Junioraufgabe 1: Parallelen

Team-ID: 00062

Team-Name: CubeFlo

Bearbeiter dieser Aufgabe:
Florian Rädiker

29. September 2019

Inhalt

Lösungsidee

Der Text wird zunächst so zerlegt, dass nur Wörter ohne Satzzeichen und Leerräume in einer Liste übrigbleiben. Dafür wird der Reguläre Ausdruck `\w+` benutzt, bei dem `\w` für alle alphanumerischen Zeichen und den Unterstrich steht und `+` bewirkt, dass `\w` mindestens einmal vorkommt. Dann wird bei jedem Wort der ersten Hälfte (bis Index 42) einmal begonnen und rekursiv solange ein neues Wort beim Index alter Index + Wortlänge besucht, bis es nicht mehr geht. Außerdem wird gespeichert, welche Indizes bereits besucht wurden, um zu verhindern, dass ein Wort mehrfach geprüft wird. Dies ist nämlich nicht nötig, der Weg von einem Wort aus ist immer derselbe.

Darstellung eines Wortes

In der Ausgabe des Programms werden Wörter als `i.Wort(x)` dargestellt, wobei `i` für den Index des Wortes steht und `x` für die Länge.

Umsetzung

Das Programm wurde in Python 3 geschrieben und mit Python 3.7 getestet. Aufgrund der Verwendung von f-Strings ist das Programm nicht mit Python unter Version 3.6 kompatibel.

Die Klasse `Parallelen` stellt alle benötigten Funktionen bereit. Die statische Methode `from_file` liest eine Datei ein, indem mit `re.findall` alle Wörter in der Datei gesucht werden. Außerdem muss dieser Funktion die Wortanzahl der ersten Hälfte (hier 43) übergeben werden. Die Methode check() kann dann aufgerufen werden. Diese ruft für jeden Index der ersten Hälfte (hier also `range(43)`) die Methode `_walk_from_word` auf, welche den Index des Wortes übergeben bekommt. Die Funktion reagiert mit einem Fehler, wenn sich der Index in der Menge der bereits besuchten Indizes befindet (`Parallelen.visited_indices`). Ansonsten wird aus der Liste der Wörter `Parallelen.words` das Wort mit dem Index herausgesucht und dessen Länge zum Index addiert, dann wird das Ergebnis der Funktion `_walk_from_word` mit dem neuen Index zurückgegeben. Sollte der neue Index zu lang sein, wird der alte Index zurückgegeben: Das letzte Wort der Reihe ist gefunden. Alle gefundenen End-Indizes werden in der Menge `Parallelen.end_indices` gespeichert.

Beispiele

Die Menge mit den End-Indizes enthält tatsächlich nur ein Element: 85, was dem Wort „verschlang“ entspricht. Die Ausgabe des Programms sieht folgendermaßen aus:

```
Starte bei Wort '0.Es(2)'
-> 2.zwei(4) -> 6.hinaus(6) ->
12.solidem(7) -> 19.bis(3) -> 22.seliges(7)
-> 29.beiden(6) -> 35.als(3) ->
38.Lichtjahre(10) -> 48.nicht(5) ->
53.Warn(4) -> 57.Sie(3) -> 60.nicht(5) ->
65.wie(3) -> 68.zusammen(8) -> 76.sie(3) ->
79.sie(3) -> 82.ihm(3) -> 85.verschlang(10)

Starte bei Wort '1.gingen(6)'
-> 7.zwei(4) -> 11.aus(3) -> 14.Sie(3) ->
17.nicht(5) -> 22.seliges(7) Wort wurde
bereits besucht

Starte bei Wort '2.zwei(4)'
Wort wurde bereits besucht

Starte bei Wort '3.Parallelen(10)'
-> 13.Haus(4) -> 17.nicht(5) Wort wurde
bereits besucht

Starte bei Wort '4.ins(3)'
-> 7.zwei(4) Wort wurde bereits besucht

Starte bei Wort '5.Endlose(7)'
-> 12.solidem(7) Wort wurde bereits besucht

Starte bei Wort '6.hinaus(6)'
Wort wurde bereits besucht

Starte bei Wort '7.zwei(4)'
Wort wurde bereits besucht
```

```
Starte bei Wort '8.kerzengerade(12)'
-> 20.an(2) -> 22.seliges(7) Wort wurde
bereits besucht

Starte bei Wort '9.Seelen(6)'
-> 15.wollten(7) -> 22.seliges(7) Wort
wurde bereits besucht

Starte bei Wort '10.und(3)'
-> 13.Haus(4) Wort wurde bereits besucht

Starte bei Wort '11.aus(3)'
Wort wurde bereits besucht

Starte bei Wort '12.solidem(7)'
Wort wurde bereits besucht

Starte bei Wort '13.Haus(4)'
Wort wurde bereits besucht

Starte bei Wort '14.Sie(3)'
Wort wurde bereits besucht

Starte bei Wort '15.wollten(7)'
Wort wurde bereits besucht

Starte bei Wort '16.sich(4)'
-> 20.an(2) Wort wurde bereits besucht

Starte bei Wort '17.nicht(5)'
Wort wurde bereits besucht

Starte bei Wort '18.schneiden(9)'
-> 27.einmal(6) -> 33.Stab(4) -> 37.zehn(4)
-> 41.sich(4) -> 45.dem(3) -> 48.nicht(5)
Wort wurde bereits besucht

Starte bei Wort '19.bis(3)'
Wort wurde bereits besucht

Starte bei Wort '20.an(2)'
Wort wurde bereits besucht

Starte bei Wort '21.ihr(3)'
-> 24.Das(3) -> 27.einmal(6) Wort wurde
bereits besucht

Starte bei Wort '22.seliges(7)'
Wort wurde bereits besucht

Starte bei Wort '23.Grab(4)'
-> 27.einmal(6) Wort wurde bereits besucht

Starte bei Wort '24.Das(3)'
Wort wurde bereits besucht

Starte bei Wort '25.war(3)'
-> 28.der(3) -> 31.Stolz(5) -> 36.sie(3) ->
39.gewandert(9) -> 48.nicht(5) Wort wurde
bereits besucht

Starte bei Wort '26.nun(3)'
-> 29.beiden(6) Wort wurde bereits besucht

Starte bei Wort '27.einmal(6)'
Wort wurde bereits besucht

Starte bei Wort '28.der(3)'
Wort wurde bereits besucht

Starte bei Wort '29.beiden(6)'
Wort wurde bereits besucht

Starte bei Wort '30.geheimer(8)'
-> 38.Lichtjahre(10) Wort wurde bereits
besucht

Starte bei Wort '31.Stolz(5)'
Wort wurde bereits besucht

Starte bei Wort '32.und(3)'
-> 35.als(3) Wort wurde bereits besucht

Starte bei Wort '33.Stab(4)'
Wort wurde bereits besucht

Starte bei Wort '34.Doch(4)'
-> 38.Lichtjahre(10) Wort wurde bereits
besucht

Starte bei Wort '35.als(3)'
Wort wurde bereits besucht

Starte bei Wort '36.sie(3)'
Wort wurde bereits besucht

Starte bei Wort '37.zehn(4)'
Wort wurde bereits besucht

Starte bei Wort '38.Lichtjahre(10)'
Wort wurde bereits besucht

Starte bei Wort '39.gewandert(9)'
Wort wurde bereits besucht

Starte bei Wort '40.neben(5)'
-> 45.dem(3) Wort wurde bereits besucht

Starte bei Wort '41.sich(4)'
Wort wurde bereits besucht

Starte bei Wort '42.hin(3)'
-> 45.dem(3) Wort wurde bereits besucht

{85}
['verschlang']

Besuchte Indizes: [0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
```

```
41, 42, 45, 48, 53, 57, 60, 65, 68, 76, 79,
82, 85]
```

Außer beim ersten Wort wird jede Folge abgebrochen, weil ein Wort schon besucht wurde.

Quellcode

junioraufgabe1.py – class Parallelen

REGEX_WORD

Dieses Klassenattribut speichert den Regulären Ausdruck als `re.compile(r"\w+")`, um die Wörter zu suchen.

__init__(words, index_second_half)

Der Parameter `words` ist eine Liste mit allen Wörtern, der zweite Parameter gibt den Index an, ab dem die zweite Hälfte beginnt (also die Länge der ersten Hälfte).

Hier werden folgende Attribute zugewiesen:

`words`	Liste aller Wörter, identisch mit Parameter `words`
`word_count`	Anzahl der Wörter
`visited_indices`	`set` mit allen besuchten Indizes
`end_indices`	`set` mit allen End-Indizes
`index_second_half`	identisch mit gleichnamigem Parameter

from_file(fileobj, index_second_half)

Erstellt ein Parallelen-Objekt aus einer Datei. Der zweite Parameter wird an den Konstruktor übergeben, der Text des ersten wird genutzt, um die Liste mit Wörtern zu erhalten: `Parallelen.REGEX_WORD.findall(fileobj.read())`.

check()

```
for start_index in range(self.index_second_half):
    print(f"\nStarte bei Wort '{self._get_word_info(start_index)}'")
    try:
        # Starte Folge von start_index und füge End-Index hinzu
        self.end_indices.add(self._walk_from_word(start_index))
        print()
    except ValueError:
        # Abbruch, weil ein Wort bereits besucht wurde
        pass
```

_walk_from_word(index)

```
if index in self.visited_indices:
    print(f"Wort wurde bereits besucht")
    raise ValueError  # abgefangen in 'check()'
self.visited_indices.add(index)
new_index = index + len(self.words[index])  # Index für nächstes Wort
if new_index >= self.word_count:
    # neuer Index wäre zu groß, End-Wort/End-Index ist gefunden
    return index
print("->", self._get_word_info(new_index), "", end="")
# von neuem Index aus weitergehen
return self._walk_from_word(new_index)
```

`_get_word_info(index)`

Funktion, um ein Wort mit Index und Länge eingefärbt darzustellen:

```
word = self.words[index]
return f"{index}.\033[33m{word}\033[0m({len(word)})"
```