Complexiteit & Algoritmiek

Backtracking

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | **Joram Sjamaar** |  | **Stijn van den Berg** |  |
|  | 462398 |  | 466072 |  |

Inhoudsopgave

[Doolhof 2](#_Toc34306360)

[Ontwerp 2](#_Toc34306361)

[DirectedLine 2](#_Toc34306362)

[Maze 2](#_Toc34306363)

[Node 2](#_Toc34306364)

[Pawn 2](#_Toc34306365)

[State 2](#_Toc34306366)

[Data representatie van een doolhof 3](#_Toc34306367)

[Ondersteuning voor meerdere doolhoffen 4](#_Toc34306368)

[Specificatie 4](#_Toc34306369)

[Voorbeeld 4](#_Toc34306370)

[Tests 5](#_Toc34306371)

[totalStepsShouldBe37 5](#_Toc34306372)

[emptyMazeShouldThrowException 5](#_Toc34306373)

[mazeWithNoSolutionShouldThrowException 5](#_Toc34306374)

[simpleMazeShouldTraverse 5](#_Toc34306375)

[Resultaat 6](#_Toc34306376)

[Waarom nu wel? 11](#_Toc34306377)

[Bijlagen 12](#_Toc34306378)

[Class Diagram 12](#_Toc34306379)

# Doolhof

In deze opdracht hebben we gewerkt met backtracking. Dit is een ‘slimme’ methode om informatie mee op te zoeken. Het is slim omdat niet elke oplossing berekend hoeft te worden.

## Ontwerp

De opdracht omschreef een doolhof met twee pionnen. Een pion mag alleen over een lijn van de kleur waar de één van de pionnen op staan.

Dat betekend dat we een aantal model classes hebben. Namelijk:

* DirectedLine
* Maze
* Node
* Pawn
* State

**Bekijk ook het Class Diagram in de bijlagen**

DirectedLine  
Heeft twee eigenschappen: Kleur en welke Node deze lijn verbind.

Maze  
Bevat alle logica en backtracking algoritme.

NodeStelt een blokje/rondje voor in het doolhof. Heeft twee eigenschappen: Een nummer en kleur.

Pawn  
Stelt een pion voor. Deze class is niet nodig. Ik heb toch voor deze class gekozen omdat het mij duidelijkheid gaf. De pion houd alleen bij op welke Node deze staat.

State  
Houd de toestand bij van het doolhof.  
  
De meest makkelijke vergelijking die ik kan maken, is een foto. Stel je voor dat je een foto maakt van een schaakspel. Je ziet dan precies waar alle spelstukken staan. Dit doet de State ook. Het slaat op waar de pionnen staan.

## Data representatie van een doolhof

Ik heb gekozen voor een adjacency list. Een node is hierin het hoofd element. Een node bevat directedlines. Zie tabel hieronder

|  |  |
| --- | --- |
| Node | DirectedLines |
| FINISH (BLUE) |  |
| 1 (PINK) | [PINK, pointsTo: Node4 (GREEN)] [BLACK, pointsTo: Node5 (GREEN)] |
| 2 (BLACK) | [GREEN, pointsTo: Node6 (ORANGE)] [PINK, pointsTo: Node12 (PINK)] |
| 3 (GREEN) | [ORANGE, pointsTo: Node4 (GREEN)] [ORANGE, pointsTo: Node1 (PINK)] |
| 4 (GREEN) | [BLACK, pointsTo: Node13 (ORANGE)] |
| 5 (GREEN) | [ORANGE, pointsTo: Node9 (PINK)] |
| 6 (ORANGE) | [PINK, pointsTo: Node10 (BLACK)] [GREEN, pointsTo: Node9 (PINK)] |
| 7 (ORANGE) | [GREEN, pointsTo: Node2 (BLACK)] |
| 8 (PINK) | [PINK, pointsTo: Node3 (GREEN)] |
| 9 (PINK) | [BLACK, pointsTo: Node14 (GREEN)] [GREEN, pointsTo: Node4 (GREEN)] |
| 10 (BLACK) | [GREEN, pointsTo: Node15 (ORANGE)] |
| 11 (ORANGE) | [PINK, pointsTo: Node10 (BLACK)] [GREEN, pointsTo: Node12 (PINK)] |
| 12 (PINK) | [GREEN, pointsTo: Node7 (ORANGE)] |
| 13 (ORANGE) | [GREEN, pointsTo: Node8 (PINK)] [GREEN, pointsTo: Node18 (BLACK)] |
| 14 (GREEN) | [GREEN, pointsTo: Node-1 (BLUE)] [ORANGE, pointsTo: Node20 (GREEN)] |
| 15 (ORANGE) | [GREEN, pointsTo: Node22 (BLACK)] [PINK, pointsTo: Node-1 (BLUE)] |
| 16 (GREEN) | [GREEN, pointsTo: Node15 (ORANGE)] |
| 17 (GREEN) | [BLACK, pointsTo: Node11 (ORANGE)] [PINK, pointsTo: Node12 (PINK)] [BLACK, pointsTo: Node16 (GREEN)] |
| 18 (BLACK) | [ORANGE, pointsTo: Node9 (PINK)] [ORANGE, pointsTo: Node20 (GREEN)] |
| 19 (ORANGE) | [GREEN, pointsTo: Node18 (BLACK)] |
| 20 (GREEN) | [BLACK, pointsTo: Node19 (ORANGE)] [ORANGE, pointsTo: Node21 (BLACK)] |
| 21 (BLACK) | [ORANGE, pointsTo: Node22 (BLACK)] [BLACK, pointsTo: Node-1 (BLUE)] |
| 22 (BLACK) | [ORANGE, pointsTo: Node17 (GREEN)] |

## Ondersteuning voor meerdere doolhoffen

De ondersteuning voor meerdere doolhoffen wordt gerealiseerd met een JSON inlezing van een doolhof. Dit wordt geplaatst in een adjacency list .

De maze wordt standaard ingelezen vanuit het *maze.json* bestand.

Dit JSON bestand bevat een array, met daarin Nodes en de daarbij behoordende properties

### Specificatie

Array

* Object *(Node)*
  + Number *(int)*
  + Color *(“PINK” | “GREEN” | “BLACK” | “ORANGE” | “BLUE”)*
  + Lines *(Array)*
    - Object
      * Color *(“PINK” | “GREEN” | “BLACK” | “ORANGE” | “BLUE”)*
      * pointsTo *(int)*

### Voorbeeld

[  
 {  
 "number": 1,  
 "color": "PINK",  
 "lines": [  
 {  
 "color": "PINK",  
 "pointsTo": 4  
 },  
 {  
 "color": "BLACK",  
 "pointsTo": 5  
 }  
 ]  
 }

]

# Tests

Om de werking van het backtracking algoritme en het doolhof te testen maken we gebruik van JUnit5.4.

Er zijn drie unittesten.

### totalStepsShouldBe37

Doolhof: *maze.json*

Deze test zal de functionaliteit van het backtracking algoritme testen. De kortst mogelijke oplossing voor dit doolhof is 37 stappen.

**Let op!** Er zit een bug in deze test. Als deze test tegelijkertijd met meerdere tests wordt gedaan dan neemt deze een andere ‘afslag’. Op stap 26 gaat deze dan naar links i.p.v. naar rechts. Als je deze test alleen runt dan is de bug er niet.

### emptyMazeShouldThrowException

Doolhof: *empty\_maze.json*

Deze test verwacht een foutmelding. Als er een leeg doolhof wordt ingelezen, dan moet dit een fout opleveren.

### mazeWithNoSolutionShouldThrowException

Doolhof: *impossible\_maze.json*

Deze test verwacht een foutmelding. Als er doolhof zonder oplossing wordt ingelezen, dan moet dit een fout opleveren.

### simpleMazeShouldTraverse

Doolhof: *simple\_maze.json*

Deze test maakt gebruik van een heel simpel doolhof. En test daarmee ook weer het backtracking algoritme.

# Resultaat

Het resultaat beperk ik tot de hoofd opdracht. Dat is de maze.json.

Als we het algoritme uittesten dan komen we uit op 37 stappen.

**Step 1**

Pawn 1: 1, PINK

Pawn 2: 2, BLACK

**Step 2**

Pawn 1: 5, GREEN

Pawn 2: 2, BLACK

**Step 3**

Pawn 1: 5, GREEN

Pawn 2: 6, ORANGE

**Step 4**

Pawn 1: 9, PINK

Pawn 2: 6, ORANGE

**Step 5**

Pawn 1: 9, PINK

Pawn 2: 10, BLACK

**Step 6**

Pawn 1: 14, GREEN

Pawn 2: 10, BLACK

**Step 7**

Pawn 1: 14, GREEN

Pawn 2: 15, ORANGE

**Step 8**

Pawn 1: 20, GREEN

Pawn 2: 15, ORANGE

**Step 9**

Pawn 1: 21, BLACK

Pawn 2: 15, ORANGE

**Step 10**

Pawn 1: 22, BLACK

Pawn 2: 15, ORANGE

**Step 11**

Pawn 1: 17, GREEN

Pawn 2: 15, ORANGE

**Step 12**

Pawn 1: 17, GREEN

Pawn 2: 22, BLACK

**Step 13**

Pawn 1: 11, ORANGE

Pawn 2: 22, BLACK

**Step 14**

Pawn 1: 11, ORANGE

Pawn 2: 17, GREEN

**Step 15**

Pawn 1: 12, PINK

Pawn 2: 17, GREEN

**Step 16**

Pawn 1: 7, ORANGE

Pawn 2: 17, GREEN

**Step 17**

Pawn 1: 2, BLACK

Pawn 2: 17, GREEN

**Step 18**

Pawn 1: 2, BLACK

Pawn 2: 16, GREEN

**Step 19**

Pawn 1: 6, ORANGE

Pawn 2: 16, GREEN

**Step 20**

Pawn 1: 9, PINK

Pawn 2: 16, GREEN

**Step 21**

Pawn 1: 4, GREEN

Pawn 2: 16, GREEN

**Step 22**

Pawn 1: 4, GREEN

Pawn 2: 15, ORANGE

**Step 23**

Pawn 1: 4, GREEN

Pawn 2: 22, BLACK

**Step 24**

Pawn 1: 13, ORANGE

Pawn 2: 22, BLACK

**Step 25**

Pawn 1: 13, ORANGE

Pawn 2: 17, GREEN

**Step 26**

Pawn 1: 8, PINK

Pawn 2: 17, GREEN

**Step 27**

Pawn 1: 8, PINK

Pawn 2: 12, PINK

**Step 28**

Pawn 1: 3, GREEN

Pawn 2: 12, PINK

**Step 29**

Pawn 1: 3, GREEN

Pawn 2: 7, ORANGE

**Step 30**

Pawn 1: 3, GREEN

Pawn 2: 2, BLACK

**Step 31**

Pawn 1: 3, GREEN

Pawn 2: 6, ORANGE

**Step 32**

Pawn 1: 1, PINK

Pawn 2: 6, ORANGE

**Step 33**

Pawn 1: 1, PINK

Pawn 2: 10, BLACK

**Step 34**

Pawn 1: 5, GREEN

Pawn 2: 10, BLACK

**Step 35**

Pawn 1: 5, GREEN

Pawn 2: 15, ORANGE

**Step 36**

Pawn 1: 9, PINK

Pawn 2: 15, ORANGE

**Step 37**

Pawn 1: 9, PINK

Pawn 2: FINISH, BLUE

## Waarom nu wel?

In de tests is er wat raars aan de hand. Dan komen we uit op 54 met dezelfde code. Waarom het gebeurd weet ik niet, maar ik weet wel waar het gebeurd. Bij stap 26 gebeurd er wat raars. De Maze gaat plotseling ergens anders heen.

|  |  |
| --- | --- |
| Apl | MazeTests |
| Step 25  Pawn 1: 13, ORANGE  Pawn 2: 17, GREEN | **Step 25**  Pawn 1: 13, ORANGE  Pawn 2: 17, GREEN |
| Step 26  Pawn 1: 8, PINK  Pawn 2: 17, GREEN | **Step 26**  Pawn 1: 18, BLACK  Pawn 2: 17, GREEN |
| Step 27  Pawn 1: 8, PINK  Pawn 2: 12, PINK | **Step 27**  Pawn 1: 18, BLACK  Pawn 2: 11, ORANGE |

Waarom dit gebeurd weet ik niet. Ik krijg het er ook niet uit. Uiteindelijk op stap 43 komt het toch nog goed.

**Step 43**

Pawn 1: 17, GREEN

Pawn 2: 8, PINK

Vanaf hier gaat het door zoals in Apl (maar dan met pion 1 en 2 omgewisseld).

# Bijlagen

## Class Diagram

