Aufgabe 2: Vollgeladen

Team-ID: 00861

Team: Team YEET

Bearbeiter/-innen dieser Aufgabe:
Robin Schupp & Leo Kling

18. November 2021

Inhaltsverzeichnis

Lösungsidee

Um vier Hotels mit den besten Bewertungen zu finden, haben wir uns dazu entschieden, einen Brute-Force Ansatz zu verfolgen.

Wir vergleichen also jedes Hotel mit jedem anderen Hotel und überprüfen anhand von uns ausgewählten Kriterien, ob ein Hotel infrage kommt. Sollte ein Hotel eine geringere Bewertung als die bis jetzt geringste Bewertung aufweisen oder es würde für einen zu großen Umweg sorgen oder der Weg zum Hotel überschreitet die maximal zulässige Tagesdistanz, kommt das Hotel für uns nicht infrage.

Durch dieses Ausschlussverfahren erhalten wir die best mögliche Auswahl an Hotels.

Umsetzung

Um das Problem per Brute-Force zu lösen, bedarf es vier ineinander verschachtelter Zählschleifen. Dadurch ich zwar die Zeitkomplexität relativ groß, was sich besonders bei Eingabe von vielen Hotels bemerkbar macht.

Die Implementierung ist vergleichbar zu einem zu lösenden Zahlenschloss mit vier benötigten Ziffern. Wir finden zuerst eine passende erste Zahl, dann eine zweite Zahl und so weiter. Nur das wir anstatt Zahlen versuchen Hotels zu finden.

Die erste Zählschleife übernimmt dabei das erste benötigte Hotel. Für jedes mögliche Hotel entlang der Strecke überprüfen wir dann, ob es eine bessere Bewertung als das bis jetzt am schlechtesten bewertete Hotel hat, ob es weniger als 360 Minuten von der momentanen Position entfernt ist und ob es weiter als eine gewisse Distanz entfernt liegt, damit wir nicht zu viele Hotels auswählen und die Strecke zum Ziel überhaupt in 5 Tagen zu schaffen ist.
Diese Distanz errechnet sich aus der maximalen Fahrzeit pro Tag und einem Toleranzwert, der aus der Differenz der maximal erlaubten Fahrdauer (1800 Fahrminuten) und der Gesamtfahrzeit gebildet wird.

Sobald das erste Hotel in der gegebenen Liste an Hotels diese Voraussetzungen erfüllt, gehen wir in die nächste Zählschleife und wiederholen den Selektionsprozess für die verbleibenden Hotels.

Sollten wir vier Hotels gefunden haben (wir sind nun in der letzten Zählschleife), haben wir aber möglicherweise noch nicht alle Hotels mit jedem andern Hotel verglichen. Es kann also durchaus noch eine bessere Kombination an Hotels mit einer besseren geringesten Bewertung geben. Deswegen speichern wir die momentan geringste Bewertung der ausgewählten Hotels, damit die restlichen Hotels mit diesem Wert verglichen werden können.

Nachdem wir alle Hotels miteinander verglichen haben, geben wir die gefundene Kombination zurück.

Beispiele

Die Ausgabe des Programms ist wie folgt aufgebaut:

```
Niedrigste Bewertung: x
Hotels:
(Fahrminuten vom Start, Bewertung)
(... Hotel 2)
(... Hotel 3)
(... Hotel 4)
```

Beispiel 1

```
Niedrigste Bewertung: 2.7
Hotels:
(347, 2.7)
(687, 4.4)
(1007, 2.8)
(1360, 2.8)
```

Ausgabe des Programms für Beispiel 1

Beispiel 2

```
Niedrigste Bewertung: 2.3
Hotels:
(341, 2.3)
(700, 3.0)
(1053, 4.8)
(1380, 5.0)
```

Ausgabe des Programms für Beispiel 2

Beispiel 3

```
Niedrigste Bewertung: 0.3
Hotels:
(360, 1.0)
(717, 0.3)
(1076, 3.8)
(1433, 1.7)
```

Ausgabe des Programms für Beispiel 3

Beispiel 4

```
Niedrigste Bewertung: 4.6
Hotels:
(340, 4.6)
(676, 4.6)
(1032, 4.9)
(1316, 4.9)
```

Ausgabe des Programms für Beispiel 4

Beispiel 5

```
Niedrigste Bewertung: 5.0
Hotels:
(317, 5.0)
(636, 5.0)
(987, 5.0)
(1286, 5.0)
```

Ausgabe des Programms für Beispiel 5

Quellcode

Brute-Force Methode

```
tolerance = 1800 - distance


def brute_force():
    highest = 0.0
    hotels = []

    for a in range(0, len(hotel_tupil)):
        a_dis, a_rat = hotel_tupil[a]
        if a_rat < highest or a_dis < 360 - tolerance or a_dis > 360:
            continue
        for b in range(a + 1, len(hotel_tupil)):
            b_dis, b_rat = hotel_tupil[b]
            if b_rat < highest or b_dis - a_dis < 360 - tolerance or b_dis - a_dis > 360:  #
Maximal 6h Zeit pro Tag
                continue
            for c in range(b + 1, len(hotel_tupil)):
                c_dis, c_rat = hotel_tupil[c]
                if c_rat < highest or c_dis - b_dis < 360 - tolerance or c_dis - b_dis > 360:
 # Maximal 6h Zeit pro Tag
                    continue
                for d in range(c + 1, len(hotel_tupil)):
                    d_dis, d_rat = hotel_tupil[d]
                    if d_rat < highest or d_dis - c_dis < 360 - tolerance or distance - d_dis
> 360 or d_dis - c_dis > 360:
                        continue

                    min_rat = min([a_rat, b_rat, c_rat, d_rat])
                    if min_rat >= highest:
                        highest = min_rat
                        hotels = [a, b, c, d]

    print("Niedrigste Bewertung: " + str(highest))
    print("Hotels:")
    for h in hotels:
        print(hotel_tupil[h])
```

Diese Methode beinhaltet die Logik des Programms. Zu sehen sind vier ineinander vernestete Zählschleifen, die jedes Hotel mit jedem anderen Hotel vergleicht und anhand den erläuterten Kriterien ausschließt oder eben akzeptiert.