Aufgabe 1: Schiebeparkplatz

Team-ID: 00067

Team-Name: Panic! at the Kernel

Bearbeiter/-innen dieser Aufgabe:
Christopher Besch

13. November 2021

Inhaltsverzeichnis

1 Lösungsidee

Im Folgenden werden querstehende Autos mit „Q-Auto“ abgekuerzt; alle nicht querstehenden Autos werden „Auto“ genannt. Ein Feld—eine moegliche Position fuer ein 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 laesst 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 Laenge eines Q-Autos zwei entspricht. Diese Verschiebungen werden ignoriert.

2. Das Feld kann genau einfach nach links verschoben werden.

3. Das Feld kann nicht nach links verschoben werden.

Diese Kategorien gelten analog fuer Rechtsverschiebungen.

Nun kann fuer jedes Feld A bestimmt werden, welche anderen Felder wieweit verschoben werden muessen, 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 fuer 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 es 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 kann. Die reduzierte Verschiebungsbestimmung entspricht der vollen Verschiebungsbestimmung Λ, wobei jeder Wert e von Λ um 1 reduziert wird, wenn $e > 0$. Dies laesst sich dadurch zeigen, dass wenn ein Q-Auto einmal weniger verschoben werden muss alle anderen notwendigen Verschiebungen ebenfalls einmal weniger durchgefuehrt

werden muessen. Die fuer eine bestimmte Verschiebung notwendige Verschiebungsbestimmung (entweder voll oder reduziert) wird notwendige Verschiebungsbestimmung genannt.

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

- Ob ein beliebiges Auto ausfahren kann: Dies ist der Fall wenn mindestens eine Kategorie des betroffenen Feldes mindestens Kategorie 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 Haelfte der Menge an Elemente e in der notwendigen Verschiebungsbestimmung mit $e > 0$.
- Welche Q-Autos wie verschoben werden muessen, 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.

1.1 Bestimmung der Kategorien und Verschiebungsbestimmungen

Das Verfahren wird fuer die Linksverschiebungen beschrieben, kann allerdings ebenfalls fuer 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 uebernaechste 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.

Fuer die Verschiebungsbestimmungen kann ein aehnliches Verfahren angewendet werden. Die Summe α zweier Verschiebungsbestimmungen Λ und Γ entspricht fuer jedes Element aus α dem Maximum der jeweiligen Element aus Λ und Γ.

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

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

Nun wird das Prinzip der dynamischen Programmierung verwendet. Es laesst sich erkennen, dass die Information eines Feldes immer nur von den Feldern zur Linken abhaengt. Daher koennen alle Felder von links nach rechts durchgegangen werden und der Kategorie und Verschiebungsbestimmung mithilfe der bereits berechneten Werte bestimmt werden.

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

2 Umsetzung und Quelltext

Die Loesungsidee wird in C++ implementiert. Die Kategorie wird als vector repraesentiert:

```
// 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 k `left_shift[a][b]` die dafuer benoetigte 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 ueberschreibt `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` fuehrt 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.

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

3 Beispiele

Parkplatz 0

```
cat examples/parkplatz0.txt | ./a.out
## LEFT
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 2   0 0 2 2 0 0 0
D 2   0 0 2 2 0 0 0
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 0 2 2 0 0
L 2   0 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 0
N 2   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
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 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 0 1 1 1 1 1 1 0 2 2
N 2   0 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 1 1 1 1 0 0 0
I 1   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 1 1 0 0 0
K 1   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
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:
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 0 2 2 0 0 0
L 2   0 0 0 0 0 0 0 0 0 0 0 2 2 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 0 1 1 0 2 2
O 2   0 0 0 0 0 0 0 0 0 0 0 1 1 0 2 2
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 0 2 2 0 1 1
L 2   0 0 0 0 0 0 0 0 0 0 0 2 2 0 1 1
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 0 1 1
O 1   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
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 2 2 0 0 0 0 0
J 2   0 0 0 0 0 0 0 0 2 2 0 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 2 2 0
N 2   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

## 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 2 2 0 0 0 0 0
J 2   0 0 0 0 0 0 0 0 2 2 0 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 1 1 0
N 1   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


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:
```

Unloesbares Beispiel

```
cat examples/my_parkplate0.txt | ./a.out
## LEFT
name  possible moves  shifts
====  ==============  ======
      A B
A 0   0 0
B 0   0 0


## RIGHT
name  possible moves  shifts
====  ==============  ======
      A B
```

```
A 0   0 0
B 0   0 0

A: unmoeglich
B: unmoeglich
```

Leeres Beispiel

```
cat examples/my_parkplate1.txt | ./a.out
## LEFT
name  possible moves  shifts
====  ==============  ======
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
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
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
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
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
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
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
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

## RIGHT
name  possible moves  shifts
====  ==============  ======
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
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
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
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
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
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
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
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 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 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:
```