Aufgabe 1: Schiebeparkplatz

Team-ID: 00067

Team-Name: Panic! at the Kernel

Bearbeiter/-innen dieser Aufgabe:
Christopher Besch

21. November 2021

Inhaltsverzeichnis

1 Abstract **1**

2 Lösungsidee **1**
2.1 Bestimmung der Kategorien und Verschiebungsbestimmungen 2

3 Umsetzung und Quellcode **3**

4 Beispiele **5**

1 Abstract

Zur Lösung der Aufgabe wird für jedes Feld bestimmt, wie weit ein quer stehendes Auto auf diesem verschoben werden kann. Ausserdem wird berechnet, welchen anderen Autos wie verschoben werden müssen, damit ein Feld so weit wie möglich verschoben werden kann. Anschliessend wird die Lösungsidee in C++ implementiert und Spezialfälle mit Beispieleingaben erläutert.

2 Lösungsidee

Im Folgenden werden querstehende Autos mit „Q-Auto“ abgekürzt; alle nicht querstehenden Autos werden „Auto“ genannt. Ein Feld—eine mögliche Position für ein (halbes) Q-Auto—selbst kann nicht verschoben werden, allerdings wird im Folgenden diese Terminologie verwendet. Hierbei wird festgelegt, dass ein Feld soweit verschoben werden kann, wie das an dieser Stelle stehendes Q-Auto verschoben werden kann. Wenn kein Q-Auto auf einem Feld steht, kann es beliebig verschoben werden.

Es lässt sich feststellen, dass jedes Feld in einer von drei Kategorien eingeordnet werden kann:

1. Das Feld kann zweifach nach links verschoben werden. Eine Verschiebung um mehr als zwei Positionen ist nie notwendig da die Länge eines Q-Autos zwei entspricht und immer nur ein einziges Feld freigemacht werden muss. Daher werden grössere Verschiebungen ignoriert.
2. Das Feld kann genau einfach nach links verschoben werden.
3. Das Feld kann nicht nach links verschoben werden.

Diese Kategorien gelten analog für Rechtsverschiebungen.

Nun kann für jedes Feld A bestimmt werden, welche anderen Felder wieweit verschoben werden müssen, damit A seiner Kategorie entsprechend maximal verschoben werden kann. Hierbei werden alle Felder, die nicht mit einem Q-Auto besetzt sind, per Definition als nicht zu verschieben angesehen. Diese Information wird linke beziehungsweise rechte volle Verschiebungsbestimmung des Feldes A genannt. Eine Verschiebungsbestimmung Λ für das Feld A bei insgesamt n Feldern entspricht der Liste mit den Indizes $0, 1, \ldots, n-1$. Das Element e am Index i entspricht:

1 wenn das Feld mit dem Index i einfach verschoben werden muss,

2 wenn es zweifach verschoben werden muss und

0 wenn es nicht verschoben werden muss.

Wenn ein Feld zweifach verschoben werden kann, das Feld allerdings nur einfach verschoben werden muss, kann die reduzierte Verschiebungsbestimmung verwendet werden. Die reduzierte Verschiebungsbestimmung entspricht der vollen Verschiebungsbestimmung Λ, wobei jeder Wert e von Λ um 1 reduziert wird, wenn $e > 0$. Dies lässt sich dadurch begründen, dass wenn ein Q-Auto einmal weniger verschoben werden muss alle anderen notwendigen Verschiebungen ebenfalls einmal weniger durchgeführt werden müssen. Die für eine bestimmte Verschiebung minimal notwendige Verschiebungsbestimmung (entweder voll oder reduziert) wird notwendige Verschiebungsbestimmung genannt.

Nun muss für jedes Feld sowohl die linke und rechte Kategorie als auch die Verschiebungsbestimmungen vorliegen. Damit lässt sich recht einfach bestimmen:

- Ob ein beliebiges Auto ausfahren kann: Dies ist der Fall wenn die Kategorie des betroffenen Feldes mindestens 1 entspricht.
- Ob die Verschiebung nach links oder recht optimal ist: Wenn die Verschiebung nach links weniger Q-Autos bewegt als die nach rechts ist sie optimal. Die Menge an zu verschiebenden Autos entspricht der Hälfte der Menge an Elemente e in der notwendigen Verschiebungsbestimmung mit $e > 0$. Dies ist der Fall, da alle Q-Autos mit zwei Feldern Länge doppelt gezählt werden.
- Welche Q-Autos wie verschoben werden müssen, damit das betreffende Auto ausfahren kann. Hierzu kann wie bereits beschrieben die notwendige Verschiebungsbestimmung angewendet werden. Es ist nur zu beachten, dass sowohl das linke als auch rechte Feld des Q-Autos in der Verschiebungsbestimmung vorkommen. Diese Doppelung muss ausgefiltert werden.

2.1 Bestimmung der Kategorien und Verschiebungsbestimmungen

Das Verfahren wird für die Linksverschiebungen beschrieben, kann allerdings ebenfalls für Rechtsverschiebungen angewendet werden.

Wenn an einem Feld kein Q-Auto vorhanden ist, kann es zweifach verschoben werden. Andernfalls gilt:

- Damit ein Feld einfach verschoben werden kann muss das direkt links am Q-Auto angrenzende Feld mindestens einfach verschiebbar sein.
- Damit ein Feld zweifach verschoben werden kann muss das links am Q-Auto angrenzende Feld zweifach verschiebbar sein. Zudem muss das übernächste linke Feld mindestens einfach verschiebbar sein.

Diese Information kann rekursiv berechnet werden, wobei der Rekursionsanker dem linken Rand entspricht. Der Rand kann nicht verschoben werden.

Für die Verschiebungsbestimmungen kann ein ähnliches Verfahren angewendet werden. Sei α die Summe zweier Verschiebungsbestimmungen Λ und Γ. Dann entspricht jedes Element aus α mit dem Index i dem Maximum der Elemente aus Λ und Γ mit dem Index i.

- Wenn ein Feld nur einfach verschoben werden kann entspricht dessen Verschiebungsbestimmung der notwendigen Verschiebungsbestimmung für die einfache Verschiebung des links angrenzenden Feldes.
- Kann es zweifach verschoben werden entspricht sie der Summe aus
  a) der notwendigen Verschiebungsbestimmung für die zweifache Verschiebung des direkt angrenzenden Feldes und
  b) der notwendigen Verschiebungsbestimmung für die einfache Verschiebung des übernächsten Feldes.

Diese Information kann ebenfalls rekursiv bestimmt werden. Die Rekursionsanker sind die freien Felder, die verschoben werden können, ohne ein anderes Feld zu berühren.

Nun wird das Prinzip der dynamischen Programmierung verwendet. Es lässt sich erkennen, dass die Information eines Feldes immer nur von den Feldern zur Linken abhängt. Daher können alle Felder von

links nach rechts durchgegangen werden und der Kategorie und Verschiebungsbestimmung mithilfe der bereits berechneten Werte bestimmt werden. Wenn das Verfahren auf Rechtsverschiebungen angewendet wird, müssen die Felder analog von rechts nach links durchgegangen werden.

Um das Verfahren zu optimieren müssen nur alle Felder mit Q-Autos betrachtet werden. Alle leeren Felder können immer beliebig verschoben werden und kein Q-Auto muss verschoben werden, um dies zu erreichen.

3 Umsetzung und Quellcode

Die Lösungsidee wird in C++ implementiert. Die Kategorie wird als vector repräsentiert:

```
// 2 -> can be moved two paces
// 1 -> can only be moved once
// 0 -> unable to be moved at all
// initially all cars can be moved as far as possible
std::vector<int> move_left_possible(cars_amount, 2);
std::vector<int> move_right_possible(cars_amount, 2);
```

Die Verschiebungsbestimmungen als zweidimensionaler vector:

```
// 0 -> doesn't have to be moved
// 1 -> has to be moved once
// 2 -> has to be moved twice
// initially no cars have to be moved
std::vector<std::vector<int>> left_shift(cars_amount, std::vector<int>(cars_amount, 0));
std::vector<std::vector<int>> right_shift(cars_amount, std::vector<int>(cars_amount, 0));
```

Hierbei entspricht `left_shift[a]` der vollen Verschiebungsbestimmung des Feldes mit dem Index a. Alle Felder werden von links bei 0 anfangend indiziert. Damit kann mit `left_shift[a][b]` die dafür benötigte Verschiebung des Feldes mit dem Index b erhalten werden.

Die Funktion `update_row_dir` bildet die Summe der beiden Verschiebungsbestimmungen `source` und `target`. Das Ergebnis überschreibt `target`.

```
// merge source into target
void update_row_dir(const std::vector<int>& source, std::vector<int>& target)
{
    for(int i = 0; i < target.size(); ++i)
        target[i] = std::max(target[i], source[i]);
}
```

Die Funktion `update_row_red` führt dasselbe durch nur reduziert es zuerst `source`.

```
// merge reduced source into target
void update_row_red(const std::vector<int>& source, std::vector<int>& target)
{
    for(int i = 0; i < target.size(); ++i)
        target[i] = std::max(target[i], source[i] - 1);
}
```

Um die Kategorien und Verschiebungsbestimmungen zu berechnen wird die Funktion `load_possible_moves` verwendet. Diese kann sowohl Links- als auch Rechtsverschiebungen berechnen, was durch den Parameter `go_left` eingestellt wird. Dementsprechend werden die Variablen `start` und `end` gesetzt. Die darauffolgende Hauptschleife erhöht (wenn von links nach rechts) oder erniedrigt (wenn von rechts nach links) den Zähler `c`. In der Schleife werden dadurch Q-Autos in der korrekten Reihenfolge durchgegangen.

Zuerst wird das Nachbarfeld (das direkt links beziehungsweise direkt rechts angrenzende Feld) und das übernächste Nachbarfeld definiert. Darauffolgend wird überprüft, wie weit diese Felder verschoben werden können. Mit dieser Information wird die Kategorien und Verschiebungsbestimmung entsprechend der Lösungsidee aktualisiert.

```
void load_possible_moves(
    const std::vector<std::pair<int, char>>& cross_cars,
    int                                      cars_amount,
```

```
    std::vector<std::vector<int>>&          shift,
    std::vector<int>&                       move_possible,
    bool                                    go_left)
{
    int start = go_left ? 0 : cross_cars.size() - 1;
    int end   = go_left ? cross_cars.size() : -1;
    for(int c = start; c != end; go_left ? ++c : --c) {
        int first, second, next, next_but_one;
        if(go_left) {
            // location of cross_car
            // this pace gets primarily moved
            first = cross_cars[c].first;
            // secondary move
            second = cross_cars[c].first + 1;
            // places left or right of cross_car
            next         = first - 1;
            next_but_one = first - 2;
        }
        else {
            first        = cross_cars[c].first + 1;
            second       = cross_cars[c].first;
            next         = first + 1;
            next_but_one = first + 2;
        }
        // check if can be moved twice
        if(next_but_one >= 0 &&
           next_but_one < cars_amount &&
           move_possible[next] == 2 &&
           move_possible[next_but_one] >= 1) {
            // can be moved twice
            move_possible[first]  = 2;
            move_possible[second] = 2;
            // this car obviously has to be moved
            shift[first][first]   = 2;
            shift[first][second]  = 2;
            shift[second][first]  = 2;
            shift[second][second] = 2;

            // this one always has to be moved two paces
            update_row_dir(shift[next], shift[first]);
            update_row_dir(shift[next], shift[second]);
            // only reduce when one too much
            if(move_possible[next_but_one] == 2) {
                // right one only has to move once
                // only half movement required
                update_row_red(shift[next_but_one], shift[first]);
                update_row_red(shift[next_but_one], shift[second]);
            }
            else {
                // full movement required
                update_row_dir(shift[next_but_one], shift[first]);
                update_row_dir(shift[next_but_one], shift[second]);
            }
        }
        // can be moved once at least?
        else if(next >= 0 &&
                next < cars_amount &&
                move_possible[next] >= 1) {
            // can be moved once
            move_possible[first]  = 1;
            move_possible[second] = 1;

            // obviously has to be moved once
            shift[first][first]   = 1;
            shift[first][second]  = 1;
            shift[second][first]  = 1;
            shift[second][second] = 1;

            if(move_possible[next] == 2) {
                update_row_red(shift[next], shift[first]);
                update_row_red(shift[next], shift[second]);
            }
            else {
```

```
                update_row_dir(shift[next], shift[first]);
78              update_row_dir(shift[next], shift[second]);
            }
80      }
        // can't be moved
82      else {
            move_possible[first]  = 0;
84          move_possible[second] = 0;
        }
86  }
}
```

Diese Funktion ist die Aufwendigste. Da sie jedes Q-Auto einmalig durchläuft, ist die Laufzeit dieser Implementation linear.

Nachdem diese Funktion für Links- und Rechtsverschiebungen ausgeführt wurde, wird für jedes Feld bestimmt, welche Verschiebung optimal ist und entsprechen ausgegeben. Hierfür wird die Funktion `better_shift` verwendet:

```
// which shift moves fewer cross cars?
2 bool better_shift(const std::vector<int>& a, const std::vector<int>& b)
{
4     int count_a {0}, count_b {0};
      int moves_a {0}, moves_b {0};
6     for(int i = 0; i < a.size(); ++i) {
          if(a[i]) {
8             ++count_a;
              moves_a += a[i];
10        }
          if(b[i]) {
12            ++count_b;
              moves_b += b[i];
14        }
      }
16    if(count_a < count_b)
          return true;
18    if(count_a > count_b)
          return false;
20    if(moves_a < moves_b)
          return true;
22    if(moves_a > moves_b)
          return false;
24    return true;
}
```

An dieser Stelle wird die Aufgabe erweitert: Laut der Aufgabenstellung sind zwei Lösungen dann gleichwertig, wenn sie gleich viele Q-Autos verschieben. Die Implementation entscheidet in diesen Fällen allerdings immer so, dass die Q-Autos so wenig wie möglich verschoben werden.

4 Beispiele

Neben der nötigen Verschiebungen, um ein Auto an einem bestimmten Feld ausfahren zu lassen, werden die Kategorien (possible moves) und vollen Verschiebungsbestimmungen (shifts) ausgegeben.

Parkplatz 0

```
1 cat examples/parkplatz0.txt | ./a.out
  ## LEFT
3 name possible moves shifts
  ==== ============== ======
5                     A B C D E F G

7 A    2              0 0 0 0 0 0 0
  B    2              0 0 0 0 0 0 0
9 C    2              0 0 2 2 0 0 0
  D    2              0 0 2 2 0 0 0
11 E   2              0 0 0 0 0 0 0
```

```
F    2              0 0 1 1 0 2 2
G    2              0 0 1 1 0 2 2


## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G

A    2              0 0 0 0 0 0 0
B    2              0 0 0 0 0 0 0
C    1              0 0 1 1 0 0 0
D    1              0 0 1 1 0 0 0
E    2              0 0 0 0 0 0 0
F    0              0 0 0 0 0 0 0
G    0              0 0 0 0 0 0 0


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

Parkplatz 1

```
cat examples/parkplatz1.txt | ./a.out
## LEFT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    1              0 1 1 0 0 0 0 0 0 0 0 0 0 0
C    1              0 1 1 0 0 0 0 0 0 0 0 0 0 0
D    1              0 1 1 1 1 0 0 0 0 0 0 0 0 0
E    1              0 1 1 1 1 0 0 0 0 0 0 0 0 0
F    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
G    2              0 1 1 1 1 0 2 2 0 0 0 0 0 0
H    2              0 1 1 1 1 0 2 2 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0
L    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0
M    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0


## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 2 2 2 2 0 1 1 0 0 0 0 0 0
C    2              0 2 2 2 2 0 1 1 0 0 0 0 0 0
D    2              0 0 0 2 2 0 1 1 0 0 0 0 0 0
E    2              0 0 0 2 2 0 1 1 0 0 0 0 0 0
F    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0
H    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0
L    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0
M    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0

```

```
A:
B: P 1 rechts, O 1 rechts
C: O 1 links
D: P 1 rechts
E: O 1 links, P 1 links
F:
G: Q 1 rechts
H: Q 1 links
I:
J:
K: R 1 rechts
L: R 1 links
M:
N:
```

Parkplatz 2

```
cat examples/parkplatz2.txt | ./a.out
## LEFT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
C    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0
D    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0
E    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
F    2              0 0 1 1 0 2 2 0 0 0 0 0 0 0
G    2              0 0 1 1 0 2 2 0 0 0 0 0 0 0
H    2              0 0 1 1 0 2 2 2 2 0 0 0 0 0
I    2              0 0 1 1 0 2 2 2 2 0 0 0 0 0
J    2              0 0 1 1 0 2 2 2 2 2 2 0 0 0
K    2              0 0 1 1 0 2 2 2 2 2 2 0 0 0
L    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
M    2              0 0 0 0 0 1 1 1 1 1 1 0 2 2
N    2              0 0 0 0 0 1 1 1 1 1 1 0 2 2

## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
C    2              0 0 2 2 0 1 1 1 1 1 1 0 0 0
D    2              0 0 2 2 0 1 1 1 1 1 1 0 0 0
E    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
F    1              0 0 0 0 0 1 1 1 1 1 1 0 0 0
G    1              0 0 0 0 0 1 1 1 1 1 1 0 0 0
H    1              0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
I    1              0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
J    1              0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
K    1              0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
L    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


A:
B:
C: O 1 rechts
D: O 1 links
E:
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
L:
M: P 1 links, Q 1 links, R 1 links, S 2 links
N: S 1 links
```

Parkplatz 3

```
cat examples/parkplatz3.txt | ./a.out
## LEFT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    1              0 1 1 0 0 0 0 0 0 0 0 0 0 0
C    1              0 1 1 0 0 0 0 0 0 0 0 0 0 0
D    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
E    2              0 1 1 0 2 2 0 0 0 0 0 0 0 0
F    2              0 1 1 0 2 2 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 2 2 0 0 0 0
J    2              0 0 0 0 0 0 0 0 2 2 0 0 0 0
K    2              0 0 0 0 0 0 0 0 2 2 2 2 0 0
L    2              0 0 0 0 0 0 0 0 2 2 2 2 0 0
M    2              0 0 0 0 0 0 0 0 2 2 2 2 2 2
N    2              0 0 0 0 0 0 0 0 2 2 2 2 2 2

## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 2 2 0 1 1 0 0 0 0 0 0 0 0
C    2              0 2 2 0 1 1 0 0 0 0 0 0 0 0
D    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
E    2              0 0 0 0 2 2 0 0 0 0 0 0 0 0
F    2              0 0 0 0 2 2 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0
I    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0
J    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0
K    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0
L    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0
M    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0


A:
B: O 1 rechts
C: O 1 links
D:
E: P 1 rechts
F: P 1 links
G:
H:
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
```

Parkplatz 4

```
cat examples/parkplatz4.txt | ./a.out
## LEFT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N O P

A    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
C    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
D    0              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
E    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
F    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
L    2              0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
M    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    2              0 0 0 0 0 0 0 0 0 0 1 1 0 2 2 0
O    2              0 0 0 0 0 0 0 0 0 0 1 1 0 2 2 0
P    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N O P

A    2              2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0
B    2              2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0
C    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0
D    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0
E    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
F    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 2 2 0 0 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 2 2 0 1 1 0
L    2              0 0 0 0 0 0 0 0 0 0 2 2 0 1 1 0
M    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
N    1              0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
O    1              0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
P    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


A: R 1 rechts, Q 1 rechts
B: R 2 rechts, Q 2 rechts
C: R 1 rechts
D: R 2 rechts
E:
F:
G: S 1 rechts
H: S 1 links
I:
J:
K: T 1 rechts
L: T 1 links
M:
N: U 1 rechts
O: U 1 links
P:
```

Parkplatz 5

```
cat examples/parkplatz5.txt | ./a.out
## LEFT
```

```
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N O

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
C    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0 0
D    2              0 0 2 2 0 0 0 0 0 0 0 0 0 0 0
E    2              0 0 2 2 2 2 0 0 0 0 0 0 0 0 0
F    2              0 0 2 2 2 2 0 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
L    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M    2              0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0
N    2              0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0
O    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


## RIGHT
name possible moves shifts
==== ============== ======
                    A B C D E F G H I J K L M N O

A    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
B    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
C    2              0 0 2 2 2 2 0 0 0 0 0 0 0 0 0
D    2              0 0 2 2 2 2 0 0 0 0 0 0 0 0 0
E    2              0 0 0 0 2 2 0 0 0 0 0 0 0 0 0
F    2              0 0 0 0 2 2 0 0 0 0 0 0 0 0 0
G    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
H    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
I    2              0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
J    2              0 0 0 0 0 0 0 0 0 2 2 0 0 0 0
K    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
L    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
M    1              0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
N    1              0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
O    2              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


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

Unlösbares Beispiel

```
cat examples/my_parkplate0.txt | ./a.out
## LEFT
name possible moves shifts
==== ============== ======
                    A B

A    0              0 0
B    0              0 0
```

```
10
   ## RIGHT
12 name possible moves shifts
   ==== ============== ======
14                      A B

16 A    0               0 0
   B    0               0 0
18

20 A: unmoeglich
   B: unmoeglich
```

Dieses Beispiel besteht aus einem Parkplatz mit zwei Feldern, auf dem ein Q-Auto steht. Es ist offensichtlich, dass keins der Autos ausfahren kann.

Leeres Beispiel

```
1  cat examples/my_parkplate1.txt | ./a.out
   ## LEFT
3  name possible moves shifts
   ==== ============== ======
5                       A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

7  A    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   B    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9  C    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   D    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 E    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   F    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13 G    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   H    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 I    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   J    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 K    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   L    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
19 M    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   N    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21 O    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   P    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
23 Q    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   R    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
25 S    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   T    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
27 U    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   V    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
29 W    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   X    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
31 Y    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   Z    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
33

35 ## RIGHT
   name possible moves shifts
37 ==== ============== ======
                        A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
39
   A    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
41 B    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   C    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
43 D    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   E    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
45 F    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   G    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
47 H    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   I    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
49 J    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   K    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
51 L    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
   M    2               0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
N     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
O     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
P     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Q     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
R     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
S     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
T     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
U     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
V     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
W     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
X     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Y     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Z     2                0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


A:
B:
C:
D:
E:
F:
G:
H:
I:
J:
K:
L:
M:
N:
O:
P:
Q:
R:
S:
T:
U:
V:
W:
X:
Y:
Z:
```

Wenn keine Q-Autos vorhanden sind, muss auch keins verschoben werden.