果游走终止则设置isStop为1。**存在问题：终止逻辑只检查了无出边情况，没有检查遇到重复边的情况**。

为什么使用AtomicInteger：

在Java中，虽然没有直接支持引用传递的机制，但可以通过创建一个包含整数值的对象来模拟引用传递。一种常见的做法是使用Java中的AtomicInteger类，它提供了一种原子更新整数的方式，并且可以在方法之间传递作为引用传递。

String GetNewText(String originText)：根据用户输入文本返回构建的新文本，功能4的实现。

2.3 VGraph

利用GraphStream可视化的图类，有UI的情况下使用。

2.3.1 属性

Graph graph：graphstream库的图实例*（下面就叫做可视化图）*。

2.3.2 Public方法

void CreatGraph(Map<String, ArrayList<String>> G)：根据MyGraph类的getVGraph方法返回值构建可视化图。

void setNodeFillRed(String nodeName)：将指定节点颜色设置为红色。用于指示桥接词查询和最短路径的word1到word2。

void setNodeFillBlue(String nodeName)：将指定节点颜色设置为蓝色。用于指示查询出来的桥接词以及随机游走到的节点。

void setNodeFillWhite(String nodeName)：将指定节点颜色设置为白色。用于节点颜色复位。

void setEdgeColorGreen(String edgeId)：将指定边设置为绿色。用于指示路径。

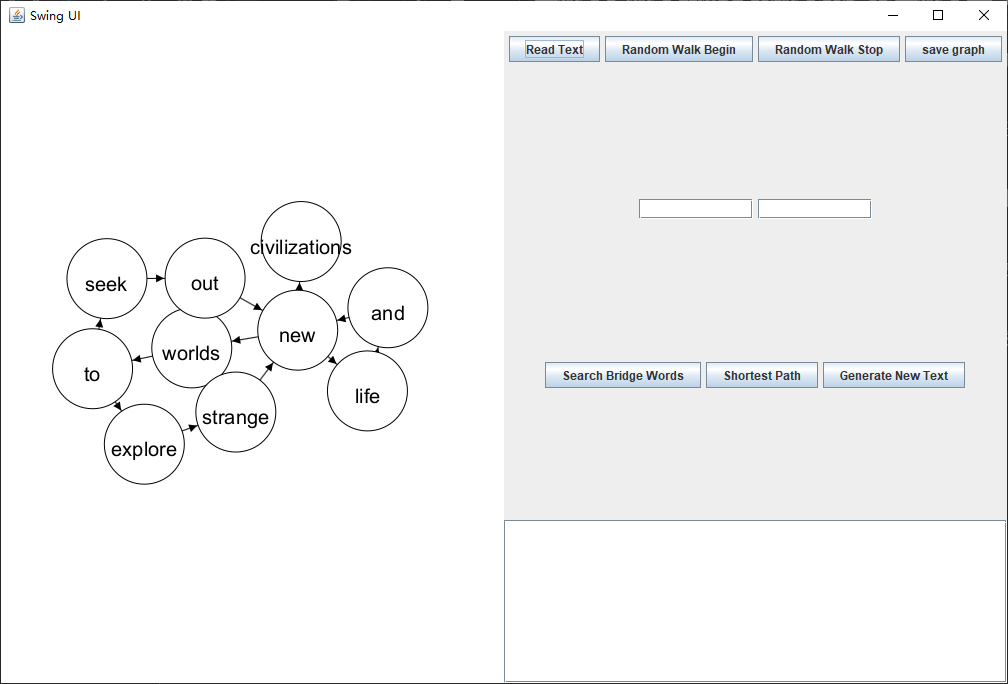
void setAllNodeWhite()：将所有节点设置为白色。用于图节点颜色复位。

void setAllEdgeDefault()：用于图边颜色复位。

void clearGraph()：清楚所有节点和边。用于用户打开另一幅图，实现对象的复用。

2.4 SwingUI

Ui类，实现用户交互逻辑实现以及图的展示。界面如下：



整体分为左右两部分：左侧用于图的展示；右侧放置用户输入以及对应功能的按钮。右侧分为四行：第一行从左到右为ReadText按钮，RandomWalkBegin按钮，RandomWalkStop按钮和saveGrap按钮；第二行为：word1和word2的输入框；第三行从左到右为SearchBrigdeWords按钮，ShortestPath按钮和GenerateNewText按钮。第四行为功能4的输入区。

2.4.1 属性

VGraph graphs：VGraph类的实例，用于可视化控制。

MyGraph G：MyGraph类的实例，用于功能的实现。

fileIO I：fileIO类的实例，用于从文件构建图。

AtomicBoolean isRandomWalkFinished：指示随机游走是否终止，用于随机游走控制逻辑。

AtomicBoolean stopTraversal：随机游走控制变量，用于随机游走控制逻辑。

AtomicInteger isError：错误指示变量，用于word1和word2输入的错误检查。

JButton readText：ReadText按钮，下同。

JButton randomWalkBegin;

JButton randomWalkStop;

JButton searchBridgeWords;

JButton shortestPath;

JButton generateNewText;

JButton saveGraph;

Thread randomWalkThread：随机游走线程。

JTextField inputWord1：word1的输入框。

JTextField inputWord2：word2的输入框。

JTextArea originText：功能4的用户输入部分。

2.4.2 Public方法

void show()：将所有组件实例化，并使用JPanel进行排版最后显示UI界面。

2.4.3 按钮响应逻辑

三 方法实现逻辑

3.1fileIO

3.2 MyGraph

3.2.1 void addNode(String name)

1. 首先判断要添加的节点是否已经在图中
2. 如果不在图中则赋予其index，值为当前图中的节点个数，这样就可以用index直接方位其邻接表
3. 设置index与name的映射关系
4. 在wordGraph中添加其邻接表记录

3.2.2 void addRelation (String source, String destination)

1. 首先获取source节点和dest节点的index分别为sourceIndex和destIndex
2. 分为两种情况，图中已经有这条边和图中无这条边
3. 遍历source节点的邻接表记录，查找是否有这条边的记录，如果有则删除原始记录并添加新的权值加一的记录
4. 如果遍历完了还是没有这条边的记录，则直接添加边记录，权值为1。

3.2.3 Map<String, Map<String, Integer>> getGraph()

1. 首先创建一个Map<String, Map<String, Integer>>变量graph
2. 遍历wordGraph，对于每一个邻接表记录
   1. 获取起点名称
   2. 获取其邻接表记录
   3. 创建一个Map<String, Integer>变量newEdge
   4. 对于邻接表的每个边记录
      1. 获取终点名称以及权值
      2. 添加到newEdge中
   5. 将该起点记录添加到graph中
3. 返回graph

3.2.4 Map<String, ArrayList<String>> getVGraph()

与2.2.3 Map<String, Map<String, Integer>> getGraph()类似，不用记录权值。

3.2.5 ArrayList<String> searchBridgeWord(String sour, String dest, AtomicInteger isError)

1. 首先调用inputCheck方法对输入进行检查
2. 新建一个ArrayList<String>变量bridgeWords存储桥接词
3. 获取sour和dest的index分别为sIndex和dIndex
4. 获取sour节点的邻接表记录
5. 对于邻接表的每个边记录
   1. 获取终点index为mIndex
   2. 找到mIndex的邻接表记录
   3. 对于mIndex邻接表每个边记录
      1. 获取终点index
      2. 判断终点index是否与dIndex相同
      3. 如果相同则将mindex对应的单词加入bridgeWords中
6. 返回bridgeWords

3.2.6 boolean inputCheck(String sour, String dest, AtomicInteger isError)

输入检查。如果只有sour不在图中，则设置isError为1并返回true；如果只有dest不在图中，则设置isError为2并返回true；如果两个都不在图中，则设置isError为3并返回true。否则返回false。

3.2.7 String GetNewText(String originText)

1. 首先将输入分割为单词构成的List为words
2. 如果words长度小于等于1，即只有一个单词或没有单词，直接返回originText
3. 新建一个ArrayList<String>变量newWords
4. 对于words中每一对单词word1和word2
   1. 首先检查是否都在图中
   2. 如果都在图中
      1. 查询桥接词
      2. 如果为空则直接将word1添加到newWords中
      3. 否则随机选择一个单词，跟随在word1后添加到newWords中
   3. 如果有一个不在图中，直接将word1添加到newWords中
5. 将words中最后一个单词添加到newWords中
6. 将newWord中单词以空格分隔组成一个字符串并返回

3.2.8 ArrayList<String> shortestPath(String source, String destination, AtomicInteger isError)

1. 首先调用inputCheck方法对输入进行检查
2. 具体的最短路逻辑
3. 返回最短路

3.2.9 String SetRandomWalkStartNode()

1. 从节点中随机选择一个节点名称nodeName
2. 设置randomWalkNodeName为nodeName
3. 返回nodeName

3.2.10 String RandomWalkOneStep(AtomicInteger isStop)

1. 获取当前节点index为currentNodeIndex
2. 获取当前节点的邻接表
3. 判断是否有出边，如果有则设置isStop为1并返回空串
4. 随机选择一个后继节点nextNode
5. 设置randomWalkNodeName为nextNode
6. 返回nextNode

3.3 Vgraph

3.4 SwingUI