41. Bundeswettbewerb Informatik

Dokumentation zur Aufgabe 1: „Störung“

Teamname: SRZinfo4/1
Team-Id: 00495
Bearbeiter: Jakob Paridon

Dresden, den 21. November 2022

Schülerrechenzentrum der TU-Dresden

Inhaltsverzeichnis

1 Lösungsidee **2**

2 Umsetzung **2**

3 Beispiele **2**
3.1 stoerung0.txt . . . 2
3.2 stoerung1.txt . . . 2
3.3 stoerung2.txt . . . 2
3.4 stoerung3.txt . . . 2
3.5 stoerung4.txt . . . 2
3.6 stoerung5.txt . . . 2

4 Quellcode **3**
4.1 Speichern aller Stellen mit gegebenen Wörtern . . . 3
4.2 Untersuchen, ob gegebene Wörter im Text an der richtigen Stelle stünden . . . 3
4.3 Auslesen des Satzes und Speichern der Lösung . . . 3
4.4 Ermitteln der gesuchten Zeile . . . 3

1 Lösungsidee

Um für den Lückentext passende Stellen zu finden, können die gegebenen Wörter und der Abstand zwischen ihnen ermittelt werden, um im Anschluss den gesamten Text zu durchlaufen und zu überprüfen, ob diese Wörter – mit den entsprechenden Abständen – vorkommen. Ein Satz, der der gesuchten Konfiguration entspricht, wird gespeichert. Mithilfe des vollständigen Satzes kann die Stelle im Buch recht einfach ermittelt werden. Nach Ablauf des Programms können alle Lösungen gesammelt ausgegeben werden.

2 Umsetzung

Das Programm wird mit `python aufgabe1.py [Name der Datei mit Lückensatz]` bzw. `aufgabe1.exe [Name der Datei mit Lückensatz]` gestartet.
Nach dem Einlesen des gesuchten Textes und des Buches – hieraus werden alle Satzzeichen entfernt, um eine korrekte Erkennung der Wörter zu ermöglichen (sonst würde z. B. die Bestimmung von “»Fressen” als “Fressen” nicht erfolgen) – werden alle Wörter (inklusive der Lücken) in Listen übertragen.
Nun werden für den Lückensatz die Stellen, an denen Wörter gegeben sind, gespeichert (siehe 4.1). Im Anschluss wird der gesamte Buchtext durchlaufen, wobei überprüft wird, ob an der untersuchten Stelle im Text alle Wörter an passenden Positionen stehen. Hierzu wird der Index des momentan untersuchten Wortes jeweils mit der Zahl der Stelle addiert, an welcher ein Wort im zu suchenden Text gegeben ist (siehe 4.2). Nur, wenn alle Wörter an der passenden Stelle stehen (siehe 4.2), werden alle gesuchten Wörter in einen Satz zusammengefasst und im Anschluss gespeichert (siehe 4.3).
Um die Zeile im Buch zu ermitteln, wird der gesamte Buchtext erneut durchlaufen, wobei für jede Zeile geprüft wird, ob sie den entsprechenden Satz enthält (siehe 4.4).

3 Beispiele

Die Testung erfolgte unter Artix Linux mit Python 3.10.7.

3.1 stoerung0.txt

```
Lösung: ,,Das kommt mir gar nicht richtig vor‘‘ in Zeile 440
```

3.2 stoerung1.txt

```
Lösung: ,,Ich muß in Clara verwandelt‘‘ in Zeile 425
Lösung: ,,Ich muß doch Clara sein‘‘ in Zeile 441
```

3.3 stoerung2.txt

```
Lösung: ,,Fressen Katzen gern Spatzen?‘‘ in Zeile 214
Lösung: ,,Fressen Katzen gern Spatzen?‘‘ in Zeile 214
Lösung: ,,Fressen Spatzen gern Katzen?‘‘ in Zeile 214
```

3.4 stoerung3.txt

```
Lösung: ,,das Spiel fing an‘‘ in Zeile 2319
```

3.5 stoerung4.txt

```
Lösung: ,,ein sehr schöner Tag‘‘ in Zeile 2293
```

3.6 stoerung5.txt

```
Lösung: ,,Wollen Sie so gut sein‘‘ in Zeile 2185
```

4 Quellcode

4.1 Speichern aller Stellen mit gegebenen Wörtern

```
1
for i in range(len(searchWords)):
3   if searchWords[i] != "_":
      searchIndices.append(i)
```

4.2 Untersuchen, ob gegebene Wörter im Text an der richtigen Stelle stünden

```
for origIndex in range(len(originWords)):
2   fit = True
    for searchIndex in searchIndices:
4     if originWords[origIndex+searchIndex].upper() != searchWords[searchIndex].upper():
          fit = False
6       break
```

4.3 Auslesen des Satzes und Speichern der Lösung

```
if fit == True:
2   solution = ""
    for j in range(len(searchWords)):
4     solution += originWords[origIndex+j] + "␣"
      solutions.append(solution[:-1])
```

4.4 Ermitteln der gesuchten Zeile

```
1 def getLine(phrase):
      lines = open("Alice_im_Wunderland.txt", encoding="utf-8").readlines()
3     for lineIndex in range(len(lines)):
          if phrase in lines[lineIndex]:
5             return lineIndex
```