Junioraufgabe 2: Treffsicherheit

Teilnahme-ID: 313

Team: TobisMa

Bearbeiter/-in dieser Aufgabe:
Tobias Maurer

30. October 2021

Inhaltsverzeichnis

Lösungsidee

Meine Lösungsidee für die Aufgabe j2 ist die Folgende:

Als erstes muss man sich eine geeignete Darstellung für die Informationen festlegen. Die Darstellung aus den Beispieldaten, dass man für jede Terminpräferenz eine Zahl von 0-2 nimmt, ist sehr nützlich:

Art/Typ	Farbe	Wert
sehr guter Termin	grün	0
mäßiger Termin	gelb	1
blöder Termin	rot	2

Je kleiner Wert des Termins, desto angenehmer ist der Termin für die Person. Somit ist ein mäßiger Termin angenehmer als ein blöder Termin, da $1<2$ wahr ist. Termine sind gleich angenehm, wenn sie logischerweise denselben Wert haben. Z. B. ist ein „sehr guter Termin“ (Wert 0) gleichwertig mit einem „sehr gute[m] Termin“ (Wert 0), da $0=0$ eine wahre mathematische Aussage ist.

Da eine Person nur blöde Termine haben kann, ist es möglich, dass der gesuchte Termin einen oder mehrere blöde Termine hat. Ebenfalls kann eine Person nur sehr gute oder mäßige Termine haben,

weshalb es passieren kann, dass der gesuchte Termin nun alle Arten von Terminen haben kann. Hinzu kommt, dass eine Person auch verschiedene Arten von Terminen haben kann. Weil es einer Person möglich ist, dass diese nur eine Art von Termin hat bis zu einer Mischung von allen Arten von Terminen, muss die beste Terminart für jede Person ermittelt werden:

Personenname	Termin 1	Termin 2	Termin 3	Termin 4	Termin 5
Tom	blöd (=2)	blöd (=2)	blöd (=2)	mäßig (=1)	mäßig (=1)
Lisa	blöd (=2)	mäßig (=1)	sehr gut (=0)	blöd (=2)	mäßig (=1)

Um bei Tom seine beste Terminart zu bestimmen, geht man in der Tabelle von links nach rechts die Termine durch und schaut sich den Wert der Terminart an. Der erste Termin in der Zeile von Tom ist „blöd“ und hat einen Wert von 2. Nun geht man die Termine weiter durch und bei *Termin 4* stößt man auf eine andere Terminart. Diese hat den Wert 1, da $1<2$ ist die beste Terminart für Tom nun „mäßig“. Bei *Termin 5* ist zwar die gleichwertiger Terminart, aber diese hat keinen Effekt auf Toms beste Terminart.

Bei Lisa passiert im Prinzip zwischen blöden Terminen und mäßigen Terminen dasselbe wie bei Tom. Bei *Termin 3* hat Lisa nun einen sehr guten Termin, wodurch der Wert des mäßigen Termins genommen wird (momentan die angenehmste Terminart für Lisa) und der Wert von der Terminart bei Lisas *Termin 3*. Diese vergleicht man jetzt. Wenn der Wert von *Termin 3* (=0) kleiner ist als der Wert eines mäßigen Termins (=1), ist Lisas neu angenehmste Terminart „sehr guter Termin“.

Generell, um die Terminart für jede Person zu bestimmen, stellt man die Terminarten mithilfe deren Werten dar und erhält somit eine Folge an Zahlen. Das Minimum dieser Folge ist die angenehmste Terminart für die jeweilige Person.

Jedoch ist die Aufgabe den sogenannten besten Termin oder den Termin, wo die wenigsten Änderungen für einen besten Termin vorgenommen werden, zu finden. Die Bedingungen für den besten Termin sind, dass jede Person seine jeweilige angenehmste Terminart bei diesem Termin hat.

Somit geht man nach dem Bestimmen, der angenehmsten Terminart für jede Person, die Spalten, welche die Termine repräsentieren, in der gegebenen Präferenztabelle durch. Man zählt dann in der jeweiligen Spalte, welche Personen nicht ihre angenehmste Terminart haben. Das Ergebnis dieser Zählung ist gleichzeitig die Anzahl der Änderungen, die man machen muss, um diesen Termin zum bestem Termin zu machen. Man merkt sich die notwendigen Änderungen und zählt dasselbe in der nächsten Spalte aus. Es gibt nun drei Fälle, welche eintreten können:

1. Es sind mehr Änderungen notwendig: Man ignoriert diesen Termin/die Spalte, da man bereits einen Termin, welcher näher an einem besten Termin ist, gefunden hat.

2. Es sind gleich viele Änderungen notwendig: In diesem Fall merkt man sich diese Spalte zusätzlich für das Entscheiden, welchen Termin man zum besten Termin macht, aufgrund von den Persönlichkeiten der Personen, Vorlieben von einem selbst oder was auch immer.

3. Es sind weniger Änderungen notwendig: Man verwirft alle Spalten, die man sich gemerkt hat und merkt sich die neue Anzahl an Änderungen und die aktuelle Spalte.

Die oben genannten drei Fälle können ebenfalls für jede nachfolgende Spalte eintreten. Das Endresultat, welche Spalte bzw. Spalten für den besten Termin notwendig sind und wie viele Änderungen bei der Spalte bzw. den Spalten notwendig sind, muss man nun nur noch ausgeben/weitergeben/sich merken.

Umsetzung

Da in jeder Datei in der ersten Zeile irrelevante Informationen stehen (zumindest für die Programmiersprache Python) kann man diese Zeile beim Einlesen ignorieren. Die restlichen Zeilen beinhalten Zahlen mit einer Leerzeichentrennung. Diese „Tabelle“ stellt man nun mithilfe einer 2d Liste dar, bei der jede Zeile aus einer Liste von Zahlen ist und die Listen der Zeilen hängt man hintereinander in eine Liste. Hierbei steht jede Zeilenliste für eine Person.

Um die angenehmste Terminart für die Personen zu bestimmen iteriert man über die Listen der Zeilen und bestimmt deren Minimum. Diesen Wert kann man sich nun in einem Dictionary speichern. Als *key* verwendet man den Index der Liste ($zeilennummer - 2$ (erste Zeile ignoriert und index startet bei 0)) Der *value* für den *key* ist das Minimum.

Da wir die Tabelle als eine 2d Liste darstellen, über Das Format *List<List<int>>* greifen wir über folgende Systematik auf einen Wert zu: *tabelle*[<*zeilenindex*>][<*spaltenindex*>]. Um nun jeden Wert Spaltenweise aufzurufen. Benötigen wir eine 2d Schleife, welche für jeden Spaltenindex alle zeilenindexe durchgeht. Bevor man jetzt aber durch die Tabelle „iteriert“ muss man Standardwerte für die zu speichernden Werte fürs Ende festlegen. Zum einem wäre dies eine Liste für die Spalten, die am nächsten des besten Termins sind, und zum anderem die Anzahl der notwendigen Änderungen.

Für jede Spalte zählen wir nun aus, ob die Person einen angenehmeren Termin hat. In diesem Fall addiert man eins auf den Zähler, wie viele Änderungen für den besten Termin notwendig sind. Wenn dieser Zähler kleiner ist setzt man die Liste mit den Spalten zurück und fügt die aktuelle Spalte hinzu. Die Anzahl der notwendigen Änderungen werden ebenfalls auf die momentane Zahl des Zählers gesetzt. Im Fall das wir dieselben Anzahl an Änderungen haben, wird die aktuelle Spalte zur Liste hinzugefügt. Im dritten Fall, wo der Zähler ist größer ist, macht man nichts mehr. Nun geht man in die nächste Spalte und wiederholt das auszählen, Variablen anpassen, falls nötig und so weiter.

Zuletzt gibt man die Anzahl der Änderungen und Spalten noch aus in die Konsole, in ein Dokument, etc.

Beispiele

Notiz: Der Spaltenoutput ist an die mathematische Zählweise angepasst. Somit ist die erste Spalte „1“, die zweite „2“ usw. Die Spalte „0“ existiert nicht in der Programmausgabe. Spalte „0“ ist sozusagen die Spalte, in der die Namen stehen würden:

x

praeferenz0.txt

```
Spalte(n): 6
Anzahl der Änderungen: 2
```

Die Präferenztabelle der Datei ist dieselbe, wie die im Beispiel bei der Aufgabenstellung. Die Spalte 6 bzw. Termin 6 zeigt, dass dieser Wert korrekt ist. Auch unter den Änderungen werden 2 angeziegt. Dies müsste ebenfalls korrekt sein, da das Niklaus und Rószas Änderungen sind. Da es nicht Bestandteil der Aufgabe ist, wird hier nicht gezeigt, was geändert werden muss.

praeferenzen1.txt

```
Spalte(n): 2, 3
Anzahl der Änderungen: 1
```

Obige Ausgabe bedeutet, dass in den Spalten 2 und 3 bzw. an den Terminen 2 und 3 eine Änderung vorzunehemen ist, um den Adas perfekten Termin zu erreichen.

praeferenzen2.txt

```
Spalte(n): 4
Anzahl der Änderungen: 0
```

Hier wird gesagt, dass Spalte/Termin 4 die Bedingnungen für Adas perfekten Termin erfüllt. Wenn man sich nun die Spalte anschaut, wird man feststellen, dass man bei keiner Person eine angenehmere Terminart findet, als bereits in der Spalte ist. Man kann die gleichwertige Terminart finden, aber dies hat keine Beeinflussung auf das Ergebnis.

praeferenzen3.txt

```
Spalte(n): 18
Anzahl der Änderungen: 7
```

praeferenzen4.txt

```
Spalte(n): 22
Anzahl der Änderungen: 14
```

praeferenzen5.txt

```
Spalte(n): 31, 56
Anzahl der Änderungen: 34
```

Hier werden zwei Spalten/Termin, nämlich „31“ und „56“ angezeigt, Somit gilt die Änderungszahl 34 für beide Spalten/Termine. Dennoch muss diese Zahl nicht dieselben Personen in beiden Spalten meinen, auch wenn sie das durchaus könnte. (siehe „Eigene Beispiele > praeferenzen6.txt“ und „Eigene Beispiele > praeferenzen7.txt“ für genauere Erklärung)

Eigene Beispiele

praeferenzen6.txt

Dateiinhalt:

```
3 3
2 1 2
1 2 0
0 0 0
```

Ouput:

```
Spalte(n): 2, 3
Anzahl der Änderungen: 1
```

Zum einem zeigt dieses Beispiel, dass Zeilen, welche nur eine Terminart haben, komplett ignoriert werden können, da dort nie eine Änderung vorfallen wird. Zum anderem wird gesagt, dass in den Spalten 2 und 3 jweils eine Änderung notwendig ist. In Spalte 2 wäre das in Zeile 2 („2“ zu „0“) und in Spalte 3 in Zeile 1 („2“ zu „1“). Diese 1 Änderung findet in zwei verschiedenen Zeilen statt. Bei jeden Termin (Spalte) ändert man also die Terminart für eine andere Person (Zeile). Dies wird allerdings nicht vermerkt.

praeferenzen7.txt

Dateiinhalt

```
3 3
0 2 2
1 0 0
1 0 0
```

Output:

Spalte(n): 2, 3
Anzahl der Änderungen: 1

In diesem Beispiel bekommt man exakt dasselbe Ergebnis, wie bei „praeferenzen6.txt“ nur mit dem Unterschied, dass zweimal die Terminart derselben Person geändert werden muss bei den verschiedenen Spalten/Terminen. Dadurch ist die Ausgabe eigentlich in der Hinsicht, welche Personen müssten was anderes eingeben recht schwach aussagend. Dies könnte man anpassen,

indem man sich merkt, welche Personen eine angenehmere Terminart in welcher Spalte haben. Dies wurde, aber nicht hinzugefügt, da Ada mit den gegebenen Informationen zufrieden ist.

Quellcode

Der Quellcode beinhaltet die Dateilesefunktion nicht. Das Programm verwendet die Module „os“, „typing“

Bestimmen der angenehmsten Terminarten für die Personen:

```
def get_best_type_of_date(table: List[List[int]]) -> Dict[int, int]:
    """Bestimmt die angenehmste Art von Termin jeder Person und gibt
       das dabei entsehende Dictionary zurück.

    Args:
        table (List[List[int]]): die Präferenztabelle

    Returns:
        Dict[int, int]: das Dictionary, welches die angenehmste Art von Termin jeder Person beinhaltet
    """
    return {i: min(person_dates) for i, person_dates in enumerate(table)}
```

Funktion zum Lösen des Programms:

```python
def solve(table: List[List[int]]) -> Tuple[List[int], int]:
    """Berechnet den/die besten Termin(e) und dessen/deren notwenidgen Änderungen

    Args:
        table (List[List[int]]): die Pröferenztabelle

    Returns:
        Tuple[List[int], int]: An erster Stelle alle Spalten
    """
    persons_best_type_of_date = get_best_type_of_date(table)

    columns_in_row = len(table[0])  # Annahme: Jede Zeile hat dieselbe Anzahl von Werten,
                                    # was der Fall in einer Tabelle ist
    res_columns = []
    res_changes: int = len(table)  # mindestens die Änderungen in einer anderen Spalte
                                   # sind größer

    for column in range(columns_in_row):
        # berechnen der Änderungen
        changes = 0
        for row in range(len(table)):
            if table[row][column] > persons_best_type_of_date[row]:
                changes += 1

        # speichervariable nach den 3 Optionen anpassen
        if changes < res_changes:
            res_changes = changes
            res_columns = [column]  # aktuelle Spalte ist die erste mit den neuen wenigsten Änderungen

        elif changes == res_changes:
            res_columns.append(column)

    return res_columns, res_changes
```

Mainfunction (Bindeglied der Abläufe für das Programm und regelt die Ausgabe; ebenfalls der Startpunkt des Porgramms)

```python
def main():
    table = read_file(input("Path (relative to %s): " % os.getcwd()))
    columns, changes = solve(table)

    # Ausgabe: in mathematischer Zählweise (Start bei 1)
    print("Spalte(n):", ", ".join(str(e + 1) for e in columns))
    print("Anzahl der Änderungen:", changes)
```