Informatica door de jaren heen

(en Tetris)

dr. Walter Kosters, Universiteit Leiden

Ouderdag, Leiden, zaterdag 21 april 2012

`www.liacs.nl/home/kosters/`

eerste, tweede en derde jaar

van onderwijs naar onderzoek: Nonogrammen, Tetris, . . .

vroeger, nu en later

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

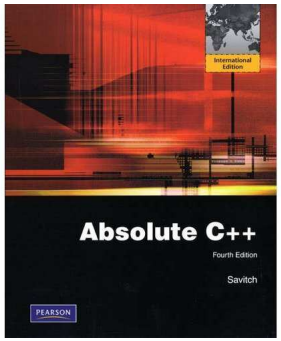
najaar	voorjaar
Programmeermethoden	Algoritmiek
Fundamentele informatica 1	Logica
Digitale technieken	Databases
Continue wiskunde	Lineaire algebra
Studievaardigheden	Challenges (app!)

Tweede en derde jaar zijn als volgt:

najaar	voorjaar
Datastructuren	Kunstmatige intelligentie
Fundamentele informatica 2	Complexiteit
Computerarchitectuur	Operating systemen
Programmeertalen	Correctheid
Requirements engineering	Software engineering
Compilerconstructie	Netwerken
Fundamentele informatica 3	Theorie van concurrency
Human computer interaction	Bachelorproject
Data mining	
Keuzevak 1	Keuzevak 2

Programmeermethoden

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



In Leiden leren alle eerstejaars studenten Informatica, Wiskunde, Natuurkunde en Sterrenkunde de taal C++. 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.

Een tweede C++-programma:

```
// dit is een simpel programma
#include <iostream>
using namespace std;
int main ( ) {
  int getal = 42;  // een variabele declareren
                   // en initialiseren
  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!

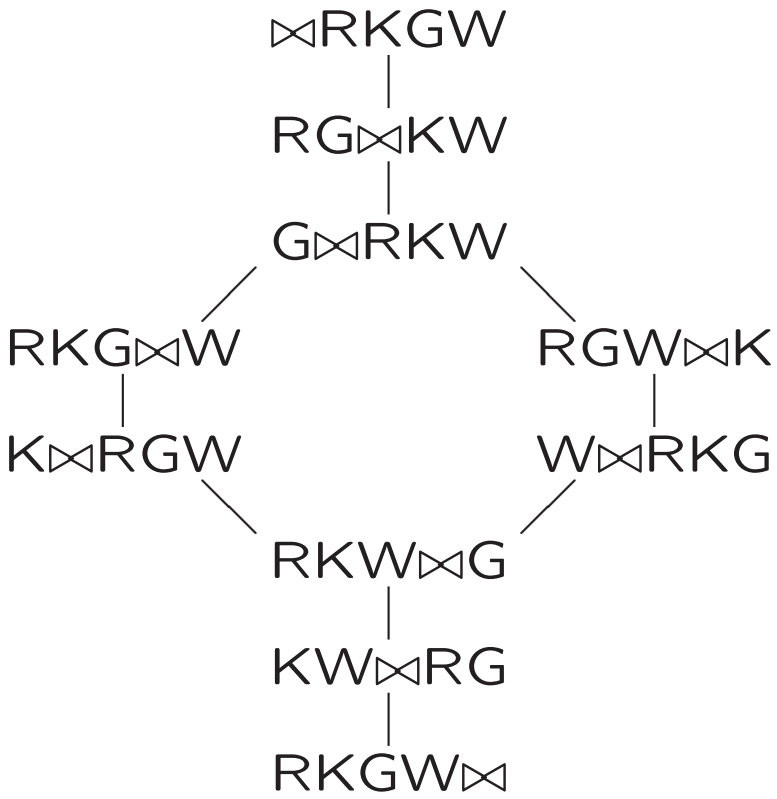
. . . **I work 9–5 in a 7–11**

Allereerst moet de bezoeker zijn/haar geboortejaar als geheel getal invoeren, en daarna de geboortemaand, ook als geheel getal. Vervolgens voert hij/zij de geboortedag in, wederom als geheel getal. Het programma berekent dan de leeftijd van de gebruiker, zowel in aantal jaren als in maanden (bijvoorbeeld: 10 jaar en 3 maanden; 123 maanden); beide worden op het beeldscherm getoond. De leeftijd in maanden wordt analoog aan die in jaren bepaald (als je op de 31ste geboren bent, wordt je iedere maand een maand ouder, maar je bent niet zo vaak maandig).

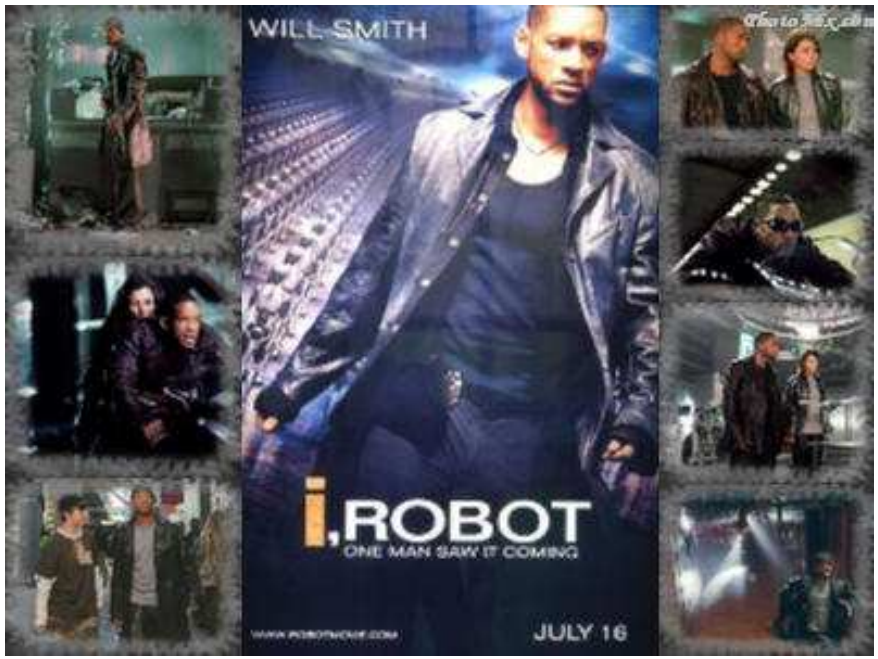
Nu moet de bezoeker zijn geboort edag (zondag, maandag, . . . , zaterdag) geven. Als deze goed is, gaat het programma verder, en anders stopt het.

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

Bij het vervolgcollege Algoritmiek komen "algoritmen" (oplossingsmethoden) aan de orde, bijvoorbeeld voor het oplossen van het probleem van de Kool, de Geit en de Wolf:



Kunstmatige intelligentie



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

- *robotica*: Hoe programmeer je een robot?
- *data mining*: Hoe zoekt Google? WikiLeaks?
- *rechtspraak*: Word je volautomatisch be/veroordeeld?
- *vertalen*: "the spirit is willing but the flesh is weak" $\rightarrow$ ... $\rightarrow$ "the vodka is good but the meat is rotten"?
- *computer games*: Hoe speelt de computer bij CoD?
- *neurale netwerken*: Kun je beurskoersen voorspellen?

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?

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

De beroemde Turing-test (1950) stelt het volgende probleem.
In een afgesloten kamer bevindt zich een mens *of* een computer, waarmee we alleen via een computerverbinding contact kunnen hebben. Kunnen wij bepalen of het een mens is, of juist een computer?
Het originele probleem was overigens met man $\leftrightarrow$ vrouw.

AI in 1964–66 schreef Joseph Weizenbaum van het MIT in Boston het beroemde programma ELIZA/DOCTOR, dat een psychotherapeut "speelt" — met veel succes.
Een voorbeeldconversatie (Jason Hutchens' HAL/HeX):

my name is bob
`Pleased to meet you.`
how are you doing, jas?
`I am just fine, thanks! And you?`
i am feeling great!
`If you are feeling great, it's your business, mate!`

Dit soort computerprogramma's doet mee in de jaarlijkse Loebner-contest.

Een robot is een "actieve, kunstmatige agent wiens omgeving de fysieke wereld" is. Het woord stamt uit 1921 (of eerder), en is gemaakt door de Tsjechische broers Capek. En softbots: RoboCom, internet programma's.

Van de science fiction schrijver Isaac Asimov (auteur van "I, Robot") zijn de drie wetten van de robotica:

1. Een robot mag een mens geen kwaad doen.

2. Een robot moet menselijke orders gehoorzamen (tenzij dat tegen 1. ingaat).

3. Een robot moet zichzelf beschermen (tenzij dat tegen 1. of 2. ingaat).

Mobiele robots

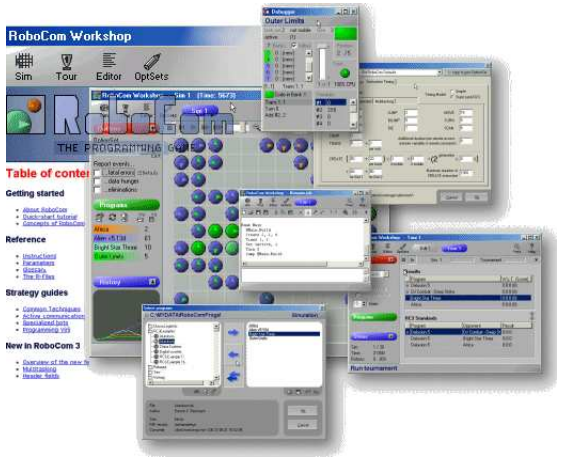
NASA Sojourner

Sony Aibo

Hiroshi Ishiguro's robot (en hijzelf)

Honda

In het programma RoboCom
vechten robot-programma's
met elkaar.




```
; een klein robot-programma
NAME Flooder/Shielder

BANK Een          ; eerste en enige bank
  @Loop           ; label
  TURN 0          ; draai linksom
  SCAN #5         ; scan reference field
  COMP #5,0       ; leeg=0? vijand=1, vriend=2
  JUMP @Loop      ; nee, verder draaien
  CREATE 2,1,0    ; ja; creeer nieuwe robot
  TRANS 1,1       ; en kopieer jezelf
  SET %Active,1   ; activeer hem/haar
  ; auto-reboot
```

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

3	12	8
2	4	6
14	5	2

Bijvoorbeeld: Maxi kiest rij 3, daarna kiest Mini 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

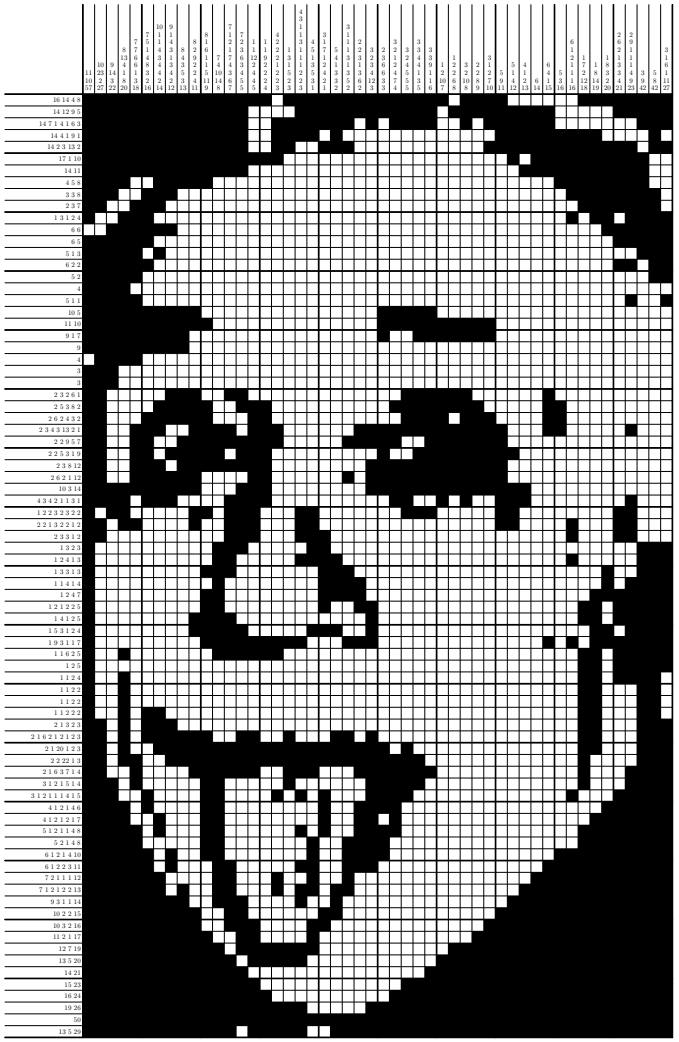
Nu merken we op dat de analyse hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit. Het α-β-algoritme onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden.

Ieder schaakprogramma gebruikt deze methode.

Schaken

IBM heeft in 2011 een computer “Jeopardy!” laten spelen:

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

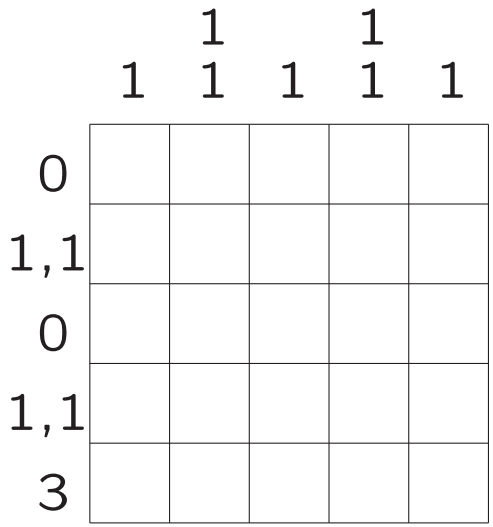
Als je Japanse puzzels zegt, denkt iedereen aan Sudoku.

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:

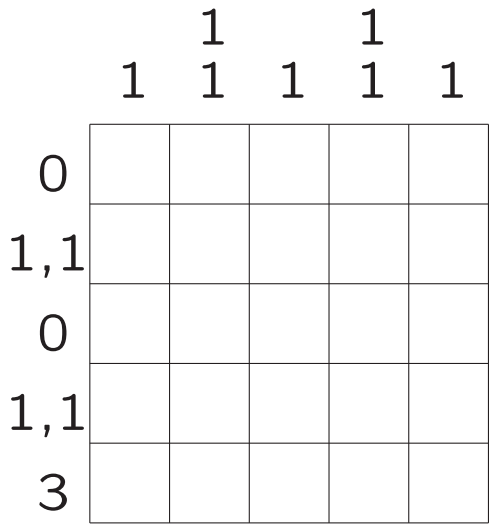


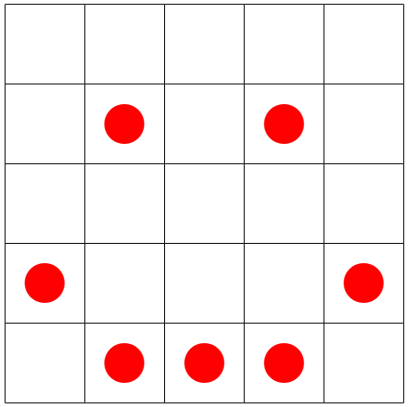
Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode (of zwarte) vakjes.

Waar moeten die rode vakjes komen?

Voorbeeld — oplossing

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.

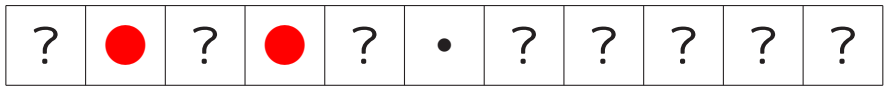
Hoe los je Nonogrammen op?

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.

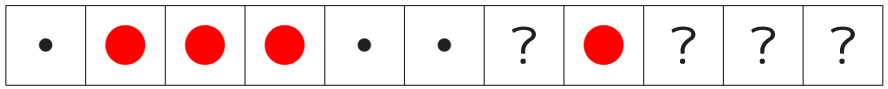
Stel dat je van een rij al weet:

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.

Voorbeeld — oplossen

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					

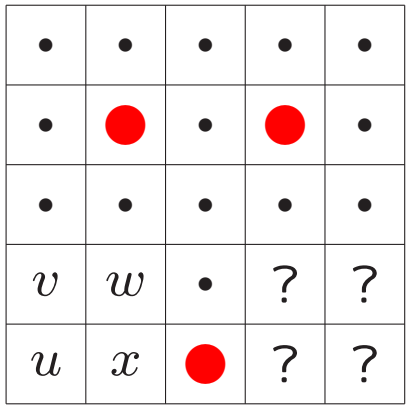
•	•	•	•	•
•	●	•	●	•
•	•	•	•	•
?	?	•	?	?
?	?	●	?	?

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

Voorbeeld — verder oplossen

Dit hadden we:

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



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.

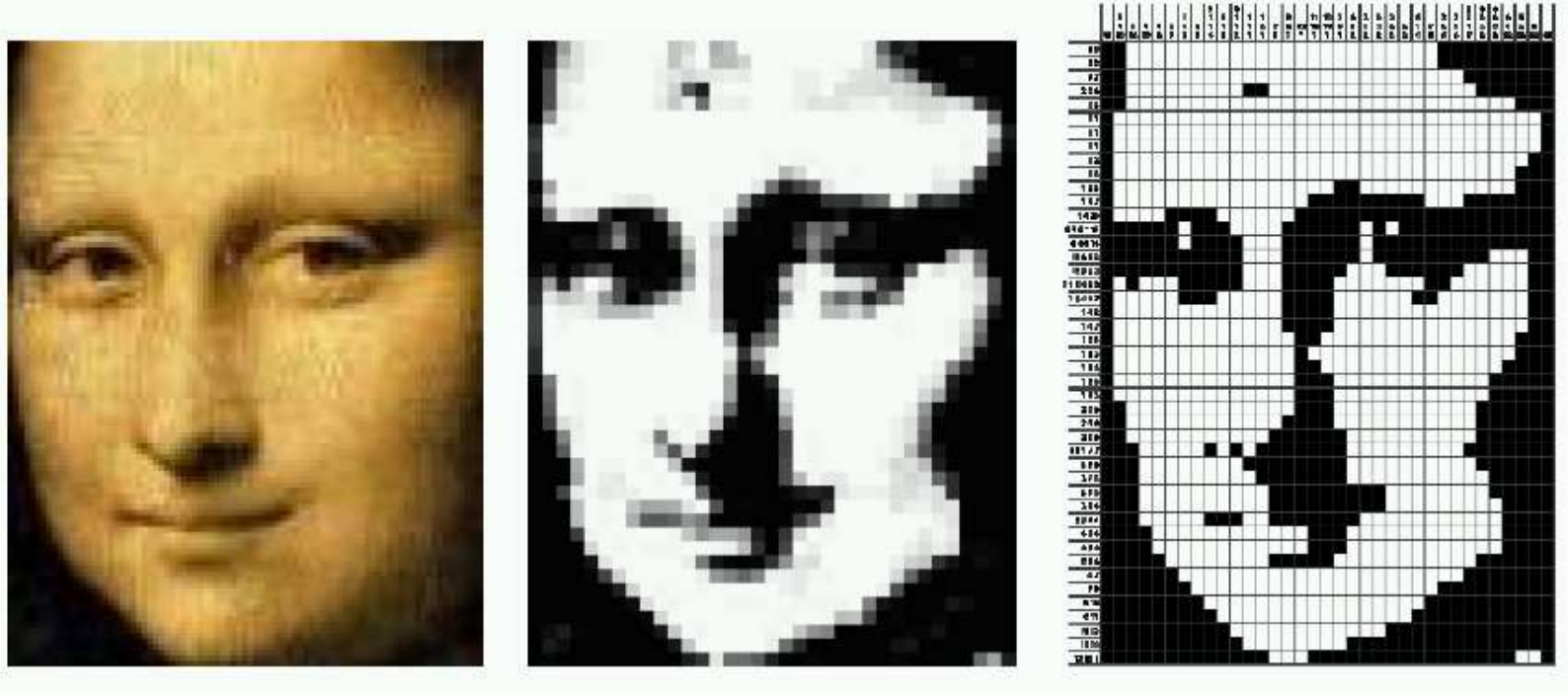
De "80×50 Einstein" heeft $2^{4000} \approx 10^{1200}$ mogelijkheden.

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?

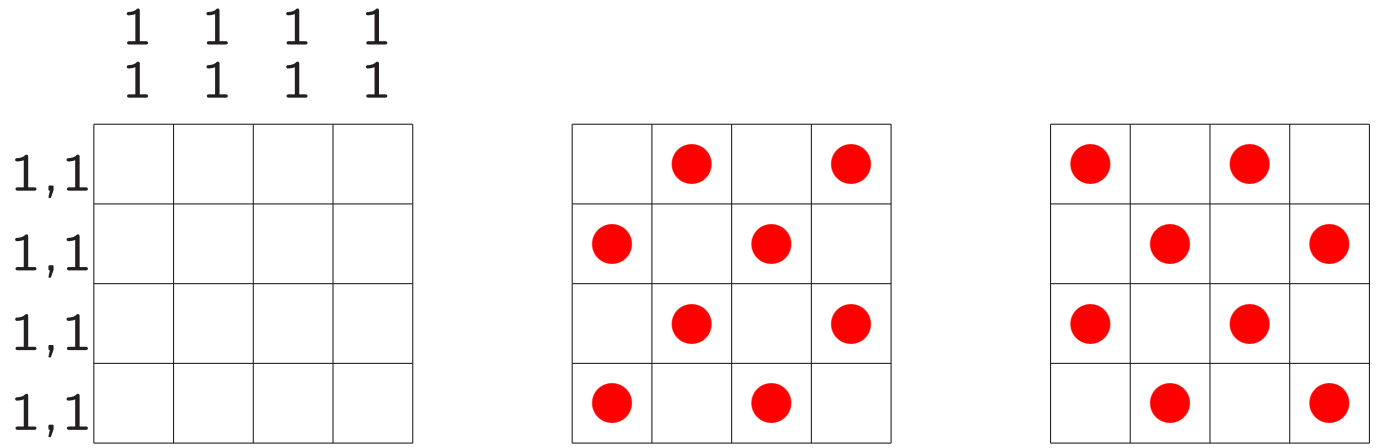


kleurenfoto grijswaarden-plaatje puzzel

`http://www.liacs.nl/home/kosters/nono/`

Denk eraan dat een *goed* Nonogram een unieke oplossing moet hebben.

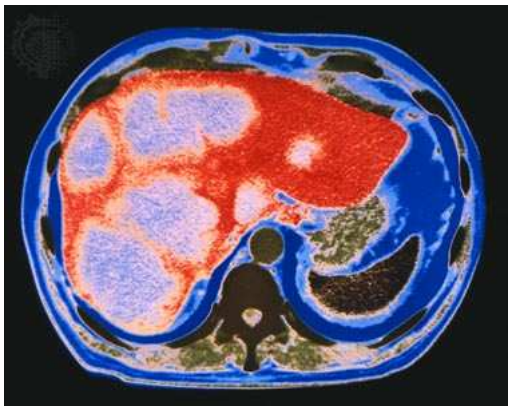
Ze hebben in het algemeen namelijk soms heel verschillende oplossingen:



Waarom doen wetenschappers Nonogrammen?

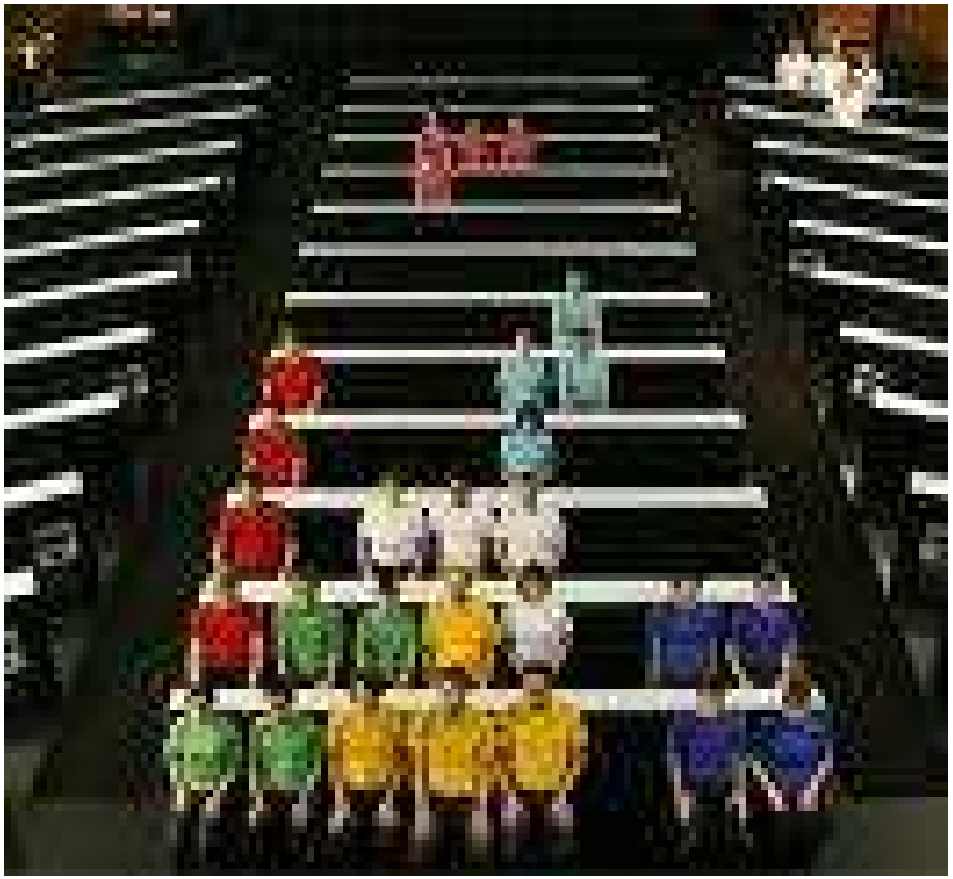
Tomografie houdt zich bezig met het volgende probleem:
Hoe reconstrueer je een object uit projecties?

Voorbeelden:



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

Tetris



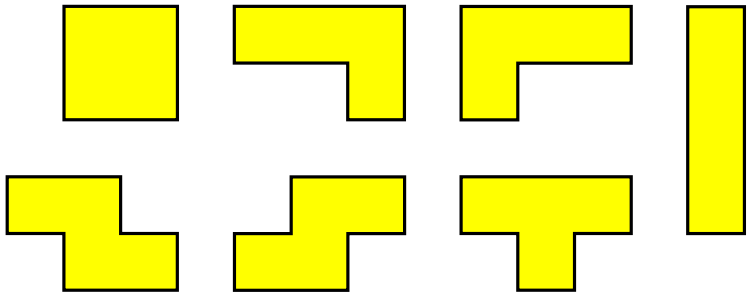
Ook aan een spel als Tetris kleven allerlei vragen:

- 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, zie:

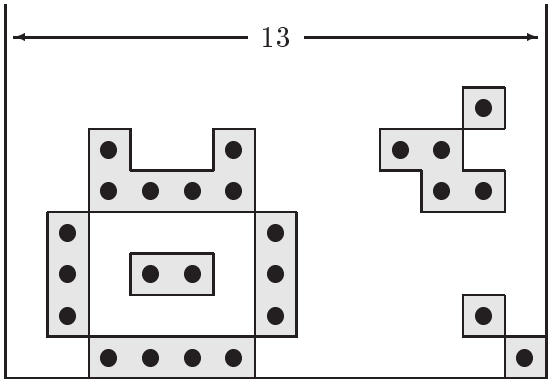
`www.liacs.nl/home/kosters/tetris/`

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!

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!

Vragen?