Informatica door de jaren heen

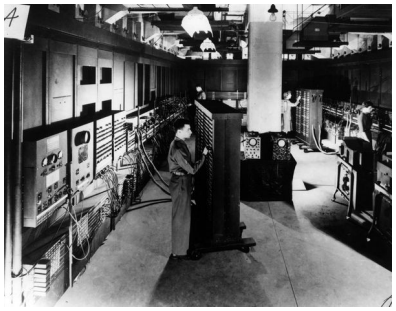
dr. Walter Kosters

Ouderdag, Leiden, zaterdag 25 maart 2023

www.liacs.leidenuniv.nl/~kosterswa/

eerste, tweede en derde jaar

van onderwijs naar onderzoek



vroeger, nu en later

Eniac

Raspberry Pi

Bij de studie krijg je per jaar een tiental vakken: de colleges.
De propedeuse Informatica, het eerste jaar van de driejarige bachelor, ziet er als volgt uit:

najaar	voorjaar
Programmeermethoden	Algoritmiek
Fundamentele informatica	Logica
Digitale systemen	Databases
Studeren en presenteren	Programmeertechnieken
Wiskunde 1	Wiskunde 2

En varianten: Informatica & Economie (I&E), Bioinformatica (BI) en Kunstmatige Intelligentie (DSAI).
Daarna: tweejarige master — Engelstalig.

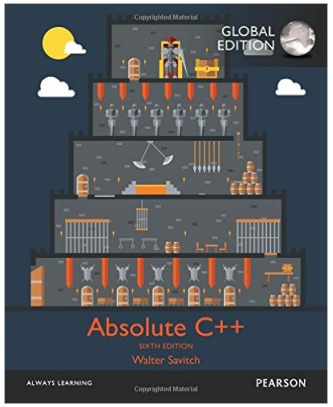
Tweede en derde jaar zijn voor Informatica als volgt:

najaar		voorjaar
Datastructuren		Kunstmatige intelligentie
Automaten		Complexiteit
Computerarchitectuur		Operating systemen
Programmeertalen		Security
Statistiek		Wetenschap & onderzoek
Vak 1 (Graphics,	M	Data mining
Vak 2 Netwerken,	I	Software engineering
Vak 3 . . . ,	N	Ethiek
Vak 4 . . . ,	O	Bachelorproject
Vak 5 . . .)	R	

Voor I&E, BI en AI: economie/biologie/psychologie-vakken.

Programmeermethoden

Je programmeert een computer in een speciale computertaal of programmeertaal, bijvoorbeeld C++, Java of Python.



In Leiden leren alle eerstejaars studenten Informatica en Wiskunde C++, en bij Informatica & Economie, Bioinformatica en Natuur/Sterrenkunde: Python. Voorkennis is niet echt nodig.

Een eerste C++-programma:

```
#include <iostream>
using namespace std;
int main ( ) {
  cout << "Dit komt op het scherm." << endl;
  return 0;
}//main
```

Dit programma zet alleen een tekstje op het beeldscherm.

Let op de — vooral voor mensen nuttige — layout.
En op hoofdletters en kleine letters.

```
// dit is een simpel programma
#include <iostream>
using namespace std;
int main ( ) {
  int getal = 42;  // een variabele
  cout << "Geef een geheel getal .. " << endl;
  cin >> getal;
  cout << "Kwadraat is: "
       << getal * getal << endl;
  return 0;
}//main
```

C++ kent de volgende controle-structuren:

keuze `if`

onbekend (maar eindig?) aantal herhalingen `while`

“vast” aantal herhalingen `for`

We gebruiken geen labels/goto’s!

. . . **I work 9–5 in a 7–11**

De gebruiker moet zijn/haar geboortejaar als geheel getal invoeren, en daarna de geboortemaand. Vervolgens voert hij/zij de geboortedag in. Het programma berekent dan de bijbehorende dag van de week.

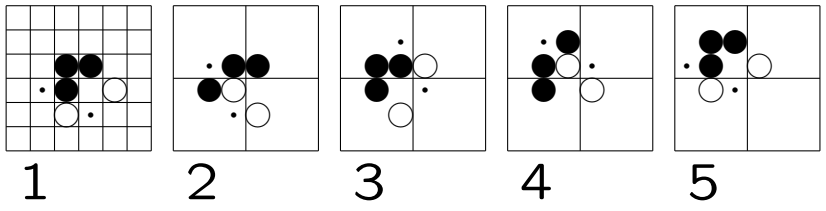
. . . **3/4.1, 0.3, 4.4, 6.6, 8.8, 10.10, 12.12**

Bijvoorbeeld: 9 november 1989
was een donderdag.

Let op schrikkeljaren;
en 1752?

Life is een "cellulaire automaat", in 1970 bedacht door John Horton Conway.

In een 2-dimensionaal oneindig groot rooster beginnen we met een eindig aantal levende vakjes oftewel cellen. Een levend vakje met minder dan 2 of meer dan 3 buren (van de 8) gaat dood, met precies 2 of 3 levende buren overleeft het. In een dood vakje met precies 3 levende buren ontstaat leven. Dit leidt tot de volgende generatie. Let erop dat dit voor alle vakjes tegelijk gebeurt.



- ● levend
- ○ gaat dood
- · (komt tot leven)

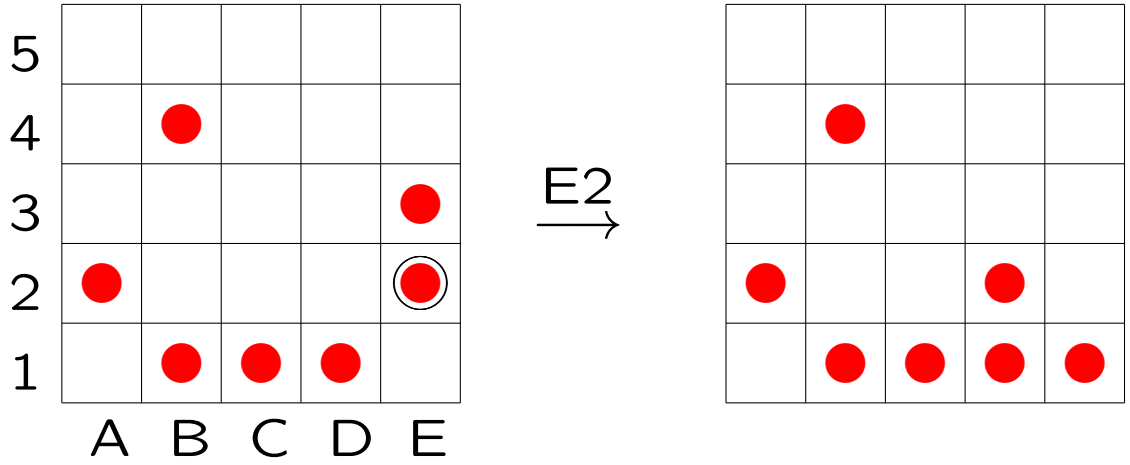
Dit patroon heet glider.

De life-configuratie breeder produceert glider guns:

www.liacs.leidenuniv.nl/~kosterswa/pm/op3pm21.php

3e programmeeropgave 2021

Bij LightsOut moet je alle lampjes uit doen:

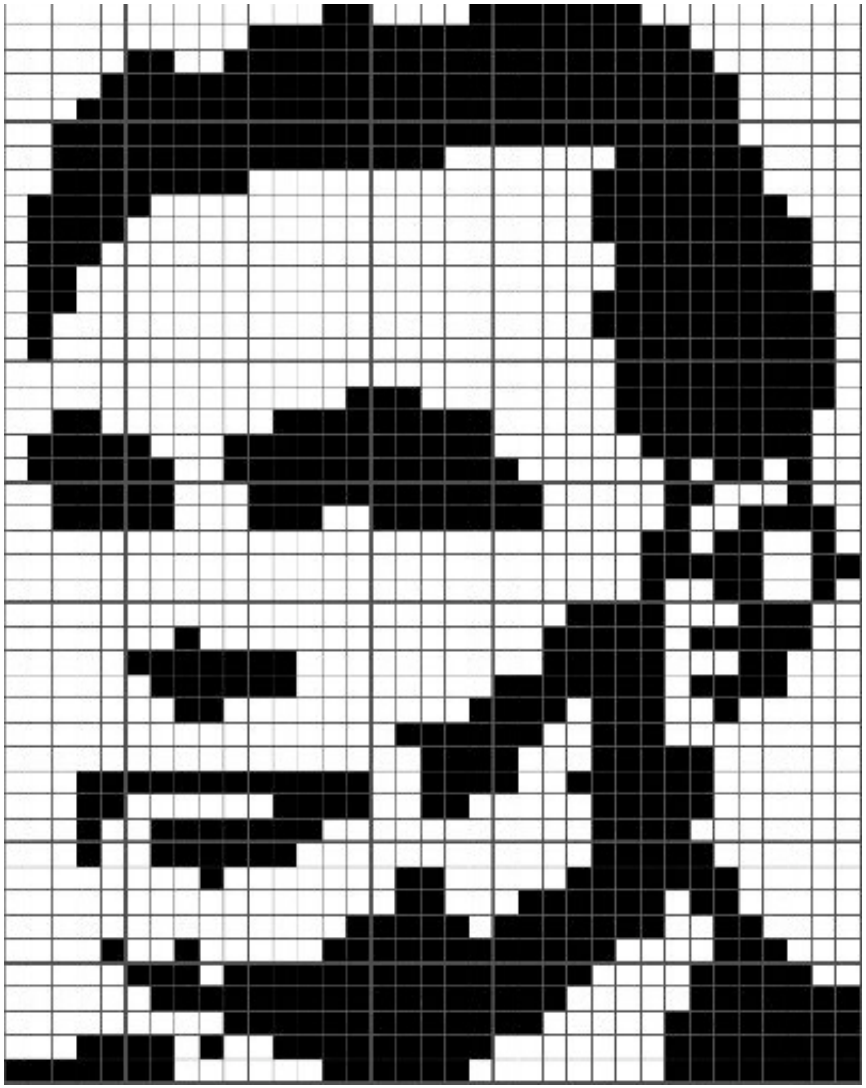


Als je een lamp selecteert, klappen die en de direct aangrenzende horizontale en verticale buren om (aan $\leftrightarrow$ uit).

Voor Life/Nonogram/LightsOut: 2-dimensionale arrays (matrices)!

2022: www.liacs.leidenuniv.nl/~kosterswa/pm/op3pm.php

Nonogrammen



Als je Japanse puzzels zegt, denkt iedereen aan Sudoku.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

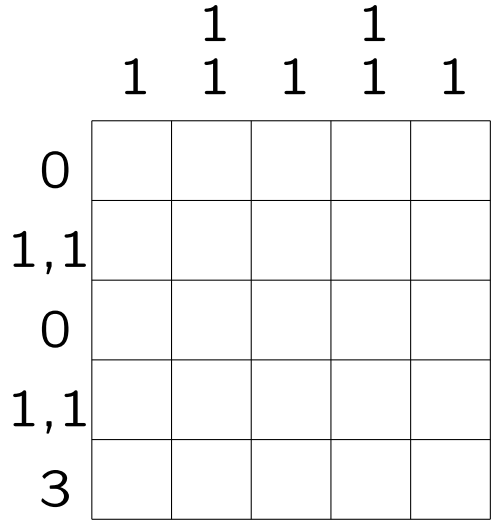
Als je Japanse puzzels zegt, denkt iedereen aan Sudoku.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

bron: Wikipedia

Maar wij gaan het hebben over Nonogrammen.

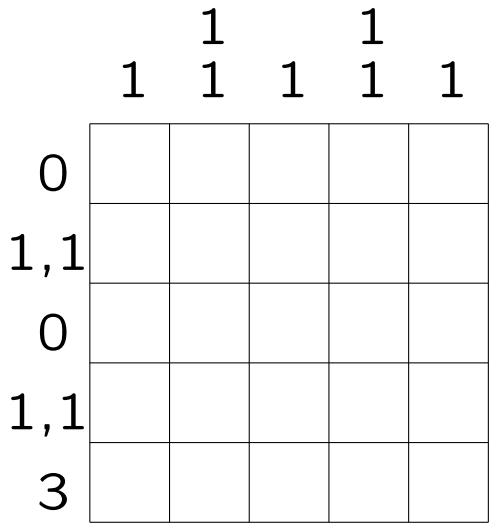
Een Nonogram is een puzzel; een klein voorbeeld:

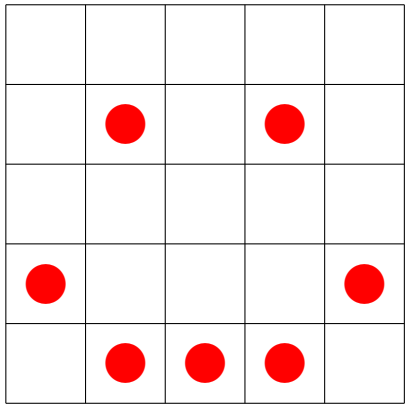


Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode (of zwarte) vakjes.

Waar moeten die rode vakjes komen?

De oplossing ziet er zo uit:





Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode (of zwarte) vakjes.

www.liacs.leidenuniv.nl/~kosterswa/pm/op3pm20.php

3e programmeeropgave 2020

Hoe los je Nonogrammen op?

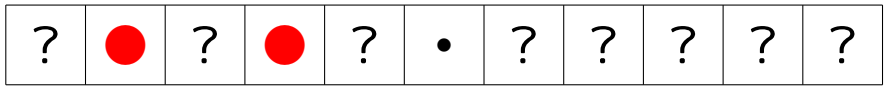
De meeste mensen gebruiken logische regels, en heuristieken = vuistregels zoals "redeneer eerst een keer via de rijen, en dan via de kolommen".

Een voorbeeld van een logische regel is: "als het getal 3 naast een rij van breedte 5 staat, moet het middelste vakje wel rood zijn". Je kijkt dan eigenlijk naar één rij of kolom.

Een heuristiek uit het dagelijks leven: als je een bedrag (zeg 80 cent) moet betalen, kun je het beste de grootste kleinere munt (50 cent?) die je hebt eerst geven.

Stel dat je van een rij al weet:

3,2,1

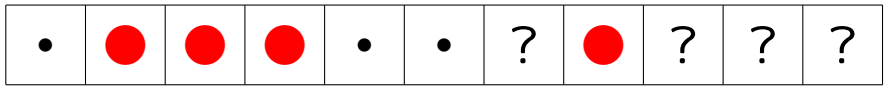


Een • betekent een zeker leeg vakje, een ● staat voor een zeker gevuld vakje. De rest is nog onbekend.

Wat kun je hier nu concluderen?

We concluderen dan dat voor deze rij geldt:

3,2,1



Een • betekent een zeker leeg vakje, een ● staat voor een zeker gevuld vakje. De rest blijft nog onbekend.

Dus door naar een enkele rij of kolom te kijken kun je vooruitgang boeken. En dat gaat goed met dynamisch programmeren.

Hoe ver komen we als je alleen per rij/kolom kijkt? Een • betekent weer een zeker leeg vakje, een ● staat voor een zeker gevuld vakje.

	1	1 1	1	1 1	1
0					
1,1					
0					
1,1					
3					

•	•	•	•	•
•	●	•	●	•
•	•	•	•	•
?	?	•	?	?
?	?	●	?	?

Maar nu zitten we vast . . . tenzij we rijen en kolommen *samen* bekijken.

Dit hadden we:

		1 1		1 1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					

•	•	•	•	•
•	●	•	●	•
•	•	•	•	•
v	w	•	?	?
u	x	●	?	?

Stel dat $u =$ ●, dan (kolom) moet v leeg zijn, en dus (rij) $w =$ ●, en dus (kolom) moet x leeg zijn. Tegenspraak (rij)! Dus u moet leeg zijn.

Dat was een lastige logische redenering, ook voor een computer. Maar de rest is nu eenvoudig.

Soms heeft een probleem meer mogelijke (goede en foute) oplossingen dan je kunt doorrekenen. Zo heeft een 5×5 Nonogram al

$$2^{25} = 2^{10} \cdot 2^{10} \cdot 2^5 = 1024 \cdot 1024 \cdot 32 \approx 32 \text{ miljoen}$$

mogelijke invullingen! Er zijn immers $5 \times 5 = 25$ vakjes die elk 2 mogelijkheden hebben.

De "45×35 Turing" heeft $2^{1575} \approx 10^{700}$ mogelijkheden.

Dus brute-force, alles domweg proberen, lost een complete puzzel niet *snel* op . . .

Het oplossen van een Nonogram is een NP-volledig probleem, net als het maken van een schoolrooster. Dat kan eigenlijk alleen goed met brute-force, maar dan duurt het veel te lang. Van een mogelijke oplossing kun je wel snel zien of hij goed of fout is.

Het grootste open probleem in de informatica $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ gaat hierover. Je kunt een miljoen dollar verdienen als je dit oplost: de Clay Prize.
Het gaat daar niet om het oplossen van één puzzel of maken van één schoolrooster, maar om het vinden van een efficiënte algemene methode — als die al bestaat.

Hoe maak = construeer = ontwerp je zelf een Nonogram?

kleurenfoto

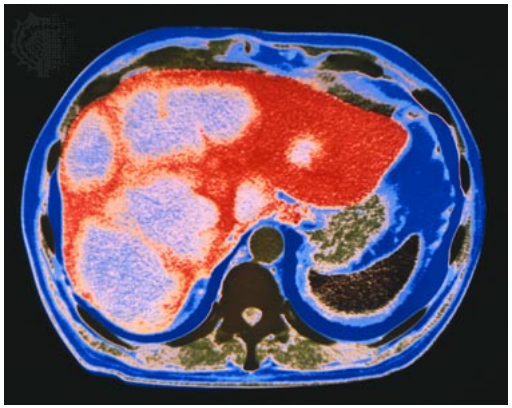
grijswaarden-plaatje

puzzel

Waarom doen wetenschappers Nonogrammen?

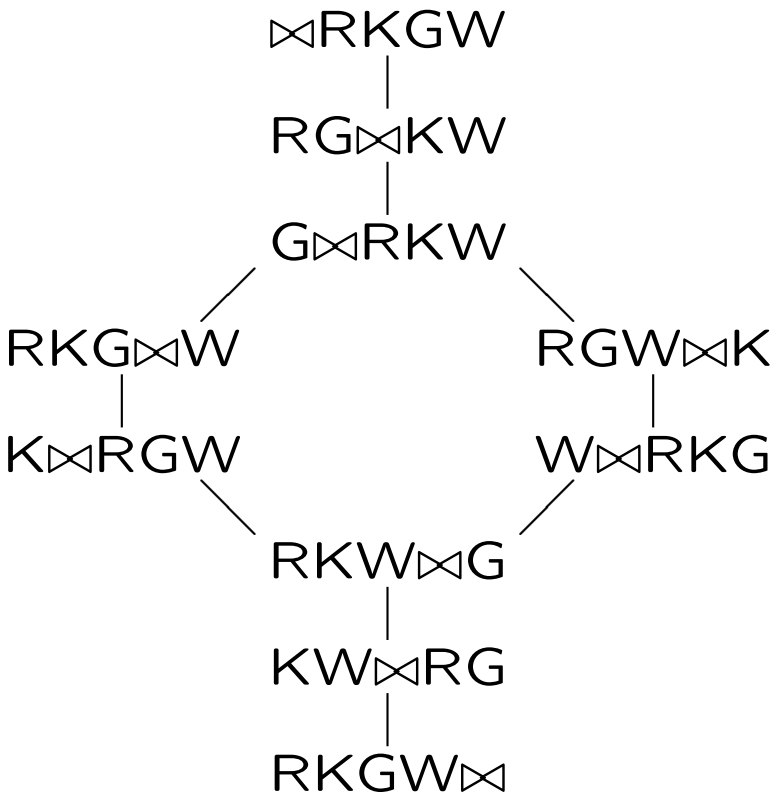
Tomografie houdt zich bezig met het volgende probleem:
Hoe reconstrueer je een object uit projecties?

Voorbeelden:

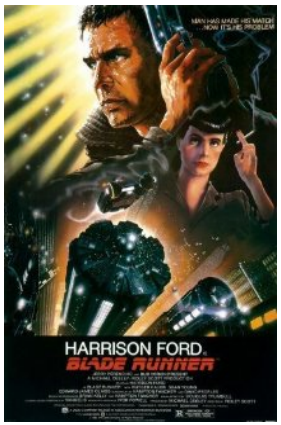


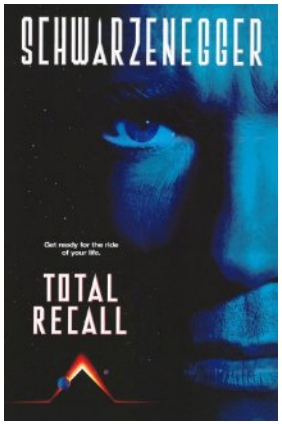
- Nonogrammen oplossen
- Hoe zien onze organen eruit, gegeven CT-scans?
- Waar zitten de "gaten" in een diamant?

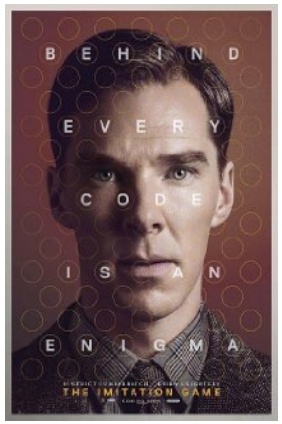
Bij het vervolgcollege Algoritmiek komen "algoritmen" (oplossingsmethoden) aan de orde, bijvoorbeeld voor het oplossen van het probleem van de **K**ool, de **G**eit en de **W**olf (en Kapitein **R**ob's **R**oeiboot):

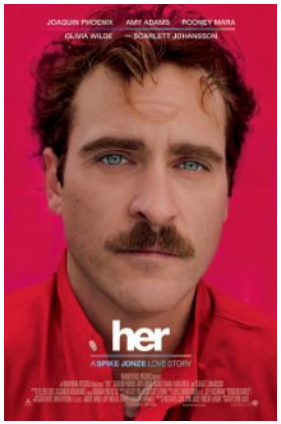


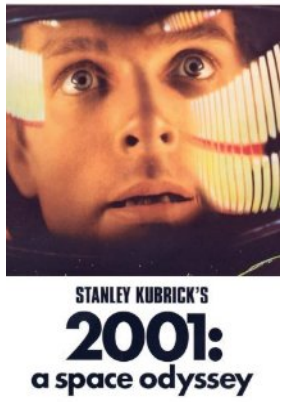
Kunstmatige intelligentie $\stackrel{?}{=}$ algoritmen











Kunstmatige intelligentie (AI, Artificial Intelligence) is een verzamelnaam voor een breed vakgebied, met vragen als:

- *robotica*: Hoe programmeer je een robot?
- *data mining*: Wat is verborgen in/op WikiLeaks?
- *rechtspraak*: Kan een machine rechter zijn?
- *vertalen*: "the spirit is willing but the flesh is weak" $\rightarrow$ ... $\rightarrow$ "the vodka is good but the meat is rotten"?
- *computer games*: Kan een computer Fortnite spelen?
- *neurale netwerken*: Kun je beurskoersen voorspellen?
- *cognitie*: Kunnen we het menselijk brein nadoen?

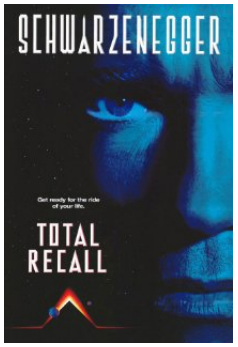
NLP ↙

Je kunt op minstens twee manieren naar Kunstmatige intelligentie kijken:

1. vanuit een meer *psychologische* of *filosofische* richting:
   Wat is het verschil tussen een mens en een computer?
   Kan een computer denken?

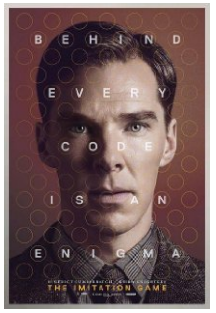
2. vanuit een meer *technische* richting:
   Hoe werkt een schaakprogramma?
   Hoe werkt een Marsrobot?

"Do androids dream
of electric sheep?" $\longrightarrow$

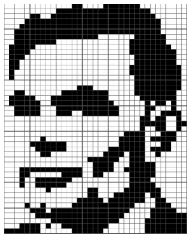


Kunstmatige intelligentie laat computers zich zo gedragen dat het intelligent zou heten als mensen het op die manier zouden doen.

De beroemde Turing-test uit 1950 vraagt (“the imitation game”):

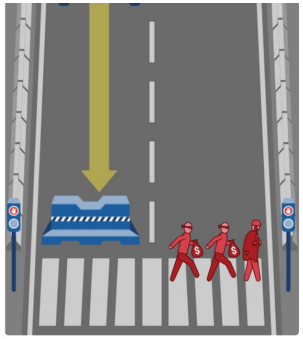


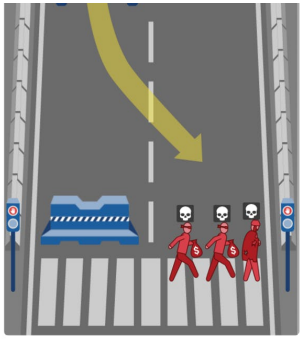
> In een afgesloten kamer bevindt zich een mens *of* een computer, waarmee we alleen via toetsenbord en beeldscherm contact hebben.
> Is het een mens of juist een computer?



Het originele probleem was overigens met man $\leftrightarrow$ vrouw.

Momenteel doen zelfrijdende systemen het erg goed.





↑computer-zien, kennis, de wet, ethiek↑, . . .

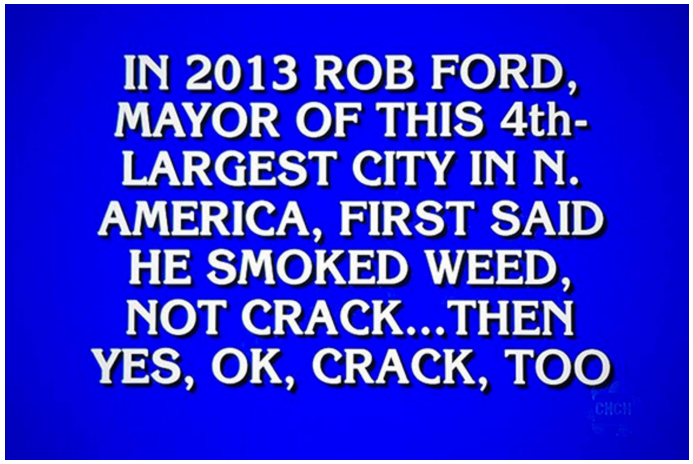
Udacity en Coursera hebben MOOC's over zelfrijdende auto's.

Deep Blue vs. Garry Kasparov, 1997

AlphaZero 2018

Marion Tinsley (1927–1995) was de beste menselijke checkers-speler (dammen op een schaakbord) ooit.
In 2007 werd door Jonathan Schaeffer bewezen dat de beginspeler altijd minstens remise kan halen.

IBM heeft in 2011 een computer "Jeopardy!" laten spelen:



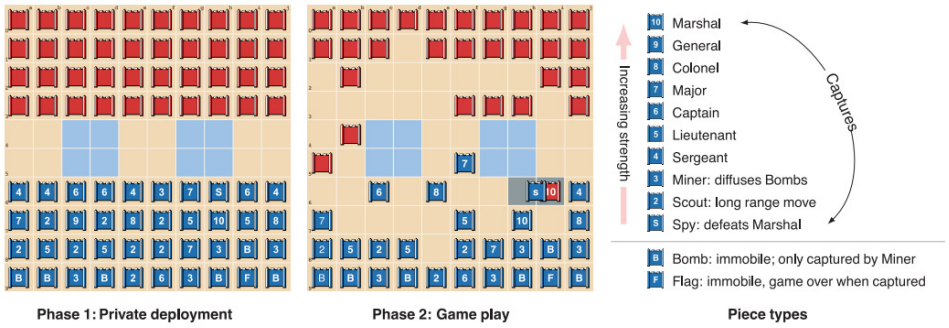
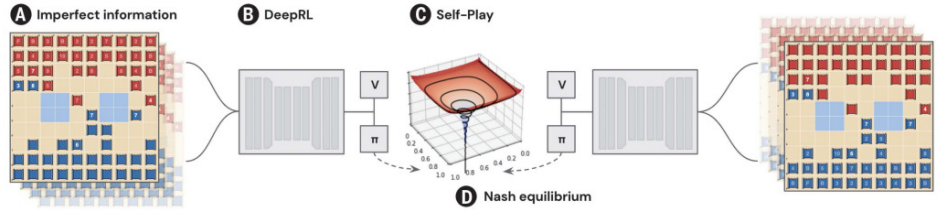


Fig. 1. Stratego is a two-player board game in which players try to capture the opponent's flag. Initially, the players secretly deploy 40 pieces of diverse strengths on the board. Then, they take turns moving pieces, possibly encountering an opponent piece that reveals both piece identities, and then the weaker piece is removed. Two lakes (indicated in blue) cannot be crossed by any piece. The complete rules are defined by the International Stratego Federation.



Replicator dynamics: $\frac{d}{dt}\pi_\tau^i(a^i) = \pi_\tau^i(a^i)\left[Q_{\pi_\tau}^i(a^i) - \sum_{b^i}\pi_\tau^i(b^i)Q_{\pi_\tau}^i(b^i)\right]$

Reward transformation: $r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta\log\left(\frac{\pi^i(a^i)}{\pi_{\text{reg}}^i(a^i)}\right) + \eta\log\left(\frac{\pi^{-i}(a^{-i})}{\pi_{\text{reg}}^{-i}(a^{-i})}\right)$

breaking news

↓

RESEARCH

MACHINE LEARNING

Mastering the game of Stratego with model-free multiagent reinforcement learning

Julien Perolat*†, Bart De Vylder*†, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer‡, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, Karl Tuyls*†

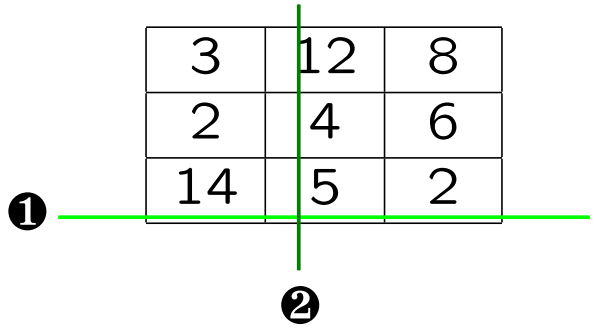
We introduce DeepNash, an autonomous agent that plays the imperfect information game Stratego at a human expert level. Stratego is one of the few iconic board games that artificial intelligence (AI) has not yet mastered. It is a game characterized by a twin challenge: It requires long-term strategic thinking as in chess, but it also requires dealing with imperfect information as in poker. The technique underpinning DeepNash uses a game-theoretic, model-free deep reinforcement learning method, without search, that learns to master Stratego through self-play from scratch. DeepNash beat existing state-of-the-art AI methods in Stratego and achieved a year-to-date (2022) and all-time top-three ranking on the Gravon games platform, competing with human expert players.

Perolat et al., Science 378 (2022) 990–996: Mastering the game of Stratego . . .

Deep neural nets, reinforcement learning, selfplay.

Kunnen computers denken? Diplomacy! link

Maxi en Mini spelen het volgende eenvoudige spel: Maxi wijst eerst een (horizontale) rij aan, en daarna kiest Mini een (verticale) kolom:



Bijvoorbeeld: Maxi ❶ kiest rij 3, daarna kiest Mini ❷ kolom 2; dat levert einduitslag 5.

Maxi wil graag een zo groot mogelijk getal, Mini juist een zo klein mogelijk getal.

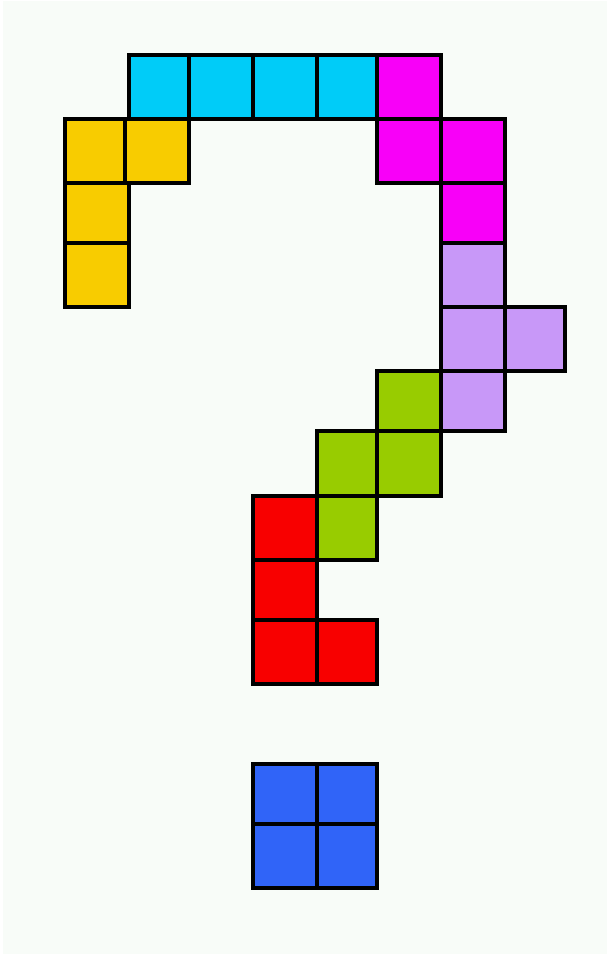
Hoe spelen we dit spel zo goed mogelijk?

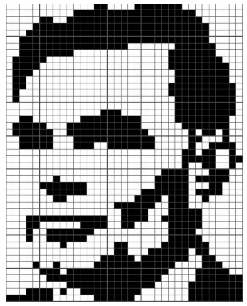
Als Maxi rij 1 kiest, kiest Mini kolom 1 (levert 3); als Maxi rij 2 kiest, kiest Mini kolom 1 (levert 2); als Maxi rij 3 kiest, kiest Mini kolom 3 (levert 2). Dus kiest Maxi rij 1!

3	12	8
2	**?**	**?**
14	5	2

Nu merken we op dat de analyse (het minimax-algoritme) hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit. Het α-β-algoritme onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden.

Vragen?





Tetris

YouTube

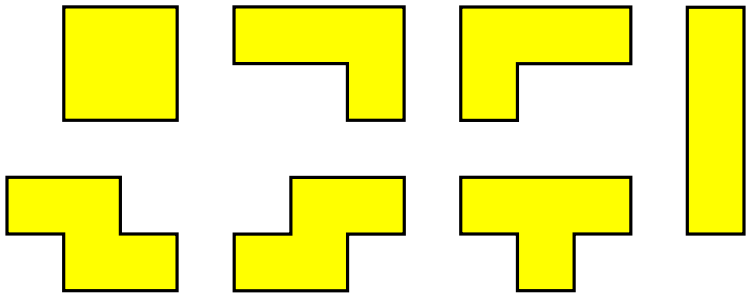
Ook aan een spel als Tetris kleven allerlei vragen:

- Hoe speel je het zo goed mogelijk? (AI)
- Hoe moeilijk is het? (complexiteit)
- Wat kan er allemaal gebeuren?

Zo is bijvoorbeeld bewezen dat sommige Tetris-problemen NP-volledig zijn, dat je bijna alle configuraties kunt bereiken, maar dat niet alle problemen "beslisbaar" zijn:

www.liacs.leidenuniv.nl/~kosterswa/tetris/

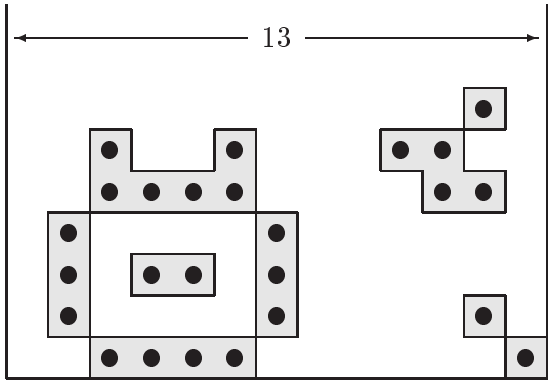
De 7 Tetris-stukken:



Stukken vallen random; volle regels worden verwijderd.
De vraag "Kun je met een gegeven serie (inclusief volgorde) van deze stukken een bord helemaal leeg spelen?" is NP-volledig. (Ander voorbeeld: kun je een gegeven postzegelwaarde plakken?)
Als iemand het bord leeg speelt kun je dat eenvoudig controleren. Als het *niet* kan, kan men (tot nu toe) niks beters verzinnen dan alle mogelijkheden één voor één na te gaan!

Alles bereikbaar?

Een "willekeurige" configuratie:



Deze kan gemaakt worden door 276 *geschikte* Tetris-stukken op de juiste plaats te laten vallen.
Let op: alleen geheel gevulde regels verdwijnen, alles daarboven zakt *één rij*.

Claim: op een bord van oneven breedte kan elke configuratie bereikt worden!