评估的个人课程1-拼写检查

学习成果

- *能够针对给定问题进行分析并因此选择合适的算法和数据结构的能力*
- *根据一系列标准算法设计和实现中等规模的程序*
- *了解抽象数据类型（ADT）属性和具体的ADT实现之间的区别*
- *意识到需要将多个ADT集成到大量程序中*
- *通过ADT重用来评估效率/担保*
- *在编程环境中发展实用的解决问题的能力*
- *能够针对给定问题进行批判性分析并因此选择合适的算法和数据结构*

1概述

在此作业中，要求您编写一个简单的拼写检查程序。您的程序应命名为 拼写检查 并且它将两个文件名作为命令行参数。第一个名称是字典文件，其中包含正确拼写的单词（例如，

dictionary.txt 文件）。第二个文件包含要进行拼写检查的文本（例如，提供的 text-to-check.txt 文件）。您的程序应首先从字典文件中读取所有单词，然后将它们插入到设置的数据结构中。然后，您的程序应从第二个文件中读取单词，并检查它们是否在集合中。对于集合中确实存在的单词，无需执行任何操作。对于不在集合中的单词，您的程序应通过打印到标准输出来建议可能的正确拼写。您应该对本节中给出的拼写错误的单词进行修饰 3 。在此作业中，您将实现并比较（实验上）基于链接列表和基于哈希表的集合。

指示性时间准则

步骤1：从第3周到 第4周

开发基于链接列表的集合，实现并测试一些拼写错误的单词修饰

步骤2：从第4周到第5周

开发基于哈希表的集合（您可以先实现线性探测冲突处理，然后再更改为双哈希），实现并测试剩余的拼写错误的单词修饰

步骤3：从第5周到第6周

最终使您实现两次哈希冲突处理，运行测试，对基于列表的实现和基于哈希表的实现进行比较，最后确定您的报告

2实施

您要基于一个集合来实现该程序，让我们称之为 *W*. 您将使用 *w ^* 用于将单词存储在第一个命令行参数指定的字典输入文件中。将所有单词存储在 *W*， 您应该打开第二个文件（将要进行拼写检查的文件）以进行阅读和查看

集合中第二个文件中的单词 *W.* 如果有话 *w* 第二个文件不在 *W*，您必须尝试所有可能的修改 *w* 上面建议的部分 3 。注意，不同的修饰可能导致相同的词。set的实现（请参阅下文）确保每个set元素都是唯一的单词。因此，为确保可能的拼写列表中没有重复项，请创建第二组 *S*（对于每个拼错的单词），并插入集合中对拼错的单词的所有修改 *W.* 完成所有修改后，打印出存储在集合中的所有单词 *S.*

3拼错单词的修饰

您应该对拼写错误的单词进行修饰以处理常见的错误：

- *字母替换*：请仔细检查拼写错误的单词中的所有字符，然后尝试用其他任何字符替换该字符。在这种情况下，如果有 k 一个单词中的字符，尝试的修改数量为26 *k*。例如，在拼写错误的单词“ lat”中，用“ c”代替“ l”将产生单词“ cat”，该单词在字典中。

- *信件遗漏*：尝试在拼写错误的单词中省略（依次一个一个）单个字符，然后查看字典中是否包含省略字符的单词。在这种情况下，k
  修改尝试在哪里 k 是单词中的字符数。例如，如果拼写错误的单词是“ catt”，则省略最后一个字符“ t”将在词典中生成单词“ cat”。

- *字母插入*：尝试在拼写错误的单词中插入字母。在这种情况下，如果单词是 k
  字符长，有 26 *（ k + 1）尝试修改，因为有 26 尝试插入的字符和 k + 1个 插入字符的位置（包括单词的开头和结尾）。例如，对于单词“ plce”，在中间插入字母“ a”将产生正确拼写的单词“ place”。

- *字母反转*：尝试每隔2个相邻字符交换一次。一言以蔽之 k，有
  k - 1个 对尝试交换。例如，在拼写错误的单词“ paernt”中，交换字母“ e”和“ r”将产生正确拼写的单词“ parent”。

对于每个拼写错误的单词，请单独打印出拼写错误的单词以及您在词典中找到的所有可能的正确拼写。例如，如果您的词典文件包含单词“ cats like on to play”，而拼写检查文件包含4个单词：“ Catts lik o play”，则输出应为：

猫=>猫
lik =>喜欢
o =>在...上

请注意，可能的正确拼写列表必须仅包含唯一单词。在上面的示例中，对于拼写错误的单词“ catts”，删除第一个“ t”或第二个“ t”会导致相同的单词“ cats”，但该单词仅在输出中出现一次。对于上述单词的修改，Java类 字符串缓冲区 它的内置方法非常有用。一个文件 FileWordRead.java 它会从打开的文件中读取下一个单词。查看文件中的评论 FileWordRead.java 的用法。在此实现中，所有单词都转换为小写，因此单词“ Cat”和“ cat”被视为一个单词。因此所有

使用该程序从文件中读取的单词将为小写单词。方法
建议 由...声明 拼写 接口并在 拼写字母
班级应该从给定的单词和给定的字典中生成建议列表。

4类和接口

你应该叉 F28DA-20-21-CW1 在GitLab Student上进行项目，并定期提交和推动您的工作（记住要添加新文件）。你应该 **不是** 邀请其他任何学生参加您的项目，或与其他学生共享您的代码（请参见第 7 ）。请记住，最终提交是通过 **想象 (** 见章节 8 ）。

拼写检查 （ **课程完成)**

这是包含主程序的类。您必须编写大多数主程序，其中的代码 SpellCheck.java 当前读取命令行参数（文件名）。班级名称必须保持不变， 拼写检查。 在您提供的版本中，您应该使用哈希表集实现。基于链表的实现应仅用于与哈希表实现进行比较（请参见第5节）。 5 ）。

拼写（ **提供的界面)**

拼写建议的界面。

SpellingImpl（ **课程完成)**

此类实现了拼写建议（请参见第 3 ）。该类必须实现 建议 提供的方法 拼写 接口，所有其他方法和任何成员变量都必须是私有的。

公共WordsSet建议（字符串单词，WordsSet dict）

建议对给定单词和给定单词词典进行单词修改。

WordsSet（ **提供的界面)**

您的单词集的界面应实现（基于链接列表的集和基于哈希表的集）。

SpellCheckException（ **提供的课程)**

如果发生意外情况，您的设备应该抛出此异常，请参阅类
HTWordsSet 和 LLWordsSet 对于引发此异常的情况。

LLWordsSet（ **类实施)**

此类基于链接列表实现一组单词。您可以使用Java的内置
链表 类（导入 java.util.LinkedList）。 此类必须实现提供的方法 单词集 界面。您必须实现以下公共方法，并且可能为该类实现的所有其他方法必须是私有的。同样，任何成员变量都必须是私有的。

公共LLWordsSet（ ）

该类的构造函数。

公共无效insWord（String word）抛出SpellCheckException

向集合中添加一个单词。抛出 SpellCheckException 如果单词已经存在，则为例外。

公共无效值rmWord（String word）引发SpellCheckException

从集合中删除一个单词。如果不存在这样的单词，则引发异常。

public boolean wordExists（String word）

如果存在单词，则返回true。

公共诠释getWordsCount（ ）

返回集合中存储的单词数。

公共Iterator <String> getWordsIterator（ ）

返回一个 *迭代器* 集中存储在集合中的所有单词。迭代结束于类的对象 *细绳*。 您可以使用 java.util.Iterator 这给了 *迭代器* 界面（要使用此界面，请导入 java.util.Iterator 在你的开始

LLWordsSet.java 文件）。

监视器（ **提供的界面）**

这是基于哈希表的集合应实现的接口。

散列（ **提供的界面）**

这是基于哈希表的集合应实现的接口。

HTWordsSet（ **类实施）**

此类实现基于哈希表的单词集，并应实现提供的 单词集 和 散列 接口。它还应实施 监视器

接口（需要 HTWordsSetProvidedExp）。 您应该使用带有双重哈希策略的开放式寻址。从大小为7的初始哈希表开始。将其大小增加到下一个质数，至少是当前数组大小的两倍（即 *N）* 当负载因子大于最大允许负载因子时（将最大允许负载因子提供给哈希表的构造函数）。您必须设计哈希函数，以使其产生很少的冲突。

您应该实现以下构造函数。

公共HTWordsSet（ ）

该类的构造函数，将最大负载因子设置为 0.5。

公共HTWordsSet（float maxLF）

该类的构造函数，允许在构造时设置最大负载系数。

您必须实现以下公共方法，并且可能为该类实现的所有其他方法必须是私有的。任何成员变量也必须是私有的。

公共无效insWord（String word）抛出SpellCheckException

向集合中添加一个单词。抛出 SpellCheckException 如果单词已经存在，则为例外。

公共无效值rmWord（String word）引发SpellCheckException

从集合中删除一个单词。如果不存在这样的单词，则引发异常。

public boolean wordExists（String word）

如果存在单词，则返回true。

公共诠释getWordsCount（ ）

返回集合中存储的单词数。

公共Iterator <String> getWordsIterator（ ）

返回一个 *迭代器* 集中存储在集合中的所有单词。迭代结束于类的对象 *细绳*。 您可以使用 java.util.Iterator 这给了 *迭代器* 界面（要使用此界面，请导入 java.util.Iterator 在你的开始

HTWordsSet.java 文件）。

public int GiveHashCode（String s）

此方法（由 散列 接口）用于获取字符串的哈希码。您必须对我们在课堂上讨论过的字符串使用多项式累积哈希码。您不应该使用Java的 hashCode 方法。

公共浮动getMaxLoadFactor（ ）

此方法（由 监视器 接口）返回最大授权负载系数。

公共浮动getLoadFactor（ ）

此方法（由 监视器 接口）返回当前的负载系数。

公共浮动getAverageProbes（ ）

此方法（由 监视器 接口）返回到目前为止您的哈希表执行的平均探测次数。您应该计算哈希表执行的操作总数（查找，插入，删除计数作为一项操作，不计算任何其他操作），以及到目前为止哈希表执行的探测总数。什么时候 getAverageProbes（ ）被称为，它应该

返回 （ float）numberOfProbes / numberOfOperations。 随着您减少
最大允许负载系数，平均探头数应下降。当您运行 HTWordsSetProvidedExp 程序，它将在不同的负载系数下运行您的哈希表，并打印出平均探测数与运行时间的关系。如果看到平均探针数随最大负载因子的增加而增加，则可能是您在正确计算探针/实现哈希表。您可以实现所需的任何其他方法，但是必须将它们声明为私有方法。

HTWordsSetProvidedExp（ **提供的课程）**

此类的主要方法进行了一些实验运行，请参见上文。

HTWordsSetProvidedTest（ **提供测试课程）**

这是一个JUnit 4测试类，我们将使用它来测试您的哈希表实现。实施后，编译并运行它 HTWordsSet 班级。它会在哈希表上运行一些测试，并让您知道哈希表通过/失败了哪些测试。阅读此类的源代码，以了解每个测试的内容，并在测试失败的情况下修正您的实现。要获得作业的满分，您必须通过所有测试。

HTWordsSetTest（ **测试类实施)**

在此测试类中为哈希表实现编写您自己的JUnit 4测试用例。

ModificationsProvidedTest（ **提供测试课程)**

这是一个JUnit 4测试类，它对单词修改建议的实现运行一个测试（请参见参考资料 3 ）。

修改测试（ **测试类工具)**

在这个测试类中，为您的单词修改/建议实现编写您自己的JUnit 4测试用例。

5哈希表与链接列表集的实现

不同大小的字典文件（ d1.txt 、。。。。 ，d6.txt ） 提供。使用这些不同的词典和相同的词典运行程序 text-to-check.txt 文件检查单词的拼写。那是第二个命令行参数保持不变，而第一个参数通过 d1.txt 、。。。。 ，d6.txt。 使用Java方法获取运行时间：

System.currentTimeMillis（ ）。 请注意，此方法将返回当前时间， **不是** 从程序开始的运行时间。因此，要获取程序完成的总时间（以毫秒为单位），您应该在程序的最开始处测量当前时间，然后在最开始处测量当前时间，并将两者相减。由于不同运行之间的变化是字典文件的大小，因此我们应将运行时间与字典文件的大小作图，即字典文件中的单词数。计算每个字典文件中的单词数，并在同一张图表上绘制单词数与基于列表和基于哈希的字典实现的运行时间。

运行时间本质上是插入所有词典单词所花费的时间，因为用于拼写检查的第二个文件只有两个单词要检查。当我们将单词插入集合时，我们还必须检查该单词是否已经在集合中。

对于哈希表，检查和插入预计将花费固定的时间，因此，将所有元素插入集合中应花费线性时间。对于链接列表，插入是固定的时间量，但是检查元素是否已在列表中是线性时间量，因此插入集合中的所有元素应花费二次时间。因此，基于哈希表的实现运行时间图应类似于线性函数，基于链表的实现应类似于二次函数。

6编码风格

您的标记将部分基于您的编码样式。以下是一些建议：

- 应该选择变量和方法名称以反映其在程序中的用途。
- 应该使用注释，缩进和空格来提高可读性。
- 变量声明不应包含在方法（“实例变量”）之外，除非它们包含要在对象之间进行维护的数据。换句话说，仅在方法内部需要的变量应在这些方法内部声明，这些变量在下一次方法调用之前不必记住其值。

•应该声明在方法之外声明的所有变量（“实例变量”）私人的
（不是 受保护的）最大限度地隐藏信息。对变量的任何访问都应使用访问器方法（例如 getVar（）和 setVar（...）对于私有变量
var）。

•打印输出时使用适当的流：正常输出应在标准输出上（使用 System.out）。 错误或警告通知应在标准错误输出上（使用 System.err）。

7关于窃和串通的说明

•课程是一项 **个人** 课程。

•您被允许 **讨论** 与同学一起进行的课程。你可以 **得到帮助** 来自讲师和实验室助手的实验会议。您可以获得帮助， **问问题** 通过GitLab-Student或通过电子邮件或在讲座开始或结束时，或在讲师上课的时间内，向讲师发给讲师。

•课程作业报告必须写在您的 **自己的话** 他们课程中的任何代码都必须是您的 **自己的代码**。 如果课程中的某些文本或代码是从其他来源获取的，则这些来源必须是 **正确引用**。 未引用从其他来源获得的作品或未复制其他学生的文字和/或代码是
窃 如果发现，将报告给学校的纪律委员会。如果发现学生犯有窃罪，则处罚可能包括使课程无效。

•学生必须 **绝不** 将课程报告的硬或软拷贝或代码提供给另一位学生。学生必须始终 **拒绝** 另一位学生要求提供其报告和/或代码副本的任何请求。

•与其他学生共享课程报告和/或代码是 **共谋，** 如果发现，将报告给学校的纪律委员会。如果被判犯有共谋罪，则处罚可能涉及使该课程无效。

- **重复使用可用代码的特别说明：** 如果您要重新使用自己编写的代码，则必须明确指出。在任何时候，您都必须弄清楚哪些不是您自己的部分，实际上是您的贡献。只要您不尝试将其作为自己的工作来传递，重复使用现有代码并对其进行修改是完全正确的，因此您必须清楚地说明它的来源。关于您为什么选择此代码的简短补充说明将带来更多好处，甚至为您的工作增加价值。如果您的代码是由标记您的工作的人在其他地方找到的，而您没有提到这一点，则您可能会发现自己不得不去纪律委员会并面临严重的后果。

8提交

提交愿景 以下内容：

资料来源 从GitLab Student提交源代码的存档。为此，1）确保
您已将所有Java文件添加/提交/推送到GitLab Student，2）下载源代码的存档（请参见下图），在Vision上提交存档。提交内容应包括所有必要的Java文件， **不是** 包括编译文件（ **不** 。 罐， **不**

。班级 文件），并且您不应修改提供的接口或称为的测试文件
假如。 如果您认为需要在界面中添加或修改某些方法签名，请首先与课程讲师联系。

报告 简短报告（不超过4页）。您的报告应：

1.在第一页上注明您的姓名，校园和课程，

2.如果您的程序完全符合规范，或者您的程序有已知的局限性，请解释一下您的设计选择

3.提供并讨论链接列表和哈希表实现之间的图表比较。

作业提交截止日期：第7周，星期二23 rd 2021年2月，下午3:30。
该课程适用大学的课程政策。

- 没有针对课程作业提交的个人扩展名。
- 迟交最多5个工作日，将从授予的商标中扣除30％。
- 迟于5个工作日提交作品将不会获得评分。
- 如果您有延期的缓解条件，请与您的私人辅导员联系，并在线提交缓解情况（MC）表格和支持文档 1个 。

您将被要求参加 **同行测试** 提交后。您将使用您准备好的类和单元测试用例。在同级测试期结束时，系统会要求您提交简短的 **反映性摘要** 在视觉上。在某些情况下， **示范** 您的课程可以组织，这需要课程的讲师批准。

9评分方案

你的 **总体评分** 将计算如下。

- 编码风格 拼写检查，SpellingImpl 和 LLWordsSet 实施20分
- HTWordsSet 执行 20分
- HTWordsSetProvidedTest 经过 20分
- 链接列表与哈希表运行时间的报告和比较表 20分
- 提交测试用例，同伴测试的参与和反思性总结 2 20分

1个 https://www.hw.ac.uk/uk/students/studies/examinations/mitigating-circumstances.htm

2 或在情况需要时演示程序