40. Bundeswettbewerb für Informatik - Aufgabe 1 (Schiebeparkplatz)

Aufgabenstellung

Schreibe ein Programm, das eine Situation auf einem Schiebeparkplatz mit einer Parkreihe einliest und für jeden Wagen auf einem normalen Platz bestimmt, welche andere Wagen wie verschoben werden müssen, damit er ausfahren kann. Dabei sollen möglichst wenige verschoben werden.

Lösungsidee

Das Hauptziel des Algorithmuses ist, dass ein gegebenes Fahrzeug den Schiebeparkplatz verlassen kann. Dabei sollen alle benötigten Lösungsschritte dokumentiert und vorallem so minimal wie möglich gehalten werden. Damit ein vertikal geparktes Fahrzeug den Parkplatz verlassen kann, darf sich unter diesem kein horizontal geparktes Fahrzeug befinden. Somit müssen also erst alle horizontalen Fahrzeuge möglichst optimal verschoben werden, damit das Fahrzeug ausfahren kann. In manchen Fällen ist es daher erforderlich, mehrere horizontal geparkte Fahrzeuge zu verschieben.

Verwendung des Programmes

Das Programm zur Lösung dieses Problemes befindet sich in der Datei "Aufgabe_1.jar". Das Programm kann mit der Befehlsszeile (CMD auf Windows, bzw. Terminal auf MacOS) ausgeführt werden.
Dafür navigiert man zuerst in den Ordner der JAR-Datei (hier Aufgabe 1). Anschließend führt man den Befehl "java -jar Aufgabe_1.jar <Eingabedatei> <Ausgabedatei>" aus.
Die Eingabedateien befindet sich in dem Ordner "src/main/resources/" (Beispiel: "src/main/resources/parkplatz0.txt").
Die Ausgabedatei ist optional. Wenn keine Ausgabedatei angegeben ist, dann wird das Ergebnis als "output.txt" in dem Ordner der JAR-Datei gespeichert.

Implementierung

Einlesen der Beispieldateien

Der folgende Code liest eine der Beispieldateien ein und verwendet die erhaltenen Informationen um ein Objekt der "ParkingArea"-Klasse erstellen zu können. Diese Klasse beeinhaltet alle Informationen zum Parkplatz und ist die Hauptklasse des Algorithmuses.

```
//Lesen der einzelnen Zeilen der Datei
String line;
while((line = streamReader.readLine()) != null)
contentBuilder.append(line).append("\n");

//Aufteilen des gelesenen Textes in die einzelnen Zeilen
String[] contentLines = contentBuilder.toString().split("\n");

//Interpretieren der Daten
```

```
//Diese Anweisung berechnet die Breite des Parkplatzes, indem der zweite
Buchstabe in ASCII konvertiert und vom Ergebnis 96 subtrahiert wird.
//Somit erhält man die Stelle des Buchstabens im Alphabet.
int verticalCars = (int)Character.toLowerCase(contentLines[0].charAt(2)) -
96;

//Initialisieren und Befüllen des Parkplatzes mit den gegebenen Daten
parkingSpots = new String[2][verticalCars];

//Diese Schleife füllt die erste Reihe des Arrays mit den Buchstaben A bis
<Zielbuchstabe>, welche vorher aus der Datei gelesen wurden.
//Dabei wird die Iterationsvariable i, der Schleife, in einen ASCII-Code
umgewandelt.
for(int i = 0; i < verticalCars; i++) parkingSpots[0][i] =
Character.toString(Character.toUpperCase((char)i + 97));

//Diese Schleife liest die horizontal geparkten Fahrzeuge aus der
gegebenen Datei und füllte diese in das Array.
//Dabei nimmt ein horizontal geparktes Fahrzeug immer zwei Einheiten ein
(i und i + 1).
int horizontalCars = Integer.parseInt(contentLines[1]);
for(int i = 0; i < horizontalCars; i++) {
    String horizontalCar = contentLines[i + 2];
    int startPosition = Integer.parseInt(horizontalCar.split(" ")[1]);

    parkingSpots[1][startPosition] = horizontalCar.split(" ")[0];
    parkingSpots[1][startPosition + 1] = horizontalCar.split(" ")[0];
}

//Diese Schleife füllt die restlichen Parkplätze mit einem Leerzeichen,
welches später benötigt wird.
for(int i = 0; i < parkingSpots[1].length; i++) {
    if(parkingSpots[1][i] == null)
        parkingSpots[1][i] = SPOT_EMPTY;
}
```

Verlassen des Parkplatzes

Die Funktion ParkingArea#leaveSpot(String car) wird verwendet, um alle Lösungsschritte zu sammeln, welche benötigt werden, damit das Fahrzeug den Parkplatz verlassen kann. Diese Funktion überprüft zuerst, ob das vertikal geparkte Fahrzeug überhaupt von einem anderen Fahrzeug blockiert wird.
Wenn dies nicht der Fall ist, kann die Prozedur abgebrochen werden, da das geparkte Fahrzeug den Parkplatz ohne Weiteres verlassen kann.
Wenn dies allerdings nicht der Fall ist, dann muss das blockierende Fahrzeug zuerst verschoben werden. Dafür wird die Funktion ParkingArea#moveHorizontalCar aufgerufen.

```
public String leaveSpot(String car) {
    //StringBuilder, welcher das Zusammenschließen der Lösungsschritte
übernimmt
    StringBuilder chainBuilder = new
```

```
StringBuilder(String.format(MOVE_PREFIX, car));

    //Ermitteln der Position des Fahrzeuges auf dem Parkplatz
    int carIndex = getVerticalCarPosition(car);

    //Das Fahrzeug wird von einem horizontalen Fahrzeug blockiert, welches
zuerst verschoben werden muss
    if(!parkingSpots[1][carIndex].equalsIgnoreCase(SPOT_EMPTY)) {
        String blockingCar = parkingSpots[1][carIndex];

        //Ragt das horizontale Fahrzeug nach links oder rechts über das
vertikale Fahrzeug hinaus?
        boolean overlappingLeft = carIndex > 1 && parkingSpots[1][carIndex
- 1].equalsIgnoreCase(blockingCar);

        //Bestimmen der Lösungsschritte, um das blockierende Fahrzeug zu
verschieben
        chainBuilder.append(
            returnSmarterMove( //Bestimmen des besseren Zuges
                moveHorizontalCar(
                    blockingCar, overlappingLeft ? -1 : 1, ""
                ), //Zug 1: Verschieben des Fahrzeuges in die optimalere
Richtung
                moveHorizontalCar(
                    blockingCar, overlappingLeft ? 2 : -2, ""
                ) //Zug 2: Verschiebenn des Fahrzeuges in die
suboptimalere Richtung
            )
        )
    }

    //Rückgabe der Schrittfolge
    return chainBuilder.toString();
}
```

```
private String moveHorizontalCar(String car, int move, String
instructions) {
    //StringBuilder, welcher das Zusammenschließen der Lösungsschritte
übernimmt
    StringBuilder chainBuilder = new StringBuilder(instructions);

    //Bestimmen der Position des horizontal geparkten Fahrzeuges auf dem
Parkplatz
    int carStartPosition = getHorizontalCarPosition(car);

    //1. Abbruchbedingung: Falls sich das horizontal geparkte Fahrzeug auf
der Position 0 befindet und nach links verschoben werden soll, dann wird
die Funktion erneut, aber mit umgekehrter Bewegungsrichtung (nach rechts)
aufgerufen, da sich links von dem Fahrzeug die Wand befindet, welche nicht
überfahren werden darf.
    if(move < 0 && carStartPosition == 0) return moveHorizontalCar(car, 2,
```

```
instructions);

    //2. Abbruchbedingung: Falls sich das horizontal geparkte Fahrzeug auf
der letzten Position befindet und nach rechts verschoben werden soll, dann
wird die Funktion erneut, aber mit umgekehrter Bewegungsrichtung
aufgerufen (nach links), da sich rechts von dem Fahrzeug die Wand
befindet, welche nicht überfahren werden darf.
    if(move > 0 && carStartPosition + 1 == parkingSpots[1].length - 1)
return moveHorizontalCar(car, -2, instructions);

    //Generieren des Lösungsschrittes, zum Verschieben des Fahrzeuges
(Beispiel: "A 2 rechts")
    String change = String.format(MOVE_PATTERN, car, Math.abs(move), (move
> 0 ? MOVE_RIGHT : MOVE_LEFT));

    //Hinzufügen des Lösungsschrittes zu den bisherigen Lösungsschritten
    //Dabei wird der Lösungsschritt allerdings am Anfang eingefügt, da
dieser ausgeführt werden muss, bevor das vorherige Fahrzeug verschoben
werden kann
    chainBuilder.insert(0, change);

    if(canMove(carStartPosition, move)) {
        //Das horizontale Fahrzeug kann, ohne ein weiteres Fahrzeug zu
bewegen, bewegt werden, um dem senkrechten Fahrzeug Platz zu machen.
        //Rückgabe des benötigten Schrittes, um das horizontale Fahrzeug
zu verschieben
        return chainBuilder.toString().startsWith(MOVE_SEPARATOR) ?
chainBuilder.substring(2) : chainBuilder.toString();
    } else {
        //Das horizontale Fahrzeug kann nur bewegt werden, wenn ein
weiteres horizontales Fahrzeug bewegt wird (rekursiver Funktionsaufruf).
        //Die Funktion wird erneut rekursiv aufgerufen, damit das
Fahrzeug, welches das zu Verschieben Fahrzeug blockiert, verschoben werden
kann.
        return moveHorizontalCar(getNextCar(carStartPosition, move),
calculateDelta(carStartPosition, move), chainBuilder.toString());
    }
}
```

Beispiele

Beispiel 1 (parkplatz0.txt)

Eingabe:

A G
2
H 2
I 5

Generierter Parkplatz des Programmes:

A	B	C	D	E	F	G
-	-	H	H	-	I	I

Generierte Lösung des Programmes:

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

Analyse der Lösung:

A: Dieses Fahrzeug kann ohne Weiteres bewegt werden.
B: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

C: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

A	B	C	D	E	F	G
-	-	H	H	-	I	I

A	B	C	D	E	F	G
-	-	-	H	H	I	I

D: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

A	B	C	D	E	F	G
-	-	H	H	-	I	I

A	B	C	D	E	F	G
-	H	H	-	-	I	I

E: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

F: Es müssen zwei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

A	B	C	D	E	F	G
-	-	H	H	-	I	I

A	B	C	D	E	F	G
-	H	H	-	-	I	I

A	B	C	D	E	F	G
-	H	H	-	I	I	-

A	B	C	D	E	F	G
-	H	H	I	I	-	-

G: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

A	B	C	D	E	F	G
-	-	H	H	-	I	I

A	B	C	D	E	F	G
-	-	H	H	I	I	-

Beispiel 2 (parkplatz1.txt)

Eingabe:

A N
4
O 1
P 3
Q 6
R 10

Generierter Parkplatz des Programmes:

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

Generierte Lösung des Programmes:

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:

Analyse der Lösung:

A: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

B: Es müssen zwei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

C: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

D: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

E: Es müssen zwei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

F: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

G: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

H: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

I: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

J: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

K: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

L: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

M: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

N: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

Beispiel 3 (parkplatz2.txt)

Eingabe:

A N
5
O 2
P 5
Q 7
R 9
S 12

Generierter Parkplatz des Programmes:

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

Generierte Lösung des Programmes:

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

Analyse der Lösung:

A: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

B: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

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

C: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

D: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

E: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

F: Es müssen zwei weitere Fahrzeuge bewegt werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

G: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

H: Es müssen zwei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

I: Es müssen zwei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

J: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

K: Es müssen drei weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

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

L: Dieses Fahrzeug kann ohne Weiteres bewegt werden.

M: Es müssen vier weitere Fahrzeuge verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

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

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

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

N: Es muss ein weiteres Fahrzeug verschoben werden, damit dieses Fahrzeug bewegt werden kann.

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

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

Beispiel 4 (parkplatz3.txt)

Eingabe:

A N
5
O 1
P 4
Q 8
R 10
S 12

Generierter Parkplatz des Programmes:

Generierte Lösung des Programmes:

Analyse der Lösung:

Beispiel 5 (parkplatz4.txt)

Eingabe:

A P
5
Q 0
R 2
S 6
T 10
U 13

Generierter Parkplatz des Programmes:

Generierte Lösung des Programmes:

Analyse der Lösung:

Beispiel 6 (parkplatz5.txt)

Eingabe:

A O
4
P 2
Q 4
R 8
S 12

Generierter Parkplatz des Programmes:

Generierte Lösung des Programmes:

Analyse der Lösung: