

Gokie Wiegers &

Tessie van Hintum &

Annemiek Meindertsma

Presenteren:



Onder de naam:

**Contactgegevens: Twee keer namen opgeschreven ?**

Gokie Wiegers :

Tessie van Hintum :

Annemiek Meindertsma :

Contents

[1. Inleiding 3](#_Toc387180784)

[2. Methodes 5](#_Toc387180785)

[2.1 Handmatig 5](#_Toc387180786)

[2.2 Random 5](#_Toc387180787)

[2.3 Breath-First Search 6](#_Toc387180788)

[3. Resultaten 7](#_Toc387180789)

[3.1 Handmatig 7](#_Toc387180790)

[3.2 Random zoeken 7](#_Toc387180791)

[4. Conclusies 8](#_Toc387180792)

[5. Referenties/bronnen 9](#_Toc387180793)

# 1. Inleiding

Voor het vak Heuristieken heeft onze groep “Pinky and the Brains” gekozen voor de opdracht “Rush Hour”. Voor deze opdracht staat op <http://wiki.phoib.net/wiki/index.php?title=Rush_Hour> het volgende beschreven;

*“Rush Hour is een ogenschijnlijk eenvoudig puzzeltje met een verrassend uitdagend karakter. In een veld van 6 hoog en 6 breed staat een rode auto, de jouwe, en die moet naar de uitgang. die recht voor je ligt. Maar andere voertuigen versperren de weg; auto's van twee eenheden lang en trucks van drie eenheden lang, die alleen in hun rijrichting bewogen mogen worden. Ze mogen niet draaien. De opdracht is simpel: beweeg je auto naar buiten, of beter: schrijf een computerprogramma om dat voor je te doen.*

***Opdracht:***

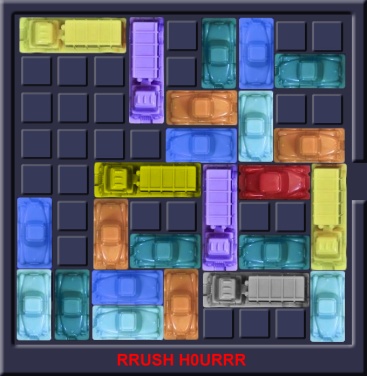
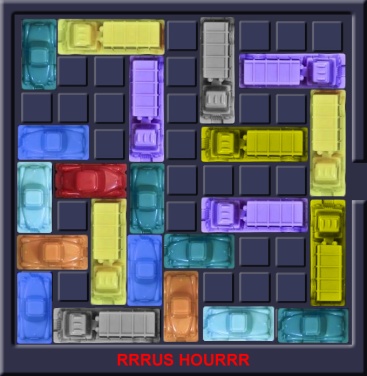
*1) Los op games 1,2 en 3 op. Doe hoe je dat goed lijkt. Het is natuurlijk goed om daar een programma te schrijven, maar als je een ander idee hebt, of het met de hand kunt, mag dat ook.*

*2) Verzin een implementatie voor het bord, en schrijf een algoritme dat series van zetten uitvoert. Laat het algoritme games 4, 5 en 6 oplossen. Kortere series zijn betere series.*

*3) Los game 7 op. Pas je algoritme aan indien dat nodig is.”*

De games 1, 2 en 3 bestaan uit oppervlaktes van 6 bij 6 eenheden gevuld met 9 tot 13 auto’s/vrachtwagens. De games 4, 5 en 6 bestaan uit oppervlaktes van 9 bij 9 eenheden gevuld met 22 tot 26 auto’s/vrachtwagens. Game 7 bestaat uit een oppervlakte van 12 bij 12 eenheden gevuld met 44 auto’s/vrachtwagens. Hieronder zijn de exacte opstellingen opgenomen in de volgorde van de games;

Games 1, 2 en 3:

Games 4, 5 en 6:

Game 7: 

De toestandsruimtegroottes is per game verschillend, omdat het aantal vakken en auto’s/vrachtwagens per game ander is.

* Voor game 1 is zal de toestandsruimte ongeveer …? Zijn
* Voor game 2 is zal de toestandsruimte ongeveer …? zijn
* Voor game 3 is zal de toestandsruimte ongeveer …? zijn
* Voor game 4 is zal de toestandsruimte ongeveer …? zijn
* Voor game 5 is zal de toestandsruimte ongeveer …? zijn
* Voor game 6 is zal de toestandsruimte ongeveer …? zijn
* Voor game 7 is zal de toestandsruimte ongeveer …? Zijn

Deze schattingen zijn gebaseerd op ….?

Rush Hour betreft een state transition problem omdat je stukken moet verschuiven in de toestandsruimte om tot een oplossing te komen en je nog niet weet hoe de oplossing eruit komt te zien. Daarnaast is Rush Hour een constrained optimization problem. omdat er beperkingen zitten in de verschuivingen, word je gedwongen om een bepaalde richting op te gaan (constrained). Daarnaast gaat het bij deze opdracht om de snelste weg te vinden naar de oplossing (optimization).

Verderop in het verslag zullen het onderzoek dat wij gedaan hebben en de oplossingen die wij hebben uitgeprobeerd worden beschreven.

# 2. Methodes

We hebben voor deze opdracht … methodes uitgevoerd om te kijken welke met een uitkomst kwam en welke met de beste uitkomst kwam. Deze methodes zullen ieder apart worden beschreven. We zullen hierin beschrijven waarom we voor deze methode hebben gekozen en hoe we de methodes hebben uitgewerkt.

## 2.1 Handmatig

We hebben de zeven games allereerst met de hand opgelost. We hebben deze methode ingezet om een inschatting te krijgen over wat een maximale oplossing zou zijn voor alle games. Deze gegevens kunnen we later in ons onderzoek gebruiken om een oordeel te vellen over de oplossingen die gevonden zijn. Wanneer dit namelijk meer graven bevat dan de handmatige oplossing weten we dat de oplossing waarschijnlijk niet het beste is.

Alle games hebben we opgelost in Excel. We hebben hier het bord in getekend en alle stappen die gedaan zijn om tot een oplossing te komen genoteerd. Het voordeel van deze methode is dat er geen kennis van programmeren aanwezig hoeft te zijn om oplossing te kunnen vinden, maar dat er wel gekeken kan worden hoe de hersenen deze games oplossen en of er in de oplossing misschien een algoritme te vinden was die we eventueel weer zouden kunnen gaan inzetten in het schrijven van een programma.

## 2.2 Random

Naast de handmatige methode zijn we ook begonnen met een programma dat ervoor zorgde dat willekeurige auto’s op het bord willekeurige kanten op gingen. Dit programma hebben we getest met game vier. Ondanks dat vooraf al duidelijk was dat deze methode niet het beste resultaat op zou gaan leveren, was het een goede test om bekender te raken met de werking achter Rush Hour. We hebben in de code geen data laten opslaan met betrekking tot bewegingen die al gedaan zijn. We hebben het programma 48 uur laten draaien om te kijken of het überhaupt tot een oplossing kwam. De code hiervoor hebben we geschreven in C en is als volgt uitgewerkt:

1. Zorg dat alle auto’s zo op het bord geplaatst zijn zoals de opdracht beschrijft;
2. Zorg ervoor dat er geen auto’s van het bord kunnen vallen;
3. Zorg dat de auto’s alleen die richting op kunnen bewegen zoals de game dit aangeeft;
4. Wanneer een auto een kant op kan bewegen laat deze dan die kant op bewegen;
5. Wanneer auto nummer één (de rode auto) voor de uitgang staat, stop het spel en geef weer hoeveel bewegingen er hebben plaatsgevonden.

## 2.3 Breadth-First Search

Als derde methode hebben we de breadth-first search (BFS) ingezet. Na wat online onderzoek bleek dat voor Rush Hour deze methode beter ingezet kan worden omdat een BFS altijd tot een oplossing komt. Dit in tegenstelling tot depth-first search (dfs) die 1 pad kiest en deze naloopt.

# 3. Resultaten

In dit hoofdstuk zullen de resultaten worden besproken van de verschillende methodes die we hebben ingezet. Elke methode zal in een aparte paragraaf worden besproken en zoveel mogelijk met cijfers en grafieken worden toegelicht. Dezelfde volgorde zal worden aangehouden als in het vorige hoofdstuk.

## 3.1 Handmatig

De handmatige oplossing van alle games kwam tot de volgende getallen;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Game 1 | Game 2 | Game 3 | Game 4 | Game 5 | Game 6 | Game 7 |
| grootte van het bord | 36 | 36 | 36 | 81 | 81 | 81 | 144 |
| aantal stukken op het bord | 9 | 13 | 13 | 22 | 25 | 26 | 44 |
| Aantal bewegingen | 55 | 14 | 21 | 38 | 33 | 47 | 30 |

Wat er in een grafiek als volgt uitziet;

Doordat de lijnen niet synchroon met elkaar lopen, is duidelijk aan te tonen dat de moeilijkheidsgraad van de game niet te maken heeft met de grootte van het bord of het aantal stukken op het bord. De moeilijkheidsgraad van een game zit hem dus in de plaatsing van de auto’s/trucs. Alhoewel deze methode niet direct heeft geleid tot een kloppend algoritme dat verder voor de opdracht kan worden ingezet, heeft deze methode wel inzicht gegeven in de werking van het spel.

## 3.2 Random zoeken

Bij het inzetten van random zoeken, hebben we inzicht gekregen hoe we Rush Hour het beste kunnen schrijven in een programma.

# 4. Conclusies

Wanneer we eindelijk een oplossing hebben, zal hier een conclusie komen over de gevonden oplossingen en waar we blij mee zijn en niet blij mee zijn.

# 5. Referenties/bronnen

Alle bronnen die ons hebben geholpen bij het vinden van een oplossing zullen we hier noteren.