4. Aufgabe

„Auto-Scrabble“- Dokumentation

36. Bundeswettbewerb Informatik 2017/18 - 1.Runde

Sebastian Baron, Simon Fiebich, Lukas Rost und Max Woithe

Team-ID: 00020

27. November 2017

Aufgabe a) Ja, der Name "TIMO" kann nicht mit Autokennzeichen geschrieben werden, da weder die Ortskennung "T", noch "TI" noch "TIM" existiert.

Aufgabe b) weitere Worte, die nicht mit Autokennzeichen geschrieben werden können:

- 4 Buchstaben: URAN
- 3 Buchstaben: OMA
- 2 Buchstaben: IN

Aufgabe c/d) **Lösungsidee**

Auf ein Autokennzeichen passen, nach den gegebenen Bildungsvorschriften, stets mindestens zwei und maximal fünf Buchstaben, unabhängig von der jeweiligen Aufteilung in Ortskennung und Erkennungszeichen. Auf Grundlage der Länge (= Anzahl an Zeichen) des zu prüfenden Wortes kann man nun mithilfe dieser Überlegungen die theoretische Minimal- und Maximalzahl an Kennzeichen ermitteln, auf die die Zeichen des Wortes verteilt werden müssen. Für jede so ermittelte hypothetisch mögliche Anzahl an Kennzeichen kann man nun alle möglichen Kombinationen für eine Verteilung von Buchstaben auf diese Kennzeichen aufstellen:

Bsp.: Anzahl Kennzeichen==3:

1.) 2+2+2	5.) 2+3+2	9.) 2+4+2	13.) 2+5+2
2.) 2+2+3	6.) 2+3+3	10.) 2+4+3	14.) 2+5+3
3.) 2+2+4	7.) 2+3+4	11.) 2+4+4	15.) 2+5+4
4.) 2+2+5	8.) 2+3+5	12.) 2+4+5	16.) 2+5+5

---

x 4 (→ analog für 3, 4 bzw. 5)

Jede dieser Kombinationen, bei der die Gesamtanzahl an Zeichen auf allen Kennzeichen der Länge des zu prüfenden Wortes entspricht, gelangt in die engere Auswahl.

Anhand jeder dieser übrigen Kombinationen muss nun das zu prüfende Wort auf die Kennzeichen und innerhalb dieser in Ortskennung und Erkennungszeichen aufgeteilt und überprüft werden. Dabei muss beachtet werden, dass es bei 3 bzw. 4 Buchstaben auf einem Kennzeichen jeweils zwei Möglichkeiten zur Aufteilung dieser Zeichen in Ortskennung und Erkennungszeichen gibt (3: 2-1 oder 1-2; 4: 3-1 oder 2-2). Bei 2 bzw. 4 Zeichen ist die Aufteilung dagegen eindeutig (2: 1-1; 5: 3-2). Damit wird an dieser Stelle bereits ausgeschlossen, dass sich am Ende nicht voll ausgenutzte Kennzeichen ergeben. Bei diesem Schritt muss außerdem beachtet werden, dass das Erkennungszeichen keine Umlaute enthalten darf. Nun wird das zu prüfende Wort auf die Kennzeichen aufgeteilt und sämtliche Ortskennungen auf ihr Vorhandensein überprüft. So bleiben zum Schluss alle korrekten Lösungen übrig.

Es gibt zwei Sonderfälle, in denen sofort abgebrochen werden kann:

1. Das Wort enthält nur einen oder weniger Buchstaben. Da die Ortskennung und das Erkennungszeichen je mindestens einen Buchstaben enthalten müssen, lassen sich mit solchen Wörtern keine voll ausgenutzten Kennzeichen bilden.
2. Das Wort enthält ein „Ä“. Da das Erkennungszeichen keine Umlaute enthalten darf und es keine Ortskennung gibt, die ein „Ä“ enthält, lassen sich solche Wörter nicht mit Kennzeichen schreiben.

Umsetzung

Die Lösungsidee wird in Python 3 implementiert. Die Mindest- bzw. Maximalanzahl an Kennzeichen, die verwendet werden könnten, ist jeweils die erste natürliche Zahl, deren Zwei- bzw. Fünffaches größer oder gleich der Länge des zu prüfenden Wortes ist. Für jede Zahl zwischen (einschließlich) der so ermittelten Mindest- und Maximalanzahl wird nun zunächst die Anfangskombination mit zwei Buchstaben pro Kennzeichen (2+2+2+...+2+2) gebildet (die Anzahl der 2en entspricht also der aktuell überprüften Anzahl an Kennzeichen.

Nun wird eine theoretisch ins Unendliche laufende while-Schleife eröffnet, in der drei wesentliche Schritte ablaufen:

1. Die aktuelle Kombination (im Programmcode als `summe` bezeichnet) wird überprüft, ob bereits die Endkombination 5+5+5+...+5+5 erreicht wurde. Falls dies der Fall ist, wird die while-Schleife abgebrochen.
2. Überprüfe die aktuelle Kombination dahingehend, ob die Summe der so auf den Kennzeichen verteilten Buchstaben überhaupt der Länge des zu prüfenden Wortes entsprechen. Falls dies der Fall ist, wird die aktuelle Kombination zu einem String umgeformt (aus 2+4+5+3 wird also bspw.: '2453') und zur Liste `solutions` hinzugefügt.
3. Erhöhe die zu prüfende Kombination. Dabei wird das letzte Element (also die Anzahl an Buchstaben auf dem letzten Kennzeichen um 1 erhöht. Falls das letzte Element bereits die 5 erreicht hat, wird dieses auf 2 zurückgesetzt und das vorletzte um 1 erhöht. Falls das vorletzte Element bereits die 5 erreicht hat, wird dieses auf 2 zurückgesetzt und das vorvorletzte um 1 erhöht usw. usw.

Nach dem Durchlaufen dieser while-Schleife (die ja für jede mögliche Anzahl an Kennzeichen durchlaufen wird) sind nun alle möglichen Kombinationen, die der Länge des zu prüfenden Wortes entsprechen, in der Liste `solutions` gespeichert.

Jede dieser vorläufigen Lösungen wird nun auf die Möglichkeit ihrer Existenz hin überprüft. Das erste Zeichen der aktuellen `solution` wird ausgelesen und abhängig davon nach den in der Lösungsidee genannten Kriterien überprüft. Das Programm arbeitet bei der Überprüfung mit einer Kopie des zu prüfenden Wortes (`pruefwort` → `pruef`). Nach einer erfolgreichen Überprüfung wird die Form des ermittelten Kennzeichens sowie die zugehörigen Buchstaben des zu prüfenden Wortes als String einer Ergebnis-Liste hinzugefügt, das erste Zeichen der `solution` sowie die ersten Zeichen des zu prüfenden Wortes (abhängig von der Anzahl an Zeichen auf dem ersten Nummernschild) werden gelöscht und die Prüfschleife beginnt für das nächste Kennzeichen der `solution`. Ist eine Überprüfung einer bei einem Kennzeichen fehlerhaft, so wird die Überprüfung dieser `solution` abgebrochen und die Ergebnis-Liste wird bis zum Ende des letzten Ergebnisses zurückgesetzt. Am Ende einer erfolgreichen Überprüfung einer `solution` wird das Listenelement `'newsolution'` der Ergebnis-Liste hinzugefügt um eine solche Zurücksetzung, durch eine Trennung der einzelnen Ergebnisse voneinander, möglich zu machen.

Um eine benutzerfreundliche Ausgabe der Ergebnisse zu gewährleisten, muss die während der Überprüfung erstellte Ergebnis-Liste zum Schluss korrekt ausgelesen werden: Es wird jedes Listenelement nacheinander gelesen und daraufhin eine Ausgabe produziert: Eine Lösung wird stets von einem `'newsolution'` in der Liste begrenzt. In diesem Fall wird eine Trennlinie ausgegeben. Danach folgt immer ein Element mit zwei Zahlen (2, 3, 4 oder 5). Diese geben die Form des Kennzeichens an. Danach folgen die Buchstaben, die auf eben dieses Kennzeichen geschrieben werden sollen, die je nach der Art des Kennzeichens in Ortskennung und Erkennungszeichen getrennt und ebenfalls ausgegeben werden.

Wurde durch das Programm keine Lösung ermittelt, besteht die Ergebnis-Liste lediglich aus einem `'newsolution'`, sodass dieser Fall ebenfalls ausgegeben werden kann.

Die Zahlen auf den Kennzeichen werden vernachlässigt und deshalb auch nicht in der Lösung ausgegeben. Weiterhin wird angenommen, dass Umlaute nicht ersetzt (z.B. Ü -> UE) werden dürfen. Sollte dies beabsichtigt sein, müssen die Umlaute explizit in dieser Form eingegeben werden.

Beispiele

Das Programm liest das zu prüfende Wort aus einer externen .txt-Datei ein, in der nur ein einziges Wort in Großbuchstaben enthalten sein darf. Diese Datei kann über einen grafischen Tkinter-Auswahldialog ausgewählt werden. Einige Beispiele für diese Dateien sowie für die zugehörigen Ausgaben in der Python-Shell sind dem Quellcode und dieser Dokumentation beigelegt. Die Kürzelliste muss sich ebenfalls im gleichen Ordner befinden.

Weiterhin ist für die vom Bwinf bereitgestellten Beispiele jeweils eine verkürzte Form der Ausgabe in dieser Dokumentation abgedruckt. Diese enthält jeweils nur die erste Möglichkeit zur Schreibung mit Kennzeichen, sofern vorhanden.

Beispiel 1:

```
------------------------------------------------------------------
Wort: BUNDESWETTBEWERB
Wortlänge: 16

Es gibt 28 Möglichkeiten, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 1-1
B - U
Kennzeichen 2-2
ND - ES
Kennzeichen 2-2
WE - TT
Kennzeichen 1-1
B - E
Kennzeichen 2-2
WE - RB
------------------------------------------------------------------
```

Beispiel 2:

```
------------------------------------------------------------------
Wort: INFORMATIK
Wortlänge: 10

Es gibt 1 Möglichkeit, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 2-2
IN - FO
Kennzeichen 1-1
R - M
Kennzeichen 2-2
AT - IK
------------------------------------------------------------------
```

Beispiel 3:

```
------------------------------------------------------------------
Wort: BIBER
Wortlänge: 5

Es gibt 2 Möglichkeiten, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 1-1
B - I
Kennzeichen 1-2
B - ER
------------------------------------------------------------------
```

Beispiel 4:

```
------------------------------------------------------------------
Wort: CLINTON
Wortlänge: 7

Es gibt 1 Möglichkeit, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 1-2
C - LI
Kennzeichen 2-2
NT - ON
------------------------------------------------------------------
```

Beispiel 5:

```
------------------------------------------------------------------
Wort: ETHERNET
Wortlänge: 8

Es gibt 6 Möglichkeiten, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 1-1
E - T
Kennzeichen 1-1
H - E
Kennzeichen 2-2
RN - ET
------------------------------------------------------------------
```

Beispiel 6:

```
------------------------------------------------------------------
Wort: SOFTWARE
Wortlänge: 8

Es gibt 6 Möglichkeiten, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 2-2
SO - FT
Kennzeichen 2-2
WA - RE
------------------------------------------------------------------
```

Beispiel 7:

```
------------------------------------------------------------------
Wort: TRUMP
Wortlänge: 5

Es gibt 1 Möglichkeit, das Wort mit Autokennzeichen zu schreiben.
Das Wort kann mit folgenden Kennzeichen geschrieben werden:
------------------------------------------------------------------
------------------------------------------------------------------
Kennzeichen 2-1
TR - U
Kennzeichen 1-1
M - P
------------------------------------------------------------------
```

Die Beispiele „TSCHÜSS“ und „DONAUDAMPFSCHIFFFAHRTSKAPITÄNSMÜTZE“ (aufgrund des zweiten Sonderfalls) können nicht mit Kennzeichen geschrieben werden. Die Beispiele „LLANFAIRPWLLGWYNGYLLGOGERYCHWYRNDROBWLLLLANTYSILIOGOGOGOCH“, „RINDFLEISCHETIKETTIERUNGSÜBERWACHUNGSAUFGABENÜBERTRAGUNGSGESETZ“ sowie „VERKEHRSWEGEPLANUNGSBESCHLEUNIGUNGSGESETZ“ konnten von unserem Programm nicht in einer annehmbaren Zeit bearbeitet werden, da ab ca. 20 Buchstaben die Anzahl der möglichen und damit

auch auszuprobierenden Kennzeichenkombinationen in der ersten Schleife exponentiell sehr stark steigt und eine ebenfalls exponentielle Laufzeiterhöhung des Programms zur Folge hat. Auch erhöht sich dadurch der Speicherverbrauch und die Prozessorbelastung.

Die Anzahl der zu bearbeitenden Kennzeichenkombinationen ergibt sich aus folgender Gleichung abhängig von der Minimal- und Maximalanzahl der Kennzeichen:

$$Kennzeichenkombinationen = \sum_{n=Minimalanzahl}^{Maximalanzahl} 4^n$$

Hieraus kann man ableiten, dass aufgrund der exponentiellen Abhängigkeit die Laufzeit zwangsläufig ins Unendliche steigen muss. Da die vorläufigen Lösungen erst in einem zweiten Schritt auf ihre tatsächliche Existenz überprüft werden, lässt sich die Zahl der Schleifendurchläufe nur schwer verkürzen, indem man nach dem ersten gültigen Kennzeichen abbricht, da die ermittelte vorläufige Lösung nicht unbedingt auch in der Realität existieren muss.

Quellcode

```
import sys,os
from tkinter import Tk
from tkinter.filedialog import askopenfilename

Tk().withdraw()
prueffilename = askopenfilename(initialdir =
os.path.dirname(os.path.realpath(__file__)),title = "Eingabedatei
auswählen",filetypes = (("Textdateien","*.txt"),("Alle Dateien","*.*")))

fileliste=open('kuerzelliste.txt','r',encoding="utf-8")
kuerzelliste=[]
for line in fileliste:
    kuerzelliste=kuerzelliste+[(line.rstrip())]
fileliste.close()
# Kürzelliste wird aus entsprechender Datei ausgelesen --> Python-Liste

filepruef=open(prueffilename,'r',encoding="utf-8")
for line in filepruef:
    pruef=line.rstrip()
filepruef.close()
# zu überprüfendes Wort wird aus entsprechender Datei ausgelesen --> String

if len(pruef)<=1 or "Ä" in pruef:
    print('-----------------------------------------------------------------')
    print('Wort:',pruef)
    print('Wortlänge:',len(pruef))
    print('')
    print('Es gibt keine Möglichkeit, das Wort mit Autokennzeichen zu
schreiben.')
    print('-----------------------------------------------------------------')
    sys.exit()
# Ausnahme für Wort mit einem Buchstaben oder Ä im Wort

for x in range(0,len(pruef)):
    if 5*x>=len(pruef):
        minaz=x
        break

for x in range(0,len(pruef)):
    if 2*x>=len(pruef):
        maxaz=x
        break                                    # Ermittlung der theoretischen
Minimal- und Maximalzahl an Kennzeichen (auf ein Kennzeichen passen zwischen
zwei und fünf Buchstaben)

solutions=[]
for x in range(minaz,maxaz+1):
    summe=[]
```

```
    while x!=0:
        summe=summe+[2]
        x=x-1
# Bildung der Anfangskombination 2+2+2+...+2+2
    while 1==1:
        pruefstring=''
        for a in range(0,len(summe)):
            if summe[a]==5:
                pruefstring=pruefstring+'a'
# für jeden Summanden==5 wird pruefstring wird um ein Zeichen verlängert
        if len(pruefstring)!=len(summe):
# prüfe, ob Endkombination 5+5+5+...+5+5 erreicht ist; wenn nicht:
            s=0
            for y in range(0,len(summe)):
                s=s+int(summe[y])
# berechne momentane Summe an Buchstaben auf den Kennzeichen
            if s==len(pruef):
                summestring=''
                for i in summe:
                    summestring=summestring+str(i)
                solutions=solutions+[summestring]
# wenn Summe der Länge des zu überprüfenden Wortes entspricht, speichere
aktuelle Kombi
            z=1
            while z<=len(summe):
                if summe[len(summe)-z]!=5:
                    summe[len(summe)-z]=(summe[len(summe)-z])+1
                    break
                else:
                    summe[len(summe)-z]=2
                    z=z+1                                                    #
setze nächste Kombination (hintere Stelle+1; falls hintere Stelle==5: hintere
Stelle=2 und vorletzte Stelle+1)
        else:
            s=0
            for y in range(0,len(summe)):
                s=s+int(summe[y])
            if s==len(pruef):
                summestring=''
                for i in summe:
                    summestring=summestring+str(i)
                solutions=solutions+[summestring]
            break
# überprüfe noch diese Kombination, danach Ende

azlsg=0
ergebnis=['newsolution']

for x in solutions:
# für jede Kombination, die theoretisch eine korrekte Buchstabenverteilung auf
dem Kennzeichen angibt
    pruefwort=pruef
    solution=str(x)
    true=1
    while len(pruefwort)!=0 and true!=0:
        while len(solution)!=0:
            true=0
            if solution[0]=='2':
                for b in range(0,len(kuerzelliste)):
                    if pruefwort[0]==kuerzelliste[b] and pruefwort[1]!='Ä' and
pruefwort[1]!='Ö' and pruefwort[1]!='Ü':
                        ergebnis=ergebnis+[11]
                        ergebnis=ergebnis+[pruefwort[0]+pruefwort[1]]
                        true=1
                        pruefwort=pruefwort[2:]
                        break

            elif solution[0]=='3':
```

```
                for b in range(0,len(kuerzelliste)):
                    if pruefwort[0]==kuerzelliste[b] and pruefwort[1]!='Ä' and
pruefwort[1]!='Ö' and pruefwort[1]!='Ü' and pruefwort[2]!='Ä' and
pruefwort[2]!='Ö' and pruefwort[2]!='Ü':
                        ergebnis=ergebnis+[12]
                        ergebnis=ergebnis+[pruefwort[0]+pruefwort[1:3]]
                        true=1
                        pruefwort=pruefwort[3:]
                        break
                if true==0:
                    for b in range(0,len(kuerzelliste)):
                        if pruefwort[0:2]==kuerzelliste[b] and pruefwort[2]!='Ä'
and pruefwort[2]!='Ö' and pruefwort[2]!='Ü':
                            ergebnis=ergebnis+[21]
                            ergebnis=ergebnis+[pruefwort[0:2]+pruefwort[2]]
                            true=1
                            pruefwort=pruefwort[3:]
                            break

            elif solution[0]=='4':
                for b in range(0,len(kuerzelliste)):
                    if pruefwort[0:2]==kuerzelliste[b] and pruefwort[2]!='Ä' and
pruefwort[2]!='Ö' and pruefwort[2]!='Ü' and pruefwort[3]!='Ä' and
pruefwort[3]!='Ö' and pruefwort[3]!='Ü':
                        ergebnis=ergebnis+[22]
                        ergebnis=ergebnis+[pruefwort[0:2]+pruefwort[2:4]]
                        true=1
                        pruefwort=pruefwort[4:]
                        break
                if true==0:
                    for b in range(0,len(kuerzelliste)):
                        if pruefwort[0:3]==kuerzelliste[b] and pruefwort[3]!='Ä'
and pruefwort[3]!='Ö' and pruefwort[3]!='Ü':
                            ergebnis=ergebnis+[31]
                            ergebnis=ergebnis+[pruefwort[0:3]+pruefwort[3]]
                            true=1
                            pruefwort=pruefwort[4:]
                            break

            elif solution[0]=='5':
                for b in range(0,len(kuerzelliste)):
                    if pruefwort[0:3]==kuerzelliste[b] and pruefwort[3]!='Ä' and
pruefwort[3]!='Ö' and pruefwort[3]!='Ü' and pruefwort[4]!='Ä' and
pruefwort[4]!='Ö' and pruefwort[4]!='Ü':
                        ergebnis=ergebnis+[32]
                        ergebnis=ergebnis+[pruefwort[0:3]+pruefwort[3:5]]
                        true=1
                        pruefwort=pruefwort[5:]
                        break                                                #
Überprüfe, ob die (je nach Länge bei der entsprechenden Mglk.) Ortskennung des
ersten Kennzeichens vorhanden ist

            if true==0:
                while ergebnis[len(ergebnis)-1]!='newsolution':
                    ergebnis.pop(len(ergebnis)-1)                            # falls
keine Lösung für diese Kennzeichenkombi, setze die Ergebnis-Liste entsprechend
zurück
                break
            else:
                solution=solution[1:]                                        # Lösche
momentanes erstes Kennzeichen (abgehakt)
        if true==1:
            ergebnis=ergebnis+['newsolution']
            azlsg=azlsg+1

print('-----------------------------------------------------------------')
print('Wort:',pruef)
print('Wortlänge:',len(pruef))
```

```
print('')
if azlsg==0:
    print('Es gibt keine Möglichkeit, das Wort mit Autokennzeichen zu
schreiben.')
    print('-----------------------------------------------------------------')
    sys.exit()
elif azlsg==1:
    print('Es gibt 1 Möglichkeit, das Wort mit Autokennzeichen zu schreiben.')
else:
    print('Es gibt',azlsg,'Möglichkeiten, das Wort mit Autokennzeichen zu
schreiben.')
if azlsg!=0:
    print('Das Wort kann mit folgenden Kennzeichen geschrieben werden:')
    print('-----------------------------------------------------------------')

for x in ergebnis:
    if x==11:
        print('Kennzeichen 1-1')
        form='11'
    elif x==12:
        print('Kennzeichen 1-2')
        form='12'
    elif x==21:
        print('Kennzeichen 2-1')
        form='21'
    elif x==22:
        print('Kennzeichen 2-2')
        form='22'
    elif x==31:
        print('Kennzeichen 3-1')
        form='31'
    elif x==32:
        print('Kennzeichen 3-2')
        form='32'
    elif x=='newsolution':
        print('-----------------------------------------------------------------
')
    else:
        if form=='11':
            print(x[0],'-',x[1])
        elif form=='12':
            print(x[0],'-',x[1:3])
        elif form=='21':
            print(x[0:2],'-',x[2])
        elif form=='22':
            print(x[0:2],'-',x[2:4])
        elif form=='31':
            print(x[0:3],'-',x[3])
        elif form=='32':
            print(x[0:3],'-',x[3:5])

if ergebnis==['newsolution']:
    print('')
    print('KEINE SCHREIBUNG MIT KENNZEICHEN MÖGLICH!!')
    print('')
# Interpretation des Ergebnis-Strings und benutzerfreundliche Ausgabe
```