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 GraphIO

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

FilePath(String)：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。

Boolean OutPut NodePairs (ArrayList<ArrayList<String>> nodePairs)：输出一系列边。用于图的展示以及路径输出。

Boolean OutPutBridgeWords(String word1,String word2,ArrayList<String> bridgeWords)：功能3的输出接口。

Boolean OutPutNewText(ArrayList<String> words)：功能4的输出接口，将List的每个单词用空格分割输出。

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

TODO：

2.2.2 Public方法

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

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

ArrayList<ArrayList<String>> GetGraph()：获取图的所有边。

ArrayList<String> SearchBridgeWords(String source,String dest)：寻找source和dest之间的桥接词。

ArrayList<ArrayLsit<String>> SingleShortestPath(String source,String dest)：寻找source和dest之间的最短路。

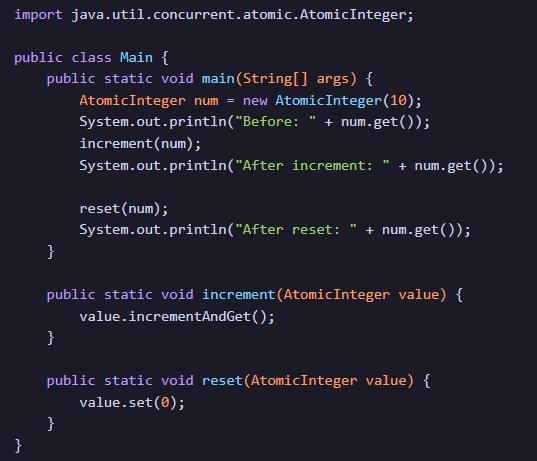
ArrayList<ArrayLsit<String>> MultiShortestPath(String sourcet)：寻找source和其余节点之间的最短路。

Boolean SetWalkStartNode(String startNode)：设置随机游走起点。

String WalkOneStep(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。

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。



2.3 Main

完成与用户的交互以及流程控制功能

TODO

2.4 ShowGraph

利用GraphStream可视化的图类，有UI的情况下使用。

TODO