# Junioraufgabe 1 - Parallelen

19.11.2019

## Ansatz

Laut der Aussage von Martin soll in dem Gedicht *‚Die zwei Parallelen‘* von *Christian Morgenstern* eine mathematische Struktur versteckt sein.
Um diese herauszufinden, wählt man irgendein Wort aus der ersten Hälfte des Gedichts und springt die Anzahl *n*, die der Länge des gewählten Wortes entspricht, an Wörtern weiter. Dies wiederholt man solange bis man am Ende des Textes angelangt, wobei man für die neue Anzahl *n* die Buchstaben des neuen Wortes zählt. Und so endet man zu Schluss immer am selben Wort.
Zumindest würde es immer dasselbe Wort ergeben, sofern Martins Aussage stimmt.
Nach eigenem Überprüfen per Hand mit mehreren verschiedenen Anfangswörtern existiert diese Struktur tatsächlich. Hierbei endet man immer bei dem Wort *‚verschlang‘*.

```
Es(2) -> zwei(4) -> hinaus(6) -> solidem(7) ->
bis(3) -> seliges(7) -> beiden(6) -> als(3) ->
Lichtjahre(10) -> nicht(5) -> Warn(4) -> Sie(3) ->
nicht(5) -> wie(3) -> zusammen(8) -> sie(3) ->
sie(3) -> ihm(3) -> verschlang(10)
[…]
Das Endwort ist jedes Mal 'verschlang'. Martin hat
recht.
```

**Ausgabe der ersten und letzten Zeile (Beginn & Ende fett gekennzeichnet)**

## Einlesen der Datei

Zu aller Erst muss das Gedicht von dem Programm eingelesen werden, um damit zu arbeiten. Danach war es nötig sowohl die Wörter einzeln aufzurufen können und nicht immer aus dem ganzen Text heraussuchen zu müssen als auch Satzzeichen, die ja schließlich nicht als Buchstaben zählen, zu entfernen. Hierbei musste auch beachtet werden, dass Zeilenumbrüche ebenfalls aussortiert werden müssen, um eine falsche Wortlänge zu verhindern, sowie das Problem gelöst werden, dass Umlaute nicht richtig angezeigt werden beziehungsweise falsche gezählt werden würden.

```
Gedichtstext
[…] Parallelen?
Sie wußtens selber nicht, –
sie flossen […]

1. Eingelesener Gedichtstext
```

```
[…] Parallelen?
Sie wuÃŸtens selber nicht, â€“
sie flossen […]
```

2. Aufteilung in einzelne Wörter *(zugleich mit Schritt 3)*

```
[…] 'Parallelen?\nSie', 'wuÃŸtens', 'selber',
'nicht,', 'â€“\nsie', 'flossen' […]
```

3. Entfernen von Zeilenumbrüchen *(zugleich mit Schritt 2)*

```
[…] 'Parallelen?', 'Sie', 'wuÃŸtens', 'selber',
'nicht,', 'â€“', 'sie', 'flossen' […]
```

4. Ausschneiden von Satzzeichen

```
[…] 'Parallelen', 'Sie', 'wuÃŸtens', 'selber',
'nicht', '', 'sie', 'flossen' […]
```

**Verarbeitung des eingelesenen Textes**

*1.Aufteilung in einzelne Wörter mit Entfernen der Zeilenumbrüche*

Um das Problem der Aufteilung des Textes in die einzelnen Wörter zu lösen wäre es möglich ganz einfach die Funktion *split()* zu benutzen und nach allen Leerstellen, entspricht allen *" "*, zu trennen. Dies ist im obigen Beispiel bei der Nummer 2 der Fall. Allerdings um zugleich das Entfernen der Zeilenumbruch-Zeichen *\n* ebenfalls durchführen zu können, wie im Schritt 3, ist es deutlich praktikabler statt *split(" ")* nur *split()* zu verwenden. So werden sowohl die Zeilenumbrüche rausgestrichen als auch Wörter nach Leerzeichen oder eben auch nach Zeilenumbrüchen getrennt.

*2.Ausschneiden der Satzzeichen*

Zur Verhinderung, dass Satzzeichen in dem Gedicht als normale Buchstaben mitgezählt werden würden, müssen diese vor dem Lösen von der Anfangsfrage entfernt werden, entsprechend der Nummer 4 im obigen Beispiel zur **Verarbeitung des Textes**. Da dies keine negativen Folgen hat, erfolgt das Ausschneiden einfach durch das Durchgehen jedes Wortes und Überprüfen, ob eines der Zeichen, die alle in der **Tabelle 1** aufgelistet sind, vorhanden ist. Sollte dies der Fall sein, wird dieses mithilfe von *strip()* rausgestrichen.

| **Mögliche Satzzeichen:** | , | . | : | ? | – | ; |
|---|---|---|---|---|---|---|

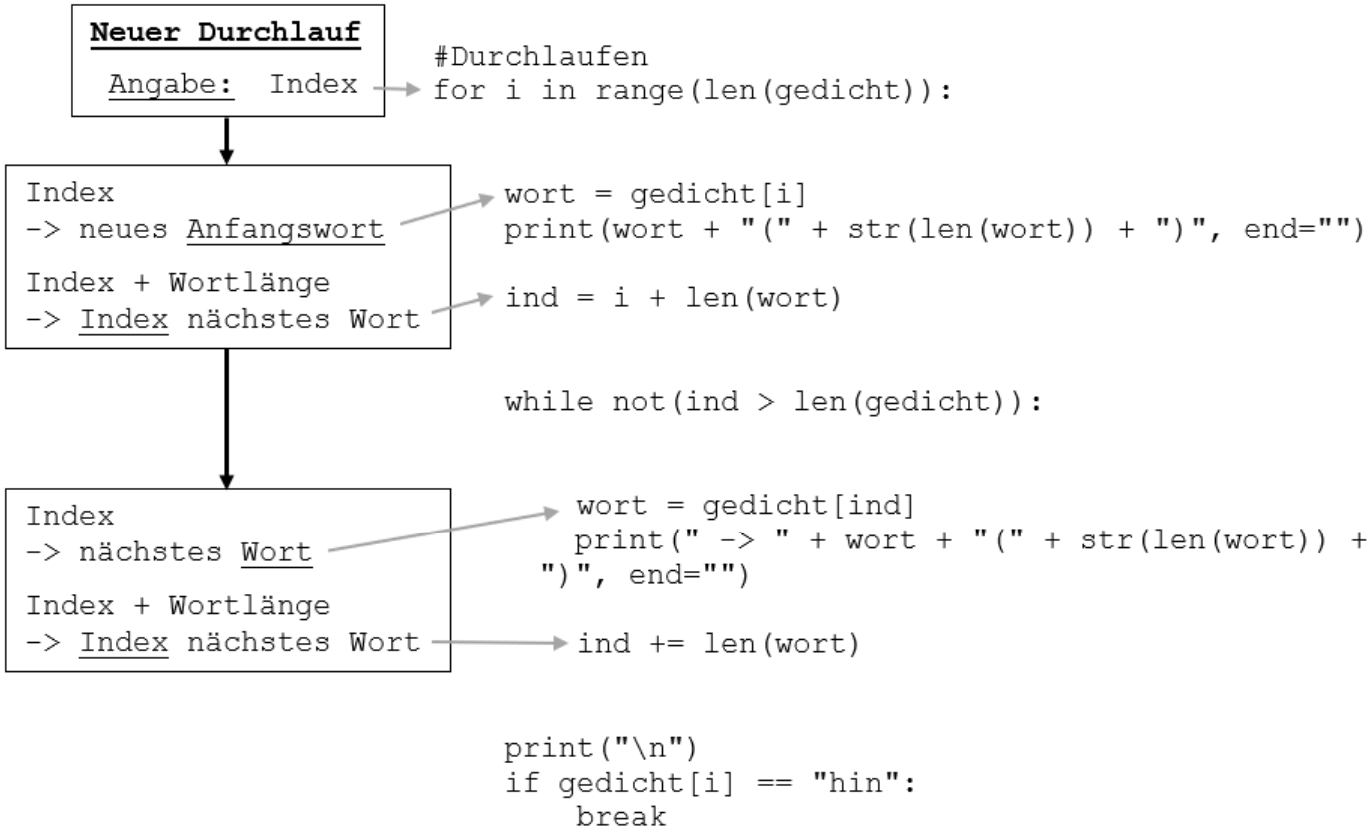
**Tabelle 1**

*3.Problemlösung von Umlauten*

Ein eindeutig schwerer lösbares Problem als das Vorherige mit dem Aufteilen des Textes in die einzelnen Wörter, der Entfernung von Zeilenumbrüchen und von Satzzeichen sind Umlaute und einzelnstehende Satzzeichen. Hierbei gibt es im gesamten Gedicht zwar nur zwei Fälle,

aber sie kommen vor. Einmal ein *‚ß‘*, welches in *‚ÃŸ‘* umgewandelt wird beim Einlesen und zudem noch ein alleinstehendes, also nicht direkt mit einem Wort verbundenes *‚-‘*, das zu *‚â€“‘* verändert wird. Dadurch entstehen zwei Probleme, jeweils eines bezogen auf jeden Vorfall.
Zuerst einmal wird, wenn das Wort *‚wußtens‘*, in welchen nun einmal der Umlaut vorkommt, erreicht werden sollte, anstatt von einer Wortlänge von sieben Buchstaben, die Wortlänge acht ausgegeben und das Programm würde falsch weiterarbeiten. Aus diesem Grund muss das *‚ß‘* wieder zurückformatiert werden.
Das andere Problem mit dem Bindestrich würde bei dessen Aufrufen dazu führen, dass dieser als ein eigenständiges Wort mit zwei Buchstaben erkannt wird, was er jedoch nicht ist. Deshalb muss dieses Zeichen aus den zum Durchlauf zur Verfügung stehenden Wörtern gelöscht werden.
Eine Lösung wäre Encoding des gesamten Programms, allerdings funktionierten weder *'UTF-8'* noch *'iso-8859-1'*. Um dennoch diesen beiden Problemen auszuweichen, wird der Bindestrich zur selben Zeit wie die Satzzeichen entfernt. Für das *‚ß‘* dagegen muss nichts geändert werden, da das Wort mit diesem Buchstaben nie durch einen Sprung erreicht wird.

**Durchlaufen aller möglichen Anfangswörter**
Das Programm muss jedes einzelne Wort des Gedichts in der ersten Hälfte einmal als Anfangswort wählen und bei dem Wort *‚hin‘* enden, das am Ende der letzten Zeile der ersten Gedichtshälfte steht. Dafür soll das Programm so oft durchlaufen wie das Wörter hat, hierfür wird bei null begonnen und aufwärts gezählt, bis, wie schon zuvor gesagt, bei der Hälfte abgebrochen wird. In **folgender Grafik** wird dies bei der *ersten Erklärung* für einen *neuen Durchlauf* übersichtlich aufgezeigt.
Nach jedem Hochzählen wird ein neues Wort, das Nächste, anhand seines Index ausgewählt. Außerdem wird die Position des Wortes, das nach dem ersten Sprung vom aus Anfangswort folgt, ermittelt, indem zu der Position des ausgewählten Wortes dessen Länge hinzuaddiert wird. Dies geschieht bei der *zweiten Erklärung*.
Für jeden Sprung muss zuerst geprüft werden, ob das nächste erreichte Wort überhaupt noch existiert und nicht schon außerhalb des Gedichts liegt. Erst wenn, dies geschehen sollte, wird beim nächsten Anfangswort von vorne begonnen. Funktioniert allerdings der nächste Sprung noch, wird zuerst das nächste Wort mithilfe seines Index gewählt und für das Nächste wird die Länge des Aktuellen zu dem Index des neuen Worts hinzugezählt, ähnlich dem Vorgehen beim Anfangswort. Dies ist mit der *dritten Erklärung* in der **folgenden Grafik** realisiert worden.




**Quelltext für das Durchlaufen jedes Anfangswort mit Erklärungen**

Um für jedes einzelne Anfangswort nachvollziehen zu können, wie man zu dessen Endwort gelangt, wird jedes einzelne Wort dazwischen, das durch einen Sprung erreicht wurde, und zudem auch die jeweilige Länge mit ausgegeben. Hierfür wird am Anfang das erste Wort und dessen Länge ausgegeben, bei allen anderen Worten, die mit einem Sprung verbunden erreicht werden, wird zur Übersichtlichkeit noch davor ein kleiner Pfeil ausgegeben. Um zu erreichen, dass jeweils alle von einem Anfangswort ausgehenden Ausgaben für Sprünge übersichtlich und zusammen ausgegeben werden, wird am Schluss jeder print-Anweisung ein *end=““* angehängt, damit jede folgende print-Anweisung in derselben Zeile ohne irgendetwas dazwischen das nächste Wort mitsamt Länge ausgibt. Damit aber zwischen den einzelnen Durchläufen eine Abtrennung besteht, folgt vor dem nächsten Hochzählen jedes Mal noch ein Zeilenumbruch. Diese Formatierungen zur besseren Übersichtlichkeit der Ausgabe sind in der **folgenden tatsächlichen Programmausgabe** dargestellt.

**Es(2)** -> zwei(4) -> hinaus(6) -> solidem(7) -> bis(3) -> seliges(7) -> beiden(6) -> als(3) -> Lichtjahre(10) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

**gingen(6)** -> zwei(4) -> aus(3) -> Sie(3) -> nicht(5) -> seliges(7) -> beiden(6) -> als(3) -> Lichtjahre(10) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

**zwei(4)** -> hinaus(6) -> solidem(7) -> bis(3) -> seliges(7) -> beiden(6) -> als(3) -> Lichtjahre(10) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

[…]

**gewandert(9)** -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

**neben(5)** -> dem(3) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

**sich(4)** -> dem(3) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

**hin(3)** -> dem(3) -> nicht(5) -> Warn(4) -> Sie(3) -> nicht(5) -> wie(3) -> zusammen(8) -> sie(3) -> sie(3) -> ihm(3) -> **verschlang(10)**

Das Endwort ist jedes Mal 'verschlang'. Martin hat recht.

**Gesamtausgabe des Programms (Beginn & Ende jedes Durchlaufs fett gekennzeichnet)**