计算机学院

算法大作业报告

动态规划实现文本查重

姓名：何禹姗

学号：2211421

专业：计算机科学与技术

2024 年 6 月 3 日

实验环境：Visual Studio 2022

一、问题描述

文本查重，也就是重复内容检测或相似度检测问题，是一个在多个领域都非常重要的任务。我利用动态规划算法中的最小编辑距离设计了一个程序，用于实现检验 text 文本与 lib 文本的相似度，从而实现文本查重功能。

二、算法描述

1、动态规划之最小编辑距离

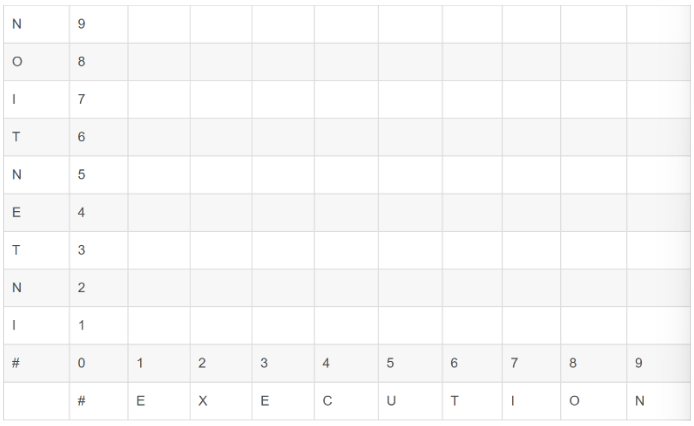
最小编辑距离就是从串 A 转换到串 B 所需的最少编辑操作次数之和。

这里的编辑操作包括：插入 删除 替换

例如：INTENTION 和 EXECUTION 两个单词之间的编辑距离。
INTE-NTION
-EXECUTION
可以看出，从 INTENTION 到 EXECUTION 需要 1 次删除，3 次替换和 1 次插入。
我们把三种操作的代价都记为 1，则其编辑距离为 5。

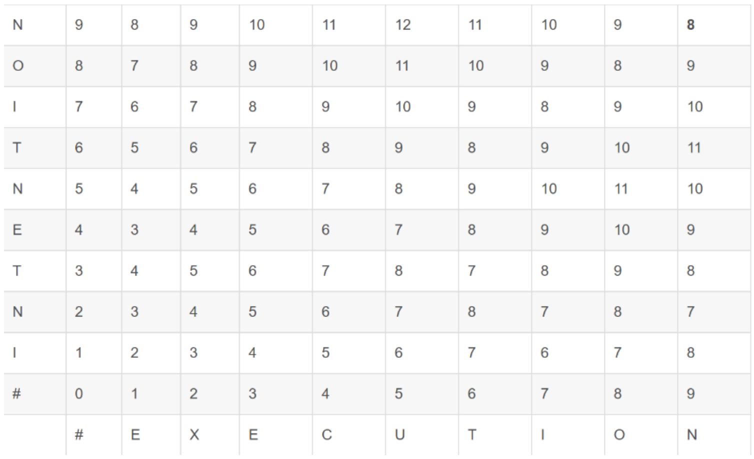
2、算法实现过程

（1）编辑距离表
先构建一张编辑距离表，第二列和倒数第二行，从 0 开始依次加 1，完成编辑距离表的初始化。



N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

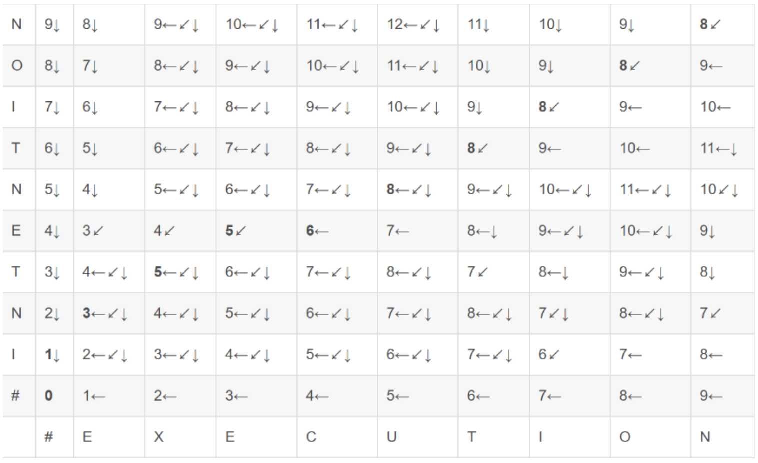
（2）表的填充
第（n,m）格的数值，就是求三个数值中的最小值，如果当前位置对应的两个字符一样，则第三个数值就是(n-1,m-1)的数值。如果当前位置对应的两个字符不一样，则应计算 min(（n-1,m）+1，（n,m-1）+1，(n-1,m-1)+1)，则为最终数值。以此类推，我们可以将整张表计算出来。最终可以算出表中右上角的数值，即为 INTENTION 和 EXEUTION 的最小编辑距离。



N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

（3）带追溯过程的最小编辑距离

从右上角开始，可以画出一条完整的追溯路径。



N	9↓	8↓	9←↙↓	10←↙↓	11←↙↓	12←↙↓	11↓	10↓	9↓	**8**↙
O	8↓	7↓	8←↙↓	9←↙↓	10←↙↓	11←↙↓	10↓	9↓	**8**↙	9←
I	7↓	6↓	7←↙↓	8←↙↓	9←↙↓	10←↙↓	9↓	**8**↙	9←	10←
T	6↓	5↓	6←↙↓	7←↙↓	8←↙↓	9←↙↓	**8**↙	9←	10←	11←↓
N	5↓	4↓	5←↙↓	6←↙↓	7←↙↓	**8**←↙↓	9←↙↓	10←↙↓	11←↙↓	10↙↓
E	4↓	3↙	4↙	**5**↙	**6**←	7←	8←↓	9←↙↓	10←↙↓	9↓
T	3↓	4←↙↓	**5**←↙↓	6←↙↓	7←↙↓	8←↙↓	7↙	8←↓	9←↙↓	8↓
N	2↓	**3**←↙↓	4←↙↓	5←↙↓	6←↙↓	7←↙↓	8←↙↓	7↙↓	8←↙↓	7↙
I	**1**↓	2←↙↓	3←↙↓	4←↙↓	5←↙↓	6←↙↓	7←↙↓	6↙	7←	8←
#	**0**	1←	2←	3←	4←	5←	6←	7←	8←	9←
	#	E	X	E	C	U	T	I	O	N

3、最小编辑距离算法伪代码

```
Algorithm 1 最小编辑距离
procedure MINDISTANCE($word1$, $word2$)
    $n \leftarrow$ size of $word1$
    $m \leftarrow$ size of $word2$
                                        ▷ 初始化一个 (n+1) x (m+1) 的二维数组 dp
    $dp \leftarrow$ a 2D array of size $(n+1) \times (m+1)$, filled with 0s
    for $i$ from 0 to $n$ do
        $dp[i][0] \leftarrow i$
    end for
    for $j$ from 0 to $m$ do
        $dp[0][j] \leftarrow j$
    end for
    for $i$ from 1 to $n$ do
        for $j$ from 1 to $m$ do
            if the $(i-1)$th character of $word1$ equals the $(j-1)$th character of $word2$ then
                $dp[i][j] \leftarrow dp[i-1][j-1]$
            else
                $dp[i][j] \leftarrow 1 + \text{minimum of}(dp[i-1][j-1], dp[i][j-1], dp[i-1][j])$
            end if
        end for
    end for
    return $dp[n][m]$
end procedure
```

三、最小编辑距离算法在文本查重问题中的实现

1、将上述的字符改成字符串（单词）实现，对比每个字符串（单词）是否一致，进行文本查重。所以需要从给定的句子中提取单词，并将它们存储在一个字符串数组中。实现此功能的伪代码如下：

Algorithm 1 Word Split Function within the sentenceTools Namespace

```
procedure WORDSPLIT($sentence$, $wordList$)
    $m \leftarrow 0$
    $Word \leftarrow$ empty string
    $text \leftarrow$ Convert $sentence$ to character array
    for $k = 0$ to length($text$) $-1$ do
        if $text[k]$ is a letter then
            $Word \leftarrow Word \oplus text[k]$
        end if
        if $text[k+1]$ is not a letter or $k$ is the last element of $text$ then
            if $Word$ is not empty then
                $wordList[m] \leftarrow Word$
                $m \leftarrow m + 1$
                $Word \leftarrow$ empty string
            end if
        end if
    end for
    return $m$ as the number of words
end procedure
```

2、我设置了计算文本相似度的规则，对比两个文本计算其最小编辑距离，编辑操作包括插入，删除和替换，三种操作的代价均记为 1，最终计算出的最小编辑距离记为 maxSimilarity。由于我是将 text.txt 中的文本与 lib 中的文本进行比对，所以最终相似度百分比用（(libCount - maxSimilarity) / libCount）计算得出，libCount 为 lib.txt 文本中每句话的单词数量。

则计算相似度的代码实现思路和伪代码如下：

与上述描述的基础的最小编辑距离算法大体一致，先初始化一个 score 矩阵，将 score[i][0]，score[0][i]均设置为 i。开始双重循环，用于填充 score 矩阵的其余部分。

如果当前比较的两个单词相等，那么 score[i][j]被设置为 score[i - 1][j - 1]，意味着不需要额外的操作来匹配这两个单词。

如果单词不匹配，score[i][j]被设置为 1 + min(score[i - 1][j - 1], min(score[i][j - 1], score[i - 1][j]))，这代表了三种操作的最小成本：替换、插入、删除。

检查当前句子对的最小编辑距离是否小于已知的最大相似度。如果是，那么更新最大相似度及相应的索引，并将 insert_space 复制到 max_insert_space 中，用于后续的分析或操作。

最后更新最大相似度为当前句子对的最小编辑距离，记录产生最大相似度的句子对的索引，计算相似度百分比。

Algorithm 1 Dynamic Programming for Sentence Similarity

```
procedure CalculateSimilarity
    Initialize $m$ as the word count of $text$[k]
    Initialize $n$ as the word count of $lib$[x]
    for $i \leftarrow 0$ to $m-1$ do
        $score[i][0] \leftarrow i$
    end for
    for $i \leftarrow 0$ to $n-1$ do
        $score[0][i] \leftarrow i$
    end for
    for $i \leftarrow 1$ to $m$ do
        for $j \leftarrow 1$ to $n$ do
            if $tempWord[i-1]$ equals $libWord[j-1]$ then
                $score[i][j] \leftarrow score[i-1][j-1]$
                $insert\_space[i][j][0] \leftarrow i-1$
                $insert\_space[i][j][1] \leftarrow j-1$
            else
                $score[i][j] \leftarrow 1 + \min(score[i-1][j-1], score[i][j-1], score[i-1][j])$
                if $score[i][j] = score[i-1][j-1] - 1$ then
                    $insert\_space[i][j][0] \leftarrow i-1$
                    $insert\_space[i][j][1] \leftarrow j-1$
                else if $score[i][j] = score[i][j-1] - 1$ then
                    $insert\_space[i][j][0] \leftarrow i-1$
                    $insert\_space[i][j][1] \leftarrow j$
                else if $score[i][j] = score[i-1][j] - 1$ then
                    $insert\_space[i][j][0] \leftarrow i$
                    $insert\_space[i][j][1] \leftarrow j-1$
                end if
            end if
        end for
    end for
    if $maxSimilarity > score[m][n]$ then
        for $i \leftarrow 0$ to $m-1$ do
            for $j \leftarrow 0$ to $n-1$ do
                $max\_insert\_space[i][j][0] \leftarrow insert\_space[i][j][0]$
                $max\_insert\_space[i][j][1] \leftarrow insert\_space[i][j][1]$
            end for
        end for
        $maxSimilarity \leftarrow score[m][n]$
        $simIndex \leftarrow k$
        $libsimIndex \leftarrow x$
    end if
    $percent \leftarrow \left(\frac{libCount - maxSimilarity}{libCount}\right) \times 100$
end procedure
```

3、 在计算相似度的基础上，我增加了文本比对功能，即将相似度高的两句话进行对齐，直观的看出进行了插入，删除，替换的哪种操作。具体代码实现思路和伪代码如下：

这段代码先对先前计算的 max_insert_space 矩阵进行回溯，以找出两个句子之间的最佳匹配路径。然后根据这个路径，构造出两个对齐的字符串 `new_text` 和 `new_lib_text`，其中使用空格进行填充以保持对齐。

创建一个栈 insert_space_stack，用于保存回溯过程中需要访问的索引位置。回溯过程的循环，从 score 矩阵的右下角开始，根据 max_insert_space 矩阵中的信息回溯到左上角，将沿途经过的索引压入栈中。

初始化 new_text 和 new_lib_text 为空字符串，用于构造最终的对齐字符串。分配动态数组 tWord 和 libWord，用于存储 text[simIndex]和 lib[libsimIndex]中的所有单词。text[simIndex] 和 lib[libsimIndex]中的单词分解并存储在 tWord 和 libWord 数组中。

初始化变量 i, j, before_i 和 before_j，用于跟踪回溯过程中的当前位置和前一个位置。当 insert_space_stack 不为空时，从栈顶取出一对索引 i 和 j，然后根据这对索引在 tWord 和 libWord 数组中找到对应的单词，将其添加到 new_text 和 new_lib_text 中。

计算两个单词的长度差 w，并通过添加空格使两个字符串在相同位置的字符对齐。根据 i 和 j 是否与 before_i 和 before_j 相等，决定是否需要添加额外的空格到 new_text 或 new_lib_text 中，以保持对齐。更新 before_i 和 before_j 为当前的 i 和 j，然后从栈中弹出这对索引，继续回溯过程直到栈为空。

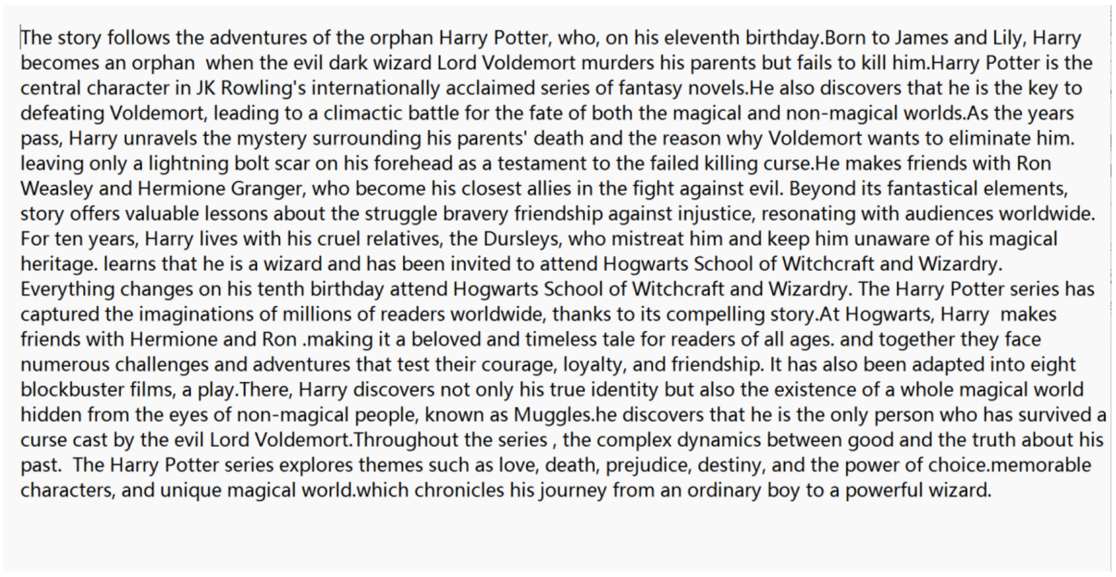
Algorithm 1 Align Text and Library Text

```
$mm \leftarrow$ word count of $text[simIndex]$
$nn \leftarrow$ word count of $lib[libsimIndex]$
Initialize stack $insert\_space\_stack$
$insert\_space\_stack.push([mm-1, nn-1])$
                                                        ▷ 开始回溯
$i \leftarrow mm-1, j \leftarrow nn-1$
while $i > 0 \land j > 0$ do
    $insert\_space\_stack.push(max\_insert\_space[i][j])$
    $temp \leftarrow i$
    $i \leftarrow max\_insert\_space[temp][j][0]$
    $j \leftarrow max\_insert\_space[temp][j][1]$
end while
Initialize strings $new\_text, new\_lib\_text$
Create arrays $tWord, libWord$ for storing words from $text[simIndex], lib[libsimIndex]$
Call $sentenceTools :: wordSplit$ to obtain all words from $text[simIndex], lib[libsimIndex]$ into $tWord, libWord$
Initialize $i, j, before\_i, before\_j$
while $insert\_space\_stack \neq \emptyset$ do
    $(i, j) \leftarrow$ top element of $insert\_space\_stack$
    if $i \neq before\_i \land j \neq before\_j$ then
        $new\_text.append(tWord[i] +$ space$)$
        $new\_lib\_text.append(libWord[j] +$ space$)$
        $w \leftarrow |tWord[i]| - |libWord[j]|$
        while $w > 0$ do
            $new\_lib\_text.append($space$)$
            $w \leftarrow w - 1$
        end while
        while $w < 0$ do
            $new\_text.append($space$)$
            $w \leftarrow w + 1$
        end while
    else if $i = before\_i$ then
        $new\_lib\_text.append(libWord[j] +$ space$)$
        for $l = 0 \to |libWord[j]|$ do
            $new\_text.append($space$)$
        end for
    else if $j = before\_j$ then
        $new\_text.append(tWord[i] +$ space$)$
        for $l = 0 \to |tWord[i]|$ do
            $new\_lib\_text.append($space$)$
        end for
    end if
    $before\_i \leftarrow i, before\_j \leftarrow j$
    $insert\_space\_stack.pop()$
end while
```

四、测试样例与结果

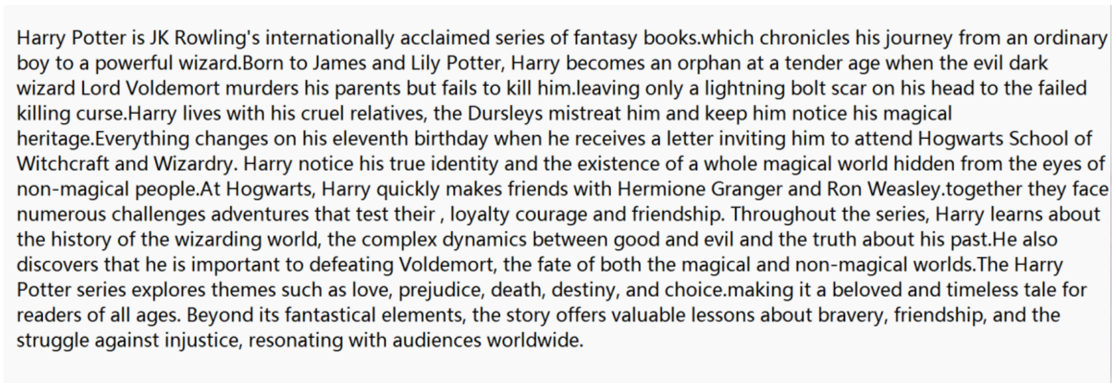
输入：以 txt 文件的格式，分别输入 lib.txt 和 text.txt 文件，文件内分别为一段文本。

Lib.txt



```
The story follows the adventures of the orphan Harry Potter, who, on his eleventh birthday.Born to James and Lily, Harry
becomes an orphan  when the evil dark wizard Lord Voldemort murders his parents but fails to kill him.Harry Potter is the
central character in JK Rowling's internationally acclaimed series of fantasy novels.He also discovers that he is the key to
defeating Voldemort, leading to a climactic battle for the fate of both the magical and non-magical worlds.As the years
pass, Harry unravels the mystery surrounding his parents' death and the reason why Voldemort wants to eliminate him.
leaving only a lightning bolt scar on his forehead as a testament to the failed killing curse.He makes friends with Ron
Weasley and Hermione Granger, who become his closest allies in the fight against evil. Beyond its fantastical elements,
story offers valuable lessons about the struggle bravery friendship against injustice, resonating with audiences worldwide.
For ten years, Harry lives with his cruel relatives, the Dursleys, who mistreat him and keep him unaware of his magical
heritage. learns that he is a wizard and has been invited to attend Hogwarts School of Witchcraft and Wizardry.
Everything changes on his tenth birthday attend Hogwarts School of Witchcraft and Wizardry. The Harry Potter series has
captured the imaginations of millions of readers worldwide, thanks to its compelling story.At Hogwarts, Harry  makes
friends with Hermione and Ron .making it a beloved and timeless tale for readers of all ages. and together they face
numerous challenges and adventures that test their courage, loyalty, and friendship. It has also been adapted into eight
blockbuster films, a play.There, Harry discovers not only his true identity but also the existence of a whole magical world
hidden from the eyes of non-magical people, known as Muggles.he discovers that he is the only person who has survived a
curse cast by the evil Lord Voldemort.Throughout the series , the complex dynamics between good and the truth about his
past.  The Harry Potter series explores themes such as love, death, prejudice, destiny, and the power of choice.memorable
characters, and unique magical world.which chronicles his journey from an ordinary boy to a powerful wizard.
```

Text.txt



```
Harry Potter is JK Rowling's internationally acclaimed series of fantasy books.which chronicles his journey from an ordinary
boy to a powerful wizard.Born to James and Lily Potter, Harry becomes an orphan at a tender age when the evil dark
wizard Lord Voldemort murders his parents but fails to kill him.leaving only a lightning bolt scar on his head to the failed
killing curse.Harry lives with his cruel relatives, the Dursleys mistreat him and keep him notice his magical
heritage.Everything changes on his eleventh birthday when he receives a letter inviting him to attend Hogwarts School of
Witchcraft and Wizardry. Harry notice his true identity and the existence of a whole magical world hidden from the eyes of
non-magical people.At Hogwarts, Harry quickly makes friends with Hermione Granger and Ron Weasley.together they face
numerous challenges adventures that test their , loyalty courage and friendship. Throughout the series, Harry learns about
the history of the wizarding world, the complex dynamics between good and evil and the truth about his past.He also
discovers that he is important to defeating Voldemort, the fate of both the magical and non-magical worlds.The Harry
Potter series explores themes such as love, prejudice, death, destiny, and choice.making it a beloved and timeless tale for
readers of all ages. Beyond its fantastical elements, the story offers valuable lessons about bravery, friendship, and the
struggle against injustice, resonating with audiences worldwide.
```

输出结果如下：首先分别输出 text 和 lib 中的句子个数，然后从 text 的第一句开始输出每一句与 lib 中的哪一句相似度最高，且输出相似度，将两个文本对齐比对，使其更直观的看出进行了增加，删除和替换中的那种编辑方式。

输出格式：

Text 中句子的个数为：____

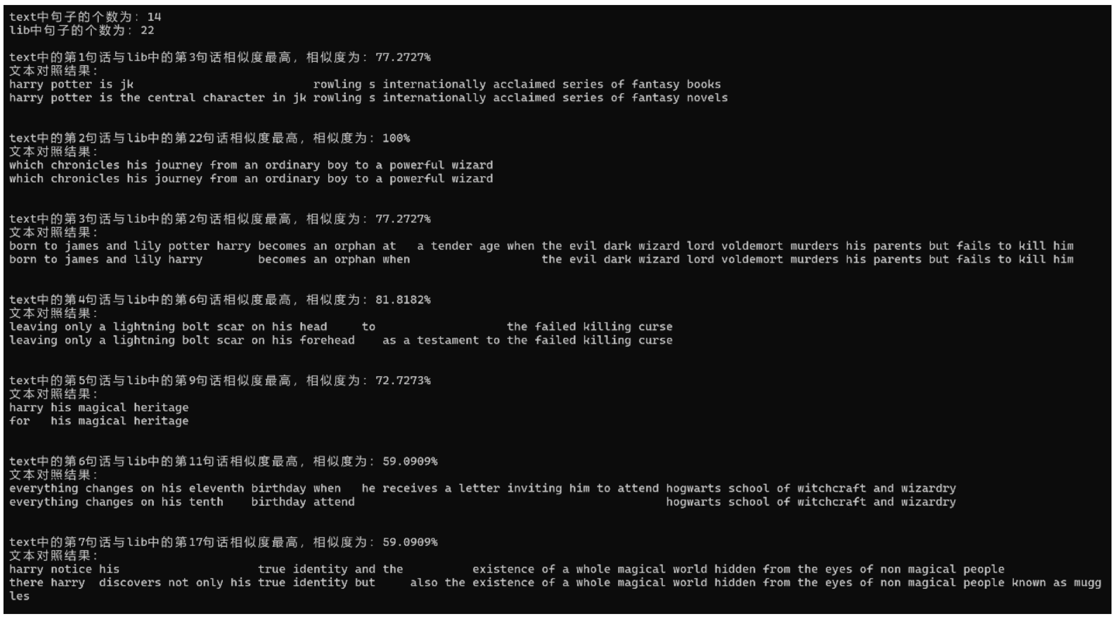
Lib 中句子的个数为：____

Text 中的第 x 句话与 lib 中的第 y 句话相似度最高，相似度为：____

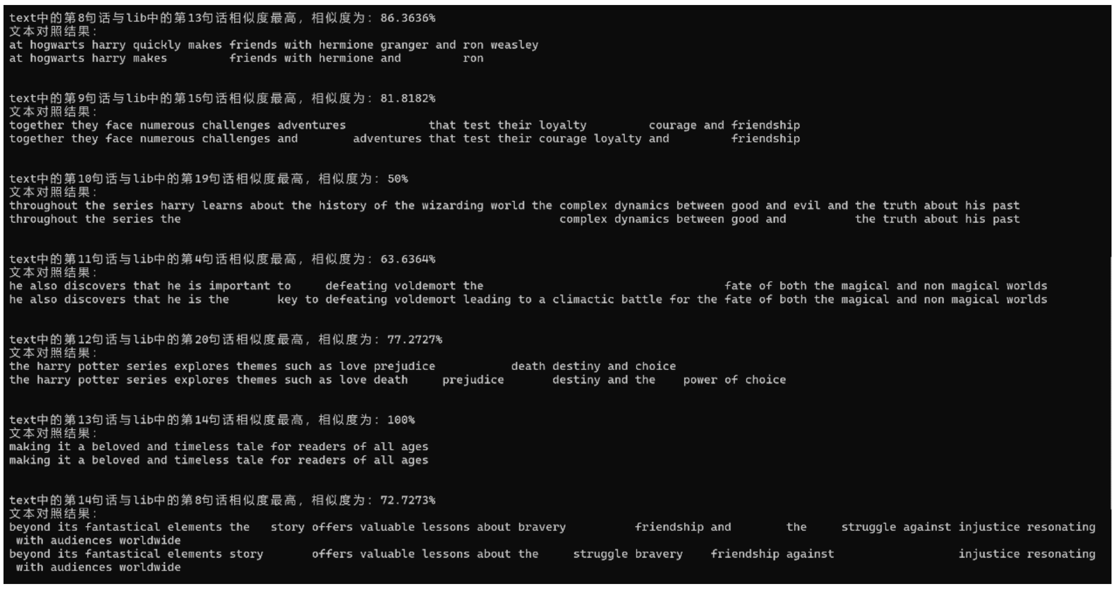
文本对照结果：

Text 文本：____

Lib 文本：____



```
text中句子的个数为: 14
lib中句子的个数为: 22

text中的第1句话与lib中的第3句话相似度最高, 相似度为: 77.2727%
文本对照结果:
harry potter is jk                          rowling s internationally acclaimed series of fantasy books
harry potter is the central character in jk rowling s internationally acclaimed series of fantasy novels


text中的第2句话与lib中的第22句话相似度最高, 相似度为: 100%
文本对照结果:
which chronicles his journey from an ordinary boy to a powerful wizard
which chronicles his journey from an ordinary boy to a powerful wizard


text中的第3句话与lib中的第2句话相似度最高, 相似度为: 77.2727%
文本对照结果:
born to james and lily potter harry becomes an orphan at   a tender age when the evil dark wizard lord voldemort murders his parents but fails to kill him
born to james and lily harry        becomes an orphan when                   the evil dark wizard lord voldemort murders his parents but fails to kill him


text中的第4句话与lib中的第6句话相似度最高, 相似度为: 81.8182%
文本对照结果:
leaving only a lightning bolt scar on his head     to                   the failed killing curse
leaving only a lightning bolt scar on his forehead    as a testament to the failed killing curse


text中的第5句话与lib中的第9句话相似度最高, 相似度为: 72.7273%
文本对照结果:
harry his magical heritage
for   his magical heritage


text中的第6句话与lib中的第11句话相似度最高, 相似度为: 59.0909%
文本对照结果:
everything changes on his eleventh birthday when   he receives a letter inviting him to attend hogwarts school of witchcraft and wizardry
everything changes on his tenth    birthday attend                                             hogwarts school of witchcraft and wizardry


text中的第7句话与lib中的第17句话相似度最高, 相似度为: 59.0909%
文本对照结果:
harry notice his                    true identity and the          existence of a whole magical world hidden from the eyes of non magical people
there harry  discovers not only his true identity but     also the existence of a whole magical world hidden from the eyes of non magical people known as mugg
les
```



```
text中的第8句话与lib中的第13句话相似度最高, 相似度为: 86.3636%
文本对照结果:
at hogwarts harry quickly makes friends with hermione granger and ron weasley
at hogwarts harry makes         friends with hermione and         ron


text中的第9句话与lib中的第15句话相似度最高, 相似度为: 81.8182%
文本对照结果:
together they face numerous challenges adventures            that test their loyalty         courage and friendship
together they face numerous challenges and        adventures that test their courage loyalty and         friendship


text中的第10句话与lib中的第19句话相似度最高, 相似度为: 50%
文本对照结果:
throughout the series harry learns about the history of the wizarding world the complex dynamics between good and evil and the truth about his past
throughout the series the                                                       complex dynamics between good and          the truth about his past


text中的第11句话与lib中的第4句话相似度最高, 相似度为: 63.6364%
文本对照结果:
he also discovers that he is important to     defeating voldemort the                                   fate of both the magical and non magical worlds
he also discovers that he is the       key to defeating voldemort leading to a climactic battle for the fate of both the magical and non magical worlds


text中的第12句话与lib中的第20句话相似度最高, 相似度为: 77.2727%
文本对照结果:
the harry potter series explores themes such as love prejudice           death destiny and choice
the harry potter series explores themes such as love death     prejudice       destiny and the    power of choice


text中的第13句话与lib中的第14句话相似度最高, 相似度为: 100%
文本对照结果:
making it a beloved and timeless tale for readers of all ages
making it a beloved and timeless tale for readers of all ages


text中的第14句话与lib中的第8句话相似度最高, 相似度为: 72.7273%
文本对照结果:
beyond its fantastical elements the   story offers valuable lessons about bravery          friendship and        the     struggle against injustice resonating
 with audiences worldwide
beyond its fantastical elements story       offers valuable lessons about the     struggle bravery    friendship against                  injustice resonating
 with audiences worldwide
```

经验证结果正确。

五、算法分析与总结

1、关于上述涉及到的代码的时间复杂度和空间复杂度分析

最小编辑距离算法:

我们需要构建一个 m+1 行、n+1 列的动态规划矩阵，矩阵中的每个单元格都需要被计算一次。具体而言，算法会遍历两个字符串的所有字符组合，对于每个字符组合，计算最少的编辑操作数，m 和 n 分别是两个字符串的长度，时间复杂度是 O(mn)。

最小编辑距离算法的空间复杂度同样是 O(m*n)，因为需要存储 m+1 行、n+1 列的动态规划矩阵。每个矩阵单元格都存储了到达该位置所需的最小编辑操作数。然而，在一些优化的实现中，可以通过只保留当前行和前一行的信息来减少空间需求，这样空间复杂度可以降低到 O(min(m,n))。这是因为计算任何单元格的值只需要前一行和前一列的值，而不必保留整个矩阵的历史状态。

最小编辑距离回溯算法：
回溯的过程是从动态规划矩阵的右下角开始，沿着最优路径回溯到左上角，每一步都依据预存的决策方向进行。在标准的实现中，每个单元格都保存了到达该单元格的前一个单元格的信息，这样可以确保回溯时每次都能确定下一个要访问的单元格。由于动态规划矩阵的尺寸是 m×n，其中 m 和 n 分别是两个字符串的长度，回溯的过程最多会访问 m+n-1 个单元格（从右下角到左上角的对角线路径），因此回溯部分的时间复杂度是 O(m+n)。

在动态规划阶段已经保存了每个单元格的前驱信息，则回溯阶段的空间复杂度主要取决于保存路径的栈，最坏情况下需要存储 m+n-1 个坐标点，因此空间复杂度为 O(m+n)。

2、 用最小编辑距离算法解决文本查重问题总结

优点：最小编辑距离能够精细地测量两段文本之间的相似度，还能处理文本中的拼写错误、同义词替换、句子结构调整等，提高了查重系统的鲁棒性，不仅能给出两个文本的相似度分数，还可以通过回溯过程找出具体的编辑操作，帮助理解文本间的差异所在。

缺点：最小编辑距离的计算涉及动态规划，对于长文本而言，时间和空间复杂度较高，分别为 O(mn)和 O(mn)，其中 m 和 n 是两段文本的长度。这在处理大规模文本数据集时可能会成为瓶颈。在实际应用中，可能需要对不同类型的编辑操作赋予不同的权重（比如替换操作的成本可能高于插入或删除）。这种参数的调整较为复杂，且缺乏统一的标准，可能影响查重结果的准确性。