Aufgabe 2: Schwierigkeiten

Team-ID: 00079

Team: Constantin Reinhold

Bearbeiter/-innen dieser Aufgabe:
Constantin Reinhold

16. November 2024

Inhaltsverzeichnis

Lösungsidee

Die Idee ist es jeder Aufgabe einen Wert zuzuordnen basierend darauf wie schwierig die Aufgabe ist. Das können wir daran bestimmen and welcher Stelle in der Aufzählung die Aufgabe steht. Manche Aufgaben kommen mehrmals vor. Dies müssen wir unsere Wertung mit einbeziehen. Falls zwei oder mehr Werte gleich sind wollen wir vergleichen welcher den höheren Maximalwert hat. Falls zwei Aufgabe auch in disem Test gleich sind überprüfen wir ihren Minimumwert. Und falls die Aufgaben dann immernoch gleich sind betrachten wir sie als gleich schwer und die Reihenfolge ist egal.

Umsetzung

Als erstes nutzen wir Loops und String-Manipulation um zwei Arrays zu erhalten die unsere Aufzählungen ohne Sonderzeichen und unsere gesuchten Werte enthalten.

Danach nutzen wir Listen-Iteration um für jeden Buchstaben die Summe aller Indexes, den höchsten und niedrigsten Index als auch die Häufigkeit zu finden.

Der Wert (W) jedes Buchstaben wird durch die folgende Formel angegeben:

$W = \frac{i}{o}$, wobei $i = \sum indexes$ und $o = occurrences$.

Danach nutzen wir die eingebaute Sortierfunktion und einen lambda key um unsere Liste der gesuchten Aufgaben in dieser Reihenfolge zu sortieren.

1. Wert jedes Buchstabens, aufsteigend (von klein nach groß)

2. Höchster aufgenommener Index, aufsteigend

3. Niedrigster aufgenommener Index, absteigend (von groß nach klein)

Somit ist die gesuchte Aufgabenliste nach der Schwierigkeit, absteigend, sortiert.

Beispiele

Da die Struktur für das einlesen der Zeilen immer die selbe ist und keine interessanten Informationen bietet sind die folgenden Beispiel Erklärungen folgendermaßen aufgebaut. Die ersten drei Zeilen bestehen aus dem value_dictionary welches alle wichtigen Daten enthält, dann dem worth_dictionary, was die berechneten allgemeinen Schwierigkeitswerte umfasst und dann die Ausgabe des Programmes. Danach werden die Beispiele in Sonderfällen erklärt.

Schwierigkeiten0:

{'B': 0, 'B_highest': 0, 'B_lowest': 0, 'B_occurrence': 1, 'A': 1, 'A_highest': 1, 'A_lowest': 0, 'A_occurrence': 2, 'D': 4, 'D_highest': 2, 'D_lowest': 0, 'D_occurrence': 3, 'F': 5, 'F_highest': 3, 'F_lowest': 0, 'F_occurrence': 3, 'G': 2, 'G_highest': 2, 'G_lowest': 0, 'G_occurrence': 2, 'E': 1, 'E_highest': 1, 'E_lowest': 0, 'E_occurrence': 1, 'C': 5, 'C_highest': 3, 'C_lowest': 0, 'C_occurrence': 2}

{'B': 0.0, 'C': 2.5, 'D': 1.33, 'E': 1.0, 'F': 1.67}

['B', 'E', 'D', 'F', 'C']

Wie wir in Zeile im zweiten Dictionary sehen können haben alle Aufgaben einen unterschiedlichen Schwierigkeitsgrad. Die Sortierung ist hier also einfach von klein nach groß. Also wie in Zeile 3 zu sehen ist, *['B', 'E', 'D', 'F', 'C']*.

Schwierigkeiten1:

{'A': 0, 'A_highest': 0, 'A_lowest': 0, 'A_occurrence': 2, 'B': 1, 'B_highest': 1, 'B_lowest': 0, 'B_occurrence': 1, 'C': 4, 'C_highest': 2, 'C_lowest': 0, 'C_occurrence': 3, 'D': 5, 'D_highest': 3, 'D_lowest': 0, 'D_occurrence': 2, 'E': 1, 'E_highest': 1, 'E_lowest': 0, 'E_occurrence': 2, 'F': 3, 'F_highest': 2, 'F_lowest': 0, 'F_occurrence': 2, 'G': 1, 'G_highest': 1, 'G_lowest': 0, 'G_occurrence': 1}

{'A': 0.0, 'C': 1.33, 'D': 2.5, 'F': 1.5, 'G': 1.0}

['A', 'G', 'C', 'F', 'D']

Genau wie Beispiel Schwierigkeiten0.

Schwierigkeiten2:

{'A': 2, 'A_highest': 2, 'A_lowest': 0, 'A_occurrence': 2, 'B': 1, 'B_highest': 1, 'B_lowest': 0, 'B_occurrence': 1, 'C': 4, 'C_highest': 2, 'C_lowest': 0, 'C_occurrence': 3, 'E': 1, 'E_highest': 1,

'E_lowest': 0, 'E_occurrence': 4, 'D': 2, 'D_highest': 2, 'D_lowest': 0, 'D_occurrence': 2, 'F': 2, 'F_highest': 1, 'F_lowest': 0, 'F_occurrence': 2, 'G': 2, 'G_highest': 2, 'G_lowest': 0, 'G_occurrence': 1, 'H': 3, 'H_highest': 3, 'H_lowest': 0, 'H_occurrence': 2}

{'A': 1.0, 'B': 1.0, 'D': 1.0, 'E': 0.25, 'F': 1.0, 'G': 2.0}

['E', 'B', 'F', 'A', 'D', 'G']

In diesem Fall haben wir 4 Aufgaben mit dem Schwierigkeitswert 1,0. Wir sortieren daher nach dem höchsten und dann dem niedrigsten Wert welche im ersten Array als *'Aufgabe_highest'* und *'Aufgabe_lowest'* vermerkt sind. Wir sehen im ersten Array, dass sowohl B und F einen höchten Wert von 1 und niedrigsten Wert von 0 haben. Es ist daher egal in welcher Reihenfolge B und F kommen, sie sind jedoch einfacher als A und D die beide einen höchsten Wert von 2 und niedrigsten Wert von 0 haben. E und G sind eindeutig.

Schwierigkeiten3:

{'A': 3, 'A_highest': 3, 'A_lowest': 0, 'A_occurrence': 2, 'B': 2, 'B_highest': 1, 'B_lowest': 0, 'B_occurrence': 2, 'C': 2, 'C_highest': 2, 'C_lowest': 0, 'C_occurrence': 2, 'D': 2, 'D_highest': 2, 'D_lowest': 0, 'D_occurrence': 1, 'E': 4, 'E_highest': 4, 'E_lowest': 0, 'E_occurrence': 1, 'F': 5, 'F_highest': 5, 'F_lowest': 0, 'F_occurrence': 1, 'G': 6, 'G_highest': 6, 'G_lowest': 0, 'G_occurrence': 1, 'H': 2, 'H_highest': 2, 'H_lowest': 0, 'H_occurrence': 2, 'I': 1, 'I_highest': 1, 'I_lowest': 0, 'I_occurrence': 2, 'J': 2, 'J_highest': 2, 'J_lowest': 0, 'J_occurrence': 1, 'K': 3, 'K_highest': 3, 'K_lowest': 0, 'K_occurrence': 1, 'L': 1, 'L_highest': 1, 'L_lowest': 0, 'L_occurrence': 1, 'M': 1, 'M_highest': 1, 'M_lowest': 0, 'M_occurrence': 2, 'N': 1, 'N_highest': 1, 'N_lowest': 0, 'N_occurrence': 2}

{'A': 1.5, 'B': 1.0, 'C': 1.0, 'D': 2.0, 'E': 4.0, 'F': 5.0, 'G': 6.0, 'H': 1.0, 'I': 0.5, 'J': 2.0, 'K': 3.0, 'L': 1.0, 'M': 0.5, 'N': 0.5}

['I', 'M', 'N', 'B', 'L', 'C', 'H', 'A', 'D', 'J', 'K', 'E', 'F', 'G']

Gleiche Vorgehensweise wie in Schwierigkeiten2.

Schwierigkeiten4:

{'A': 0, 'A_highest': 0, 'A_lowest': 0, 'A_occurrence': 2, 'B': 2, 'B_highest': 1, 'B_lowest': 0, 'B_occurrence': 3, 'C': 3, 'C_highest': 2, 'C_lowest': 0, 'C_occurrence': 2, 'D': 8, 'D_highest': 3, 'D_lowest': 0, 'D_occurrence': 3, 'E': 9, 'E_highest': 4, 'E_lowest': 0, 'E_occurrence': 3, 'J': 9, 'J_highest': 5, 'J_lowest': 0, 'J_occurrence': 2, 'I': 13, 'I_highest': 6, 'I_lowest': 0, 'I_occurrence': 3, 'H': 8, 'H_highest': 5, 'H_lowest': 0, 'H_occurrence': 3, 'K': 17, 'K_highest': 6, 'K_lowest': 0, 'K_occurrence': 7, 'S': 2, 'S_highest': 2, 'S_lowest': 0, 'S_occurrence': 3, 'G': 1, 'G_highest': 1, 'G_lowest': 0, 'G_occurrence': 2, 'O': 9, 'O_highest': 3, 'O_lowest': 0, 'O_occurrence': 4, 'M': 7, 'M_highest': 5, 'M_lowest': 0, 'M_occurrence': 3, 'N': 8, 'N_highest': 4, 'N_lowest': 0, 'N_occurrence': 3, 'P': 2, 'P_highest': 2, 'P_lowest': 0, 'P_occurrence': 3, 'Q': 2, 'Q_highest': 1, 'Q_lowest': 0, 'Q_occurrence': 2, 'R': 2, 'R_highest': 2, 'R_lowest': 0, 'R_occurrence': 2, 'F': 5,

'F_highest': 3, 'F_lowest': 0, 'F_occurrence': 4, 'T': 12, 'T_highest': 4, 'T_lowest': 0, 'T_occurrence': 5, 'U': 15, 'U_highest': 8, 'U_lowest': 0, 'U_occurrence': 3, 'V': 3, 'V_highest': 3, 'V_lowest': 0, 'V_occurrence': 2, 'W': 12, 'W_highest': 5, 'W_lowest': 0, 'W_occurrence': 5, 'Z': 12, 'Z_highest': 5, 'Z_lowest': 0, 'Z_occurrence': 5, 'Y': 10, 'Y_highest': 7, 'Y_lowest': 0, 'Y_occurrence': 3, 'X': 11, 'X_highest': 6, 'X_lowest': 0, 'X_occurrence': 3, 'L': 3, 'L_highest': 3, 'L_lowest': 0, 'L_occurrence': 1}

{'B': 0.67, 'W': 2.4, 'I': 4.33, 'N': 2.67, 'F': 1.25}

['B', 'F', 'W', 'N', 'I']

Dieses Beispiel scheint zwar auf den ersten Blick kompliziert, da es das ganze Alphabet umfasst. Die Schwierigkeitswerte für alle gesuchten Werte B, W, I, N, F sind jedoch alle eindeutig.

Schwierigkeiten5:

{'H': 0, 'H_highest': 0, 'H_lowest': 0, 'H_occurrence': 1, 'S': 2, 'S_highest': 1, 'S_lowest': 0, 'S_occurrence': 4, 'C': 2, 'C_highest': 2, 'C_lowest': 0, 'C_occurrence': 2, 'A': 7, 'A_highest': 4, 'A_lowest': 0, 'A_occurrence': 2, 'G': 5, 'G_highest': 4, 'G_lowest': 0, 'G_occurrence': 2, 'O': 3, 'O_highest': 1, 'O_lowest': 0, 'O_occurrence': 4, 'J': 2, 'J_highest': 2, 'J_lowest': 0, 'J_occurrence': 1, 'L': 4, 'L_highest': 3, 'L_lowest': 0, 'L_occurrence': 2, 'F': 8, 'F_highest': 4, 'F_lowest': 0, 'F_occurrence': 3, 'M': 6, 'M_highest': 4, 'M_lowest': 0, 'M_occurrence': 4, 'X': 13, 'X_highest': 3, 'X_lowest': 0, 'X_occurrence': 5, 'D': 7, 'D_highest': 4, 'D_lowest': 0, 'D_occurrence': 3, 'U': 9, 'U_highest': 4, 'U_lowest': 0, 'U_occurrence': 3, 'E': 2, 'E_highest': 2, 'E_lowest': 0, 'E_occurrence': 2, 'N': 3, 'N_highest': 3, 'N_lowest': 0, 'N_occurrence': 1, 'B': 4, 'B_highest': 4, 'B_lowest': 0, 'B_occurrence': 2, 'P': 6, 'P_highest': 4, 'P_lowest': 0, 'P_occurrence': 2, 'R': 0, 'R_highest': 0, 'R_lowest': 0, 'R_occurrence': 1, 'T': 7, 'T_highest': 4, 'T_lowest': 0, 'T_occurrence': 2, 'V': 3, 'V_highest': 3, 'V_lowest': 0, 'V_occurrence': 1, 'Z': 0, 'Z_highest': 0, 'Z_lowest': 0, 'Z_occurrence': 1, 'Q': 1, 'Q_highest': 1, 'Q_lowest': 0, 'Q_occurrence': 1, 'K': 2, 'K_highest': 2, 'K_lowest': 0, 'K_occurrence': 1, 'I': 6, 'I_highest': 4, 'I_lowest': 0, 'I_occurrence': 2, 'W': 1, 'W_highest': 1, 'W_lowest': 0, 'W_occurrence': 1, 'Y': 3, 'Y_highest': 3, 'Y_lowest': 0, 'Y_occurrence': 1}

{'A': 3.5, 'B': 2.0, 'C': 1.0, 'D': 2.33, 'E': 1.0, 'F': 2.67, 'G': 2.5, 'H': 0.0, 'I': 3.0, 'J': 2.0, 'K': 2.0, 'L': 2.0, 'M': 1.5, 'N': 3.0, 'O': 0.75, 'P': 3.0, 'Q': 1.0, 'R': 0.0, 'S': 0.5, 'T': 3.5, 'U': 3.0, 'V': 3.0, 'W': 1.0, 'X': 2.6, 'Y': 3.0, 'Z': 0.0}

['H', 'R', 'Z', 'S', 'O', 'Q', 'W', 'C', 'E', 'M', 'J', 'K', 'L', 'B', 'D', 'G', 'X', 'F', 'N', 'V', 'Y', 'I', 'P', 'U', 'A', 'T']

Dieses Beispiel funktioniert exakt wie Schwierigkeiten3. Es enthält einfach nur mehr Aufgaben. Die Komplexität bleibt jedoch die gleiche.

Sonderfälle:

Es gibt keine Sonderfälle mit dem das Programm nicht umgehen kann solange die Eingabe Dateien keine Aufgaben gesuchten Aufgaben in der letzten Zeile enthalten die nicht in den n Zeilen vorher erwähnt werden. Sogar der spezielste Fall indem alle Werte gleich sind kann das Programm ohne Probleme bewältigen.

Quellcode

```
filename = "schwierigkeitenX.txt"

def check_highest(old_value, new_value):
    if old_value < new_value:
        return new_value
    return old_value

def check_lowest(old_value, new_value):
    if old_value > new_value:
        return new_value
    return old_value

# Liest die Aufgabedatei ein
with open(filename, 'r') as file:
    content = file.readlines()
    content.pop(0) # Die erste Zeile kann wird nicht benötigt
    searched = []
    searched = content[-1].strip().split() # Speichert die gesuchten Werte ab
    content.pop() # Löscht die gesuchten Werte aus der Werteliste

# Räumt die Werteliste auf, entfernt zusätzliche Zeichen
processed_content = []
for item in content:
    subarray = item.replace('<', '').replace('\n', '').split()
    processed_content.append(subarray)

# Speichert alle wichtigen Werte ab
value_dictionary = {}

# Findet den höchsten und niedrigsten Auftritt und die Häufigkeit
# jedes Buchstabens in der Werteliste
for item in processed_content:
    for index, letter in enumerate(item):
        value_dictionary.update({letter: value_dictionary.get(letter, 0)+index})
        value_dictionary.update({f"{letter}_highest": check_highest(value_dictionary.get(f"{letter}
_highest", 0), index)})
        value_dictionary.update({f"{letter}_lowest": check_lowest(value_dictionary.get(f"{letter}
_lowest", 0), index)})
```

```
        value_dictionary.update({f"{letter}_occurrence": value_dictionary.get(f"{letter}
_occurrence", 0)+1})

# Kalkuliert den Wert jedes Buchstabens
worth_dictionary = {}
for letter in searched:
    worth_dictionary.update({letter:
round(value_dictionary.get(letter)/value_dictionary.get(f"{letter}_occurrence"),2)})

# Sortiert die Liste an gesuchten Werten basierend auf Wert, dann dem höchsten und dann
niedrigsten Auftritt
sorted_searched = sorted(
    searched,
    key = lambda item: (worth_dictionary.get(f"{item}"),
                        value_dictionary.get(f"{item}_highest", 0),
                        -value_dictionary.get(f"{item}_lowest", 0)
    )
)
# Gibt die sortierte Liste (Ergebnisse) aus
print(sorted_searched)
```