Aufgabe 1: Schiebeparkplatz

Team-ID: 00861

Team: Team YEET

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

18. November 2021

Inhaltsverzeichnis

Lösungsidee

Um für einen normal geparkten Wagen zu bestimmen, welche quer geparkten Wagen wie verschoben werden müssen, müssen wir zunächst prüfen, ob vor dem Parkplatz überhaupt ein Wagen quer steht.

Falls nicht, müssen wir nichts verschieben. Falls schon, müssen wir den quer stehenden Wagen verschieben.

Um einen Wagen verschieben zu können, müssen auch dessen beide Zielfelder frei sein, falls dort ein Wagen steht, muss dieser auch verschoben werden. Diesen Vorgang setzen wir fort, bis entweder der als letztes zu verschiebende Wagen nicht blockiert ist oder das Ende des Parkplatzes erreicht ist.

Umsetzung

Um die nötigen Verschiebungen für einen Wagen herauszufinden, prüfen wir das Feld vor dem Wagen. Falls dies nicht leer ist, probieren wir, den blockierenden Wagen in beide Richtungen zu verschieben. Wann immer wir einen Wagen verschieben, prüfen wir den Parkplatz zwei Parkplätze weiter von dem, was wir freilegen wollen, da ein die Verschiebung blockierender Wagen auf jeden Fall diesen Parkplatz belegt.

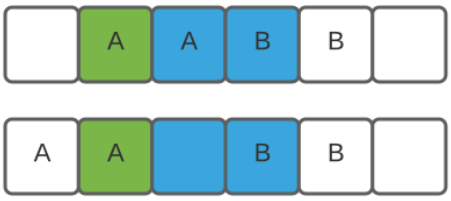

Fig. 1: Blockierung eines zu verschiebenden Wagens
(Grün markiert: Der ursprünglich freizulegende Parkplatz, Blau markiert: Der neue Platz des verschobenen Wagens, A: Der zu verschiebende Wagen, B: Der blockierende Wagen)

Falls der ursprünglich zu verschiebende Wagen blockiert ist, prüfen wir rekursiv, wie wir den blockierenden Wagen verschieben können (dieser kann natürlich nur in dieselbe Richtung verschoben werden wie der Wagen vorher). Dies machen wir solange, bis ein zu verschiebender Wagen nicht mehr blockiert ist oder schon am Rand des Parkplatzes steht. Beim Zusammenstellen der Liste on Verschiebungen muss dann der letzte Wagen als erstes hinzugefügt werden, da die anderen noch blockiert sind.

Schlussendlich haben wir zwei Wege, Wagen zu verschieben, um den normal geparkten Wagen ausfahren zu lassen. Wir wählen einfach den aus, der weniger Verschiebungen beinhaltet und haben somit unser Ergebnis.

Beispiele

Beispiel 0

In diesem Beispiel sind die Wagen wie folgend angeordnet:

Fig. 2: Beispiel 0

Die Ausgabe des Programms ist passend dazu wie folgt:

```
C: H 1 rechts
D: H 1 links
F: H 1 links, I 2 links
G: I 1 links
```

Beispiel 1

In diesem Beispiel sind die Wagen wie folgend angeordnet:

Fig. 3: Beispiel 1

Die Ausgabe des Programms ist passend dazu wie folgt:

```
B: P 1 rechts, O 1 rechts
C: Q 1 rechts, P 2 rechts, O 2 rechts
D: P 1 rechts
E: Q 1 rechts, P 2 rechts
G: Q 1 rechts
H: Q 2 rechts
K: R 1 rechts
L: R 1 links
```

Beispiel 2

In diesem Beispiel sind die Wagen wie folgend angeordnet:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
		O	O		P	P	Q	Q	R	R		S	S

Fig. 4: Beispiel 2

Die Ausgabe des Programms ist passend dazu wie folgt:

```
C: O 1 rechts
D: O 1 links
F: O 1 links, P 2 links
G: P 1 links
H: R 1 rechts, Q 1 rechts
I: P 1 links, Q 1 links
J: R 1 rechts
K: P 1 links, Q 1 links, R 1 links
M: P 1 links, Q 1 links, R 1 links, S 2 links
N: S 1 links
```

Beispiel 3

In diesem Beispiel sind die Wagen wie folgend angeordnet:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
	O	O		P	P			Q	Q	R	R	S	S

Fig. 5: Beispiel 3

Die Ausgabe des Programms ist passend dazu wie folgt:

```
B: O 1 rechts
C: O 1 links
E: P 1 rechts
F: P 2 rechts
I: Q 2 links
J: Q 1 links
K: Q 2 links, R 2 links
L: Q 1 links, R 1 links
M: Q 2 links, R 2 links, S 2 links
N: Q 1 links, R 1 links, S 1 links
```

Beispiel 4

In diesem Beispiel sind die Wagen wie folgend angeordnet:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Q	Q	R	R			S	S			T	T		U	U	

Fig. 6: Beispiel 4

Die Ausgabe des Programms ist passend dazu wie folgt:

```
A: R 1 rechts, Q 1 rechts
B: R 2 rechts, Q 2 rechts
C: R 1 rechts
D: R 2 rechts
G: S 1 rechts
H: S 2 rechts
K: T 1 rechts
L: T 1 links
N: T 1 links, U 2 links
O: U 1 links
```

Beispiel 5

In diesem Beispiel sind die Wagen wie folgend angeordnet:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
		P	P	Q	Q			R	R			S	S	

Fig. 7: Beispiel 5

Die Ausgabe des Programms ist passend dazu wie folgt:

```
C: Q 1 rechts, P 1 rechts
D: P 1 links
E: Q 1 rechts
F: Q 2 rechts
I: R 1 rechts
J: R 2 rechts
M: S 2 links
N: S 1 links
```

Alle Beispiele geben die richtige Ausgabe. (Wagen, die keine Verschiebung benötigen sind hier ausgelassen)

Quellcode

Der Quellcode für das Überprüfen von Verschiebungen nach rechts sieht aus wie folgt:

```
# Anzahl an Verschiebungen nach rechts
def free_right(place, ret):
    if hori[place] == 0:  # Falls der Platz leer ist, muss nicht mehr geschoben werden
        return ret
    elif place >= len(hori) or place >= len(hori) - 4 + hori[place]:  # Auto kann nicht aus dem
Parkplatz herausgeschoben werden
        return None
    elif hori[place] == 1:  # Das Auto muss 1 Parkplatz nach rechts verschoben werden
        x = [(hori_names[place], 1)]
        for a in ret:
            x.append(a)
        return free_right(place + 2, x)  # Überprüfen des Parkplatzes, auf den das Auto
geschoben wird
    elif hori[place] == 2:  # Das Auto muss 2 Parkplätze nach rechts verschoben werden
        x = [(hori_names[place], 2)]
        for a in ret:
            x.append(a)
        return free_right(place + 2, x)  # Überprüfen des Parkplatzes, auf den das Auto
geschoben wird
```

Die Methode nimmt als Parameter den Parkplatz, der freigelegt werden soll, als auch die Liste von bisherigen Verschiebungen. Falls der Platz schon leer ist, wird diese Liste einfach zurückgegeben, falls ein Wagen am rechten Rand des Parkplatzes steht, None.

Danach wird einfach je nachdem, ob die linke oder die rechte Hälfte eines Wagens auf dem Platz steht, eine Verschiebung dieses Wagens um ein oder zwei Plätze nach rechts vorne an die Liste angefügt (vorne, da der aktuelle Wagen vor dem letzten - blockierten - verschoben werden muss). Dann wird der Platz zwei Plätze weiter rechts rekursiv überprüft.

Genauso gibt es eine Methode free_left, die die Verschiebung nach links überprüft.

Hier der Quellcode für das Überprüfen von allen normal geparkten Wagen:

```
def res():
   for auto in vert:
       left = free_left(auto, [])
       right = free_right(auto, [])

       if left is None:  # Es kann nur nach rechts geschoben werden
           actions = right
           direction = "rechts"
       elif right is None:  # Es kann nur nach links geschoben werden
           actions = left
           direction = "links"
       elif len(right) <= len(left):  # Nach rechts muss weniger geschoben werden
           actions = right
           direction = "rechts"
       else:  # Nach links muss weniger geschoben werden
           actions = left
           direction = "links"

       if actions is None:  # Es kann nicht geschoben werden
           print(chr(auto + vert_first) + ": unmöglich")
           continue
       else:
           print_actions(auto, actions, direction)
```

Hier werden einfach alle normal geparkten Wagen nacheinander durchgegangen und für jeden sowohl die Verschiebungen nach links als auch die nach rechts geprüft.

Dann wird die kürzere der beiden zurückgegebenen Listen ausgegeben.