Lösungsidee

In einer Klasse Main werden die Eingabedateien gelesen.
Der Rahmen der Eingabedatei wird gelöscht und nur die Seite und Position der Öffnung gespeichert. Der Rest wird in einem zweidimensionalen Feld aus ganzen Zahlen gespeichert. Freie Plätze werden durch Null ersetzt, alle Zahlen werden inkrementiert, da in den Eingabedateien die Zahl des ersten Stäbchens Null ist.
Anschließend wird mit dem Feld und den Daten der Öffnung ein Objekt der Klasse Puzzle erstellt und dieses gelöst.
Die einfachste Möglichkeit wäre, mit einer rekursiven Methode per „Brute-Force“ auf die Lösung zu kommen. Um festzustellen, ob es überhaupt eine Lösung gibt werden alle Felder, die während dem Drehvorgang entstehen, gespeichert. Wenn nun ein Feld bereits existierte, wird der Rechenvorgang bei diesem Feld nicht fortgesetzt.
Allerdings ist hierbei der Rechenaufwand nicht akzeptabel und das Herausfinden der kürzesten Lösung umständlich, da durch das rekursive System zunächst so lange wie möglich nur in eine Richtung gedreht wird.
Stattdessen werden alle Möglichkeiten überprüft, die bei einer gewissen Anzahl an Drehungen auftreten. Gestartet wird bei einer Drehung und geendet wird, wenn es keine neuen Felder mehr gibt, oder die schnellste Lösung gefunden wurde. Dies funktioniert mit einer bedingten Wiederholung. Alle möglichen Felder und die dazugehörigen Drehfolgen einer Runde werden in einer Liste gespeichert. In der nächsten Runde werden diese dann verwendet.
Nun werden die ersten beiden Beispiele in Sekundenbruchteilen gelöst, die dritte dauert auf einem gewöhnlichen Rechner immer noch ca. 50 Minuten. Dies liegt daran, dass die Überprüfung, ob das Feld neu ist, ab einer gewissen Anzahl an Feldern sehr lange dauert.
Aus diesem Grund werden die alten Felder auf andere Art und Weise gespeichert: Man gruppiert die Felder anhand der Seite der Öffnung und der Position der einzelnen Stäbchen in der rekursiven Datenstruktur Baum.
Hierfür gibt es vier Listen für die vier Seiten, an denen die Öffnung sein kann. Jede Liste speichert dann alle bereits vorgekommenen Positionen des ersten Stäbchen. Dieses speichert alle Positionen des zweiten Stäbchen, wenn das erste Stäbchen an dieser Position war, usw.
Somit wird das dritte Beispiel auch innerhalb von wenigen Sekunden gelöst.
Für den Drehvorgang selbst gibt es eine Methode, die für jede Position des Feldes beschreibt, auf welcher Position deren Wert im neuen Feld ist. Anschließend wird die Schwerkraft simuliert. Hierfür wird für jede einzelne Reihe (außer der untersten) von unten startend untersucht, ob und wenn ja wie oft Stäbchen nach unten verschoben werden können. Für horizontal liegende Stäbchen muss überprüft werden, ob alle Positionen unter dem Stäbchen frei sind.

Beispiele

Die Lösungen für die verfügbaren Beispiele sind:

rotation1.txt: LLLLRL
rotation2.txt: RRRRLRRLLLLLLLRRRRRRLR
rotation3.txt:
LRRRRLLLRRRLLLLLLLRRLLRRRRRRRLRRRRRRRRRRRLLLRRRRRLLLLLRRRRLLRRRRRRRLRLLLLLLLLLLLRLLRRRRRLR

Quelltext

Die Suche nach der schnellsten Lösung:

```
    public String bestSolution(){
//Sobald ein neues Feld in dieser Runde entsteht, ändert sich der Wert zu wahr und es gibt, wenn noch keine Lösung gefunden wurde, eine weitere Runde
        boolean done;
        do{
```

```
            done = true;
//Anzahl der Befehle der letzten Runde
            int i=lastOrders.size();
//Hier werden alle neuen Felder und die dazugehörige Drehfolge gespeichert. Diese werden in der nächsten Runde benötigt
            Vector<String> newLastOrders = new Vector();
            Vector<int[][]> newLastOrdersField = new Vector();
//Jedes Feld der letzten Runde wird in beide Richtungen gedreht. Anschließend wird überprüft, ob die Lösung gefunden wurde und ob das Feld neu ist
            for(int j=0; j<i;j++){
               int[][] newField = change(true,lastOrdersField.get(j));
               int state = check(newField,lastOrders.get(j)+"L");
//Wenn die Lösung gefunden wurde, wird die Drehfolge zurückgegeben
               if(state==0){
                  return lastOrders.get(j)+"L";
//Wenn das Feld neu ist, wird es für die nächste Runde gespeichert
               }else if(state==1){
                  newLastOrders.add(lastOrders.get(j)+"L");
                  newLastOrdersField.add(newField);
               }
               newField = change(false,lastOrdersField.get(j));
               state = check(newField,lastOrders.get(j)+"R");
//Wenn die Lösung gefunden wurde, wird die Drehfolge zurückgegeben
               if(state==0){
                  return lastOrders.get(j)+"R";
//Wenn das Feld neu ist, wird es für die nächste Runde gespeichert
               }else if(state==1){
                  newLastOrders.add(lastOrders.get(j)+"R");
                  newLastOrdersField.add(newField);
                  done = false;
               }
            }
//Die neuen Felder entstehen
            lastOrders=newLastOrders;
            lastOrdersField=newLastOrdersField;
         }while(!done);
         return "No solution found!";
      }
```

Berechnung der Position der Stäbchen:

```
      private Piece makeField(int[][] field)
      {
//x- und y-Koordinaten der Stäbchen (links oder oben) und deren Lage
         int[] x = new int[pieces];
         int[] y = new int[pieces];
         boolean[] horizontal = new boolean[pieces];
//Zur Überprüfung, ob die Position eines Stäbchens schon gefunden wurde
         for(int i=0; i<pieces; i++)
         {
            x[i] = -1;
         }
//Für jede Position des Feldes wird überprüft, ob ein noch nicht identifiziertes Stäbchen dort ist. Falls ja, wird es gespeichert
         for(int i=0; i<field.length; i++)
         {
            for(int j=0; j<field[0].length; j++)
            {
               if(field[i][j]!=0 && x[field[i][j]-1]<0)
               {
                  int number = field[i][j]-1;
                  x[number] = j;
                  y[number] = i;
```

```
                horizontal[number] = (j+1<field[0].length && field[i][j+1]-1==number);
            }
        }
    }
//Speichert alle Stäbchen in der Datenstruktur und gibt das erste zurück
    return new Piece(x,y,horizontal);
  }
```

Drehen des Feldes:

```
//Jede Wert bekommt im neuen Feld die jeweilige neue Position
  private int[][] change(boolean left, int[][] field){
    int[][] newField = new int[field.length][field[0].length];
    for(int i=0; i<field.length; i++){
      for(int j=0; j<field[0].length;j++){
        if(left){
          newField[i][j] = field[j][field.length-1-i];
        }else{
          newField[i][j] = field[field.length-1-j][i];
        }
      }
    }
//Schwerkraft wird simuliert
    return gravity(newField);
  }
//Jede Reihe wird einzeln von unten startend untersucht
  private int[][] gravity(int[][] field){
    for(int i=field.length-2;i>=0;i--)
    {
      field = barGravity(i,0,field);
    }
    return field;
  }

  private int[][] barGravity(int i,int j,int[][] field){
//Leere Positionen werden übersprungen
    while(j+1<field.length && field[i][j]==0)
    {
      j++;
    }
    int k=1;
    if(field[i][j]!=0){
//Wenn Stäbchen horizontal liegen, müssen alle Positionen darunter frei sein
      while(j+k<field[0].length && field[i][j]==field[i][j+k]){
        k++;
      }
      boolean possible=true;
      int m=1;
//Solange möglich, Bewegung nach unten
      while(possible){
        for(int l=j;l<j+k;l++){
          if(field[i+m][l]!=0)
          {
            possible = false;
          }
        }
        if(possible){
          for(int l=j;l<j+k;l++){
            field[i+m][l]=field[i+m-1][l];
            field[i+m-1][l]=0;
          }
          if(i+m+1<field.length)
          {
```

```
                m++;
            }
            else
            {
                possible = false;
            }
        }
    }
//Wenn Ende der Reihe nicht erreicht, wiederholen
    if(k+j<field[0].length)
    {
        field = barGravity(i,k+j,field);
    }
  }
  return field;
 }
```

Überprüfung, ob das Feld neu ist:

```
//Diese Methode wird nur aufgerufen, wenn dieses Stäbchen gleich ist wie das in dem neuen Feld
public boolean isNewField(int[] x,int[] y,boolean[] horizontal)
 {
//Wenn das letzte Stück erreicht wurde
  if(nextPieces==null)
  {
    return false;
  }
//Überprüfung, ob ein Nachfolger die Koordinaten des nächsten Stäbchen hat
  for(int i=0; i<nextPieces.size(); i++)
  {
    if(nextPieces.get(i).same(x[1],y[1],horizontal[1]))
    {
//Wenn ja, gibt dieser zurück, ob das Feld neu ist und fügt gegebenenfalls ein
      return nextPieces.get(i).isNewField(Arrays.copyOfRange(x,1,x.length),
        Arrays.copyOfRange(y,1,x.length),Arrays.copyOfRange(horizontal,1,x.length));
    }
  }
//Ansonsten ist das Feld neu und die Stäbchen werden eingefügt
  nextPieces.add(new Piece(Arrays.copyOfRange(x,1,x.length),
      Arrays.copyOfRange(y,1,x.length),Arrays.copyOfRange(horizontal,1,x.length)));
  return true;
 }
```