Een miljoen dollar verdienen in de kerstvakantie? Het enige dat u hoeft te doen, is een polynomiaal algoritme te vinden om een sudoku mee op te lossen. **Niels Oosterling** schetst waar u dan rekening mee moet houden.

Puzzels en wiskunde

Inleiding

Op 10 en 11 maart dit jaar werden in het Italiaanse Lucca de wereldkampioenschappen sudoku-oplossen gehouden. De eenendertigjarige Tsjechische accountant Jana Tylova hield met 56 punten de zesentwintigjarige Harvardstudent Thomas Snyder ver achter zich. In een koortsachtige eindstrijd waarin tien finalisten volgens een knock-outsysteem streden om de wereldtitel, kwam Snyder niet verder dan 17 punten, nog net genoeg voor het zilver. Betere sudoku-oplossers dan deze puzzelaars brengt de mensheid momenteel niet voort.

Net als het Rubikkubusdraaien zal de populariteit van het oplossen van sudoku's in wedstrijdverband binnen enkele jaren alweer zijn verdwenen. Een opgeloste sudoku of rechtgedraaide Rubikkubus zijn dan ook niet echt resultaten die je trots aan je vrienden toont. Evenmin zul je na een wedstrijd in de kroeg opscheppen over hoe je vakje (6,3) nou zo snel gevonden had. Toch zou je er een hoop vrienden bij krijgen als je echt een handige manier hebt gevonden om een sudoku op te lossen. Het Clay Institute for Mathematics keert een miljoen dollar uit aan degene die een 'slimme' strategie vindt voor het oplossen van sudoku's en elke zichzelf respecterende universiteit zal in de rij staan om jou met je slimme methode binnen te halen. Hoe slim moet 'slim' zijn?

Die 'slimme' strategie om sudoku's op te lossen is dus nog niet bekend. Dit artikel gaat verder in op een voorbeeld van een ander puzzeltje waarvoor wel een 'slim' algoritme bestaat. Dat is wiskundig gezien een stuk leuker, maar van het puzzelen blijft dan niet zoveel meer over. Eerst maar eens een korte uitleg over algoritmes en de snelheid van een algoritme.

Sudoku

Bij het aanvullen van een sudoku is het de bedoeling de lege vakjes in te vullen met een getal van 1 tot en met 9. Elk getal mag slechts één keer in elke rij, kolom en met dikkere randen aangegeven 3 × 3-vierkant voorkomen.

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

Algoritme

Jana's strategie, of algoritme, die haar de oplossing gaf, was waarschijnlijk niet wezenlijk sneller dan die van haar concurrenten. Hieronder zullen we de begrippen algoritme en de snelheid van een algoritme iets formeler maken.

Een algoritme is een eindige reeks instructies om vanuit een gegeven begintoestand het daarbij horende doel te bereiken. Een gegeven begintoestand van het algoritme kan worden voorgesteld door een rij getallen. De reeks instructies kan worden uitgesplitst in elementaire stappen. Voorbeelden van elementaire stappen zijn het bij elkaar optellen, met elkaar vermenigvuldigen of vergelijken van twee getallen.

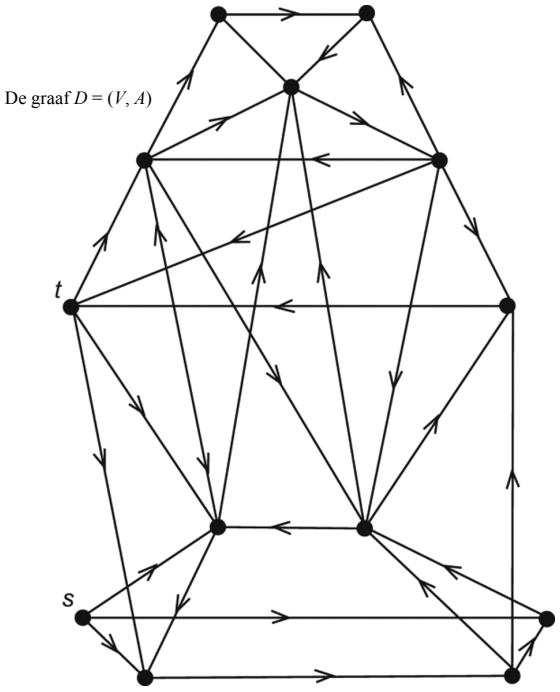
Binnen de wiskunde wordt de snelheid van een algoritme uitgedrukt in het aantal elementaire stappen dat moet worden gezet om tot een oplossing te komen. De aanname is dat al deze elementaire stappen op een denkbeeldige computer een bepaalde tijdseenheid kosten. Voor veel problemen is het aantal elementaire stappen dat nodig is om de oplossing te bereiken, afhankelijk van de grootte van de begintoestand. De totale tijd dat een algoritme nodig heeft kan dan worden uitgedrukt in de grootte van de

input voor het algoritme. Stel bijvoorbeeld dat we van een verzameling $\{x_1, x_2, ..., x_n\}$ het grootste getal willen bepalen. De input voor een algoritme dat het grootste getal bepaalt, is die verzameling getallen. De getallen moeten op een slimme manier met elkaar vergeleken worden. In semi-programmeertaal zou je een algoritme kunnen schrijven als:
$\max(0) = x_1$
Herhaal onderstaande regel totdat $i = n + 1$. Begin met $i = 1$.
Als $\max(i-1)$, dan $\max(i) = x_i$. Anders $\max(i) = \max(i-1)$.
$i = i + 1$.

Het grootste getal van de verzameling $\{x_1, x_2, ..., x_n\}$ is nu $\max(n)$. Het is duidelijk dat het aantal elementaire stappen dat gezet moet worden groter wordt naarmate de verzameling met getallen groter wordt.

Voorbeeld

Met onderstaand voorbeeld proberen we inzichtelijker te maken wat wordt bedoeld met een algoritme, elementaire stappen en de snelheid van een algoritme. Het probleem dat in dit voorbeeld wordt opgelost is: Wat is de afstand tussen twee gegeven punten in een gegeven gerichte graaf. Grafen zijn veelgebruikte representaties van problemen in de discrete wiskunde. Het oplossen van sudoku's is ook een voorbeeld van een discreet probleem.



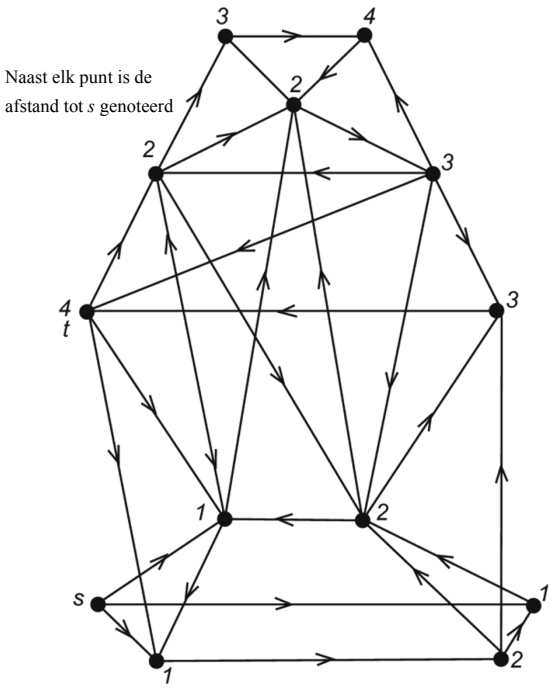
Laat $D = (V, A)$ een gerichte graaf zijn. V is een eindige verzameling punten en A een verzameling geordende paren uit V. Deze geordende paren kunnen gemakkelijk worden voorgesteld als pijlen die van het eerste van het paar punten naar het tweede punt lopen. We kiezen twee (willekeurige) punten s en t uit V. We introduceren een paar begrippen:

- Een wandeling van s naar t is een rij $(s, a_1, v_1, ..., v_{n-1}, a_n, t)$ waarbij $a_i = (v_{i-1}, v_i)$.
- Als alle punten in de wandeling verschillend zijn, noemen we een wandeling ook wel een pad.
- De lengte van een pad is het aantal punten dat in een pad voorkomt.
- De lengte van het kortste pad van het punt s naar t in een graaf noemen we de afstand tussen s en t .

We willen graag de afstand tussen s en t weten. Met andere woorden: we zoeken een minimum aantal punten en paren die een s-t pad vormen. Hiervoor bestaat een gemakkelijk algoritme. In dit algoritme bepalen we zelfs voor elk punt de afstand tot s.

We verdelen alle punten over de verzamelingen V_i. De verzamelingen V_i worden als volgt gedefinieerd:
$V_0 = \{s\}$ en
$V_i = \{v \in V \setminus (V_0 \cup \ldots \cup V_{i-1}) \mid \exists u \in V_{i-1} : (u, v) \in A\}$.

In de tekening is de graaf D gerepresenteerd als een puntenverzameling met tussen elk tweetal punten u en v een pijl van u naar v als $(u, v) \in A$. De verzameling V_1 bestaat dan uit de punten v waarvoor geldt dat er een pijl van s naar v wijst, V_2 uit de punten w waarvoor er een pijl vanuit de verzameling V_1 naar w wijst, enzovoort. De verzamelingen V_i kunnen ook gezien worden als de punten die op afstand i van s liggen. In de tekening zijn voor alle punten de afstanden tot s gegeven.



Wanneer voor zekere m geldt dat $V_m = 0$, stopt het algoritme. Het punt t zit nu in een van de verzamelingen V_j. De afstand van s tot t is dan j. In het voorbeeld zit t in V_4.

Het enige dat in dit algoritme gebeurt, is het verdelen van de punten $v \in V$ over de verzamelingen V_i. Startend in s, kan lijn voor lijn worden bepaald op welke afstand de punten liggen. Een bovengrens voor het aantal stappen dat gezet moet worden is dan het aantal lijnen. Voor elke pijl die uit een zekere V_i vertrekt, moet immers worden nagegaan of het punt waarnaar deze pijl wijst al in een van de V_j met $j \leq i$ voorkomt.

We hebben dan de snelheid van het algoritme in termen van de grootte van de input uitgedrukt. De grootte van de input is namelijk het aantal punten $|V|$ van de graaf plus het aantal lijnen $|A|$ van de graaf. Het algoritme gebruikt niet meer dan $|A|$ elementaire stappen.

P, NP en NP-volledig

De snelheid van een algoritme waarmee een bepaald wiskundig vraagstuk kan worden opgelost, is tegelijk ook een maat voor de complexiteit van het probleem. Het aantal elementaire stappen dat in het voorbeeld nodig is om tot een oplossing te komen, kan worden begrensd door een lineaire functie. We noemen dergelijke algoritmes lineair en dat is erg snel. Het vinden van de afstand tussen twee punten in een gerichte graaf wordt dan ook niet als een 'moeilijk' probleem gezien. We zullen in de volgende sectie de complexiteit van het oplossen van een sudoku behandelen. Om dit in een perspectief te plaatsen, leggen we eerst uit wat op dit moment de meest gangbare categorisatie van de complexiteit van wiskundige vraagstukken is.

Vraagstukken waarvoor een lineair algoritme bestaat, behoren tot een grotere groep problemen. Dit zijn de problemen waarvoor de oplossing met een algoritme kan worden gevonden waarvoor het aantal gebruikte elementaire stappen kan worden begrensd door een polynoom in de grootte van de input. Zo'n algoritme heet een polynomiaal algoritme en de problemen die met polynomiale algoritmes kunnen worden opgelost, vormen de klasse P.

Een mogelijk nog grotere klasse problemen is de klasse NP. NP betekent Non-deterministisch Polynomiaal. Voor deze problemen eisen we niet dat er een polynomiaal algoritme bestaat dat de oplossing geeft. We eisen 'slechts' dat er een polynomiaal algoritme bestaat voor het volgende probleem: *Gegeven een probleem en een oplossing, is deze oplossing correct?*
Voor deze NP-vraagstukken geldt dus dat als er een correcte oplossing voor is gegeven, de waarheid hiervan met een polynomiaal algoritme kan worden gecontroleerd.
Als voor een probleem de oplossing kan worden gevonden met een polynomiaal algoritme, dan wordt de correctheid van deze oplossing ook gelijk bepaald met dit algoritme. De klasse P is dus bevat in NP.

Of het omgekeerde ook waar is, geldt als een van de grote onopgeloste vraagstukken binnen de wiskunde. Het staat op de door het Clay Institute for Mathematics opgestelde lijst met millenniumproblemen. Diegene die één van deze millenniumvraagstukken oplost, krijgt een miljoen dollar en eeuwige roem. Wellicht dat het commercieel verstandiger is om de oplossing geheim te houden en in de vorm van alle mogelijke toepassingen te gelde te maken. Praktische vraagstukken die gemodelleerd kunnen worden als een NP-probleem zijn te vinden in alle hoeken van de samenleving. Input van zulke problemen is vaak erg omvangrijk. Voorbeelden van veel toegepaste NP-vraagstukken zijn het handelsreizigersprobleem en lineair programmeren met gehele getallen.

Het handelsreizigersprobleem kan als volgt worden geformuleerd. Als er n steden gegeven zijn die een handelsreiziger moet bezoeken, samen met de afstand tussen ieder paar van deze steden, vind dan de kortste weg die kan worden gebruikt, waarbij iedere stad precies eenmaal wordt bezocht.

Een lineair programmeerprobleem bestaat uit een te maximaliseren lineaire functie $c^{\mathrm{T}}x$ waarbij $c, x \in R^n$, met als restrictie $Ax \leq b$, A een $m \times n$ matrix en $b \in R^m$. Voor dergelijke problemen zijn inmiddels polynomiale algoritmes gevonden. Als de restrictie $x \in Z^n$ wordt toegevoegd, is er geen polynomiaal algoritme meer bekend die het probleem oplost. Algoritmes die deze problemen exact oplossen verschillen vaak maar weinig van het nagaan van alle mogelijke oplossingen en daar de beste uit te kiezen, de zogenaamde *brute-force search*. De hoeveelheid mogelijke oplossingen wordt in veel gevallen exponentieel groter met de grootte van de input. Het aantal elementaire stappen dat gezet moet worden om tot een oplossing te komen dus ook. Een polynoom stijgt op den duur een stuk minder snel dan een exponentiële functie.

Het zoeken naar een antwoord op de vraag of P = NP (of niet) kreeg een extra impuls toen Stephan Arthur Cook in 1971 een paper schreef waarin hij het begrip NP-volledig formaliseerde. Hij toonde aan dat elk probleem in de klasse NP met een polynomiaal algoritme kan worden herleid tot het satisfiability probleem, dat we SAT zullen noemen. In SAT bekijken we een Boolese uitdrukking in een aantal variabelen met de connectieven EN ($\wedge$), OF ($\vee$) en NIET ($\neg$). De vraag is of aan de variabelen de waarden TRUE of FALSE kunnen worden toegekend waarmee de hele uitdrukking waar is.

Laat bijvoorbeeld de volgende Boolese functie f gegeven worden door

$$f(x_1, \ldots, x_4) = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_2) \wedge (x_4 \vee \neg x_2) \wedge (x_1 \vee \neg x_4) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_4)$$

Is het mogelijk om de waarden TRUE of FALSE aan de variabelen x_i toe te kennen waardoor de functie f waar wordt? Er is geen polynomiaal algoritme bekend dat SAT oplost.

Inmiddels zijn er veel meer NP-volledige problemen gevonden. Stel dat er een polynomiaal algoritme bestaat voor een of ander NP-volledig probleem, dat we NPCOMPL zullen noemen. Als zo'n algoritme bestaat, zou dus voor elk NP-probleem in polynomiale tijd een oplossing kunnen worden gevonden. Laten we een willekeurig NP-probleem RANDOMNP nemen. RANDOMNP kan in polynomiale tijd worden teruggebracht tot het NP-volledige probleem NPCOMPL. Als NPCOMPL tot de klasse P behoort, hoort RANDOMNP ook in P. Twee polynomiale algoritmes achter elkaar uitvoeren is namelijk weer een polynomiaal algoritme. Als zo'n algoritme gevonden wordt, is dat dus een bewijs dat P = NP.

Een algoritme dat een NP-volledig probleem in polynomiale tijd oplost, wordt door velen als onbestaanbaar geacht. Toch heeft tot op heden nog niemand dat kunnen bewijzen. Erover meedenken blijkt voor meer mensen weggelegd dan bovenstaande uitleg doet vermoeden.

Sudoku blijkt NP-volledig

P, NP of NP-volledige problemen zijn geen zaken die alleen op de werktafel van een wiskundige voorkomen. Het oplossen van een $n^2 \times n^2$ sudoku is een probleem dat tot de klasse NP behoort. De beginsituatie is een deels ingevulde sudoku en de bedoeling is deze aan te vullen tot een volledige en correcte sudoku. Het is gemakkelijk na te gaan dat het aantal elementaire stappen dat gezet moet worden om een correcte oplossing van een $n^2 \times n^2$ sudoku te controleren kan worden begrensd door een functie van de orde n^6. Men hoeft voor elk vakje alleen maar na te gaan of dit niet overeenkomt met de andere vakjes in dezelfde rij, kolom en $n \times n$ vierkant waarbinnen het ligt.

Het vinden van een oplossing is echter een ander verhaal. Een algoritme om een sudoku op te lossen is bijvoorbeeld alle mogelijke complete sudoku's na te gaan en te vergelijken met de in te vullen sudoku. Dit gaat nog wel voor $n = 2$. Er kunnen dan 288 mogelijke sudoku's geteld worden, maar als n = 3, zijn dit er al 6.670.903.752.021.072.936.960. Voor grotere n wordt het echt niet beter.

Er is meer. In 2003 bewees Takayuki Yato in zijn master thesis dat het oplossen van een sudoku zelfs een NP-volledig probleem is. Hij kon het oplossen van een $n \times n$ Latijns vierkant herleiden tot het aanvullen van een $n^2 \times n^2$ sudoku. Charles J. Colbourn had in 1984 al aangetoond dat het aanvullen van een Latijns vierkant NP-volledig was.

Latijns Vierkant

Een Latijns vierkant is een $n \times n$-tafel waarin n getallen zo moeten worden geplaatst dat elk getal een keer in iedere rij en kolom voorkomt. De naam Latijns vierkant komt van Leonhard Euler die Latijnse letters gebruikte in zijn vierkanten. Een Latijns vierkant kan een leuke puzzel zijn als een gedeelte van de vakjes nog niet ingevuld is. Het is dan de bedoeling het vierkant aan te vullen tot een geldig Latijns vierkant.

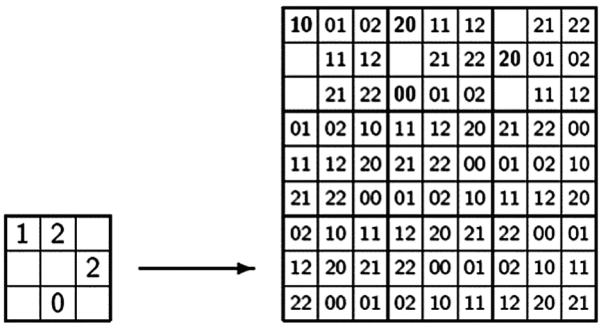
1	2	3	4	5	6	7	8	9
2	3	1	5	6	4	8	9	7
3	1	2	6	4	5	9	7	8
4	5	6	7	8	9	1	2	3
5	6	4	8	9	7	2	3	1
6	4	5	6	7	8	3	1	2
7	8	9	1	2	3	4	5	6
8	9	7	2	3	1	5	6	4
9	7	8	3	1	2	6	4	5

Alle NP-vraagstukken kunnen dus gereduceerd worden tot het aanvullen van een Latijns vierkant. Het aanvullen van een $n \times n$ Latijns vierkant kan op zijn beurt weer worden gereduceerd tot het aanvullen van een $n^2 \times n^2$ sudoku. Het gaat wat ver om dit hier uitgebreid te behandelen, maar onderstaand plaatje is een voorbeeld van hoe het aanvullen van een 3 × 3 Latijns vierkant kan worden vertaald naar het aanvullen van een 9 × 9 sudoku.

		4		7	9	6	8	5
			8	4	5	7	9	6
8		9	4		6		5	7
6	9		7		4	5		8
	6	8		5		4	7	9
4	5	6		8	7	9		
9	7		5	6		8	4	
7	8	5	6	9				4
5	4	7	9		8		6	

Kan dit worden aangevuld tot een Latijns vierkant?

Dit betekent dat als er een polynomiaal algoritme wordt gevonden voor het oplossen van een $n^2 \times n^2$ sudoku, dan bestaat er een polynomiaal algoritme om Latijnse vierkanten aan te vullen. Vanwege de NP-volledigheid van beide problemen is dat algoritme het bewijs van P = NP. Een polynomiaal algoritme voor het oplossen van een sudoku of Latijns vierkant zou dus de oplossing zijn voor één van de milleniumproblemen. Dat is leuk om in het achterhoofd te houden bij het puzzelen.



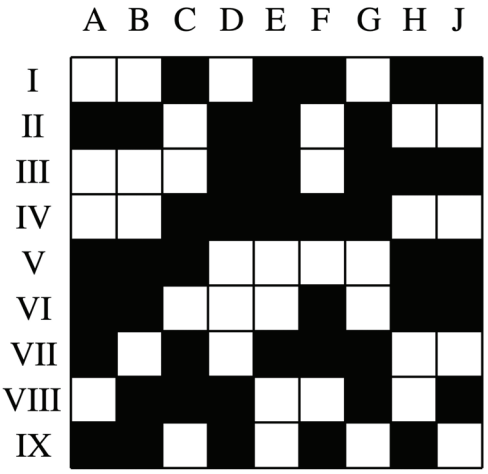
1	2	
		2
	0	

10	01	02	20	11	12		21	22
	11	12		21	22	20	01	02
	21	22	00	01	02		11	12
01	02	10	11	12	20	21	22	00
11	12	20	21	22	00	01	02	10
21	22	00	01	02	10	11	12	20
02	10	11	12	20	21	22	00	01
12	20	21	22	00	01	02	10	11
22	00	01	02	10	11	12	20	21

Het aanvullen van een Latijs vierkant (links) kan worden herleid tot het aanvullen van een sudoku (rechts). De oplossing van de sudoku geeft ook een oplossing voor het Latijnse vierkant.

Roosterpuzzel

Voor het aanvullen van sudoku's of Latijnse vierkanten bestaat geen polynomiaal algoritme. Als we deze vraagstukken een beetje versimpelen, krijgen we een puzzel waarvoor wel een mooi algoritme bestaat dat tot de oplossing leidt. We bekijken een roosterpuzzel. Een methode om deze puzzel op te lossen is polynomiaal en goed te begrijpen. Bovendien vormt de methode ook gelijk het bewijs van een belangrijke stelling over de lijnkleuring van bipartiete grafen. Het zal blijken dat het vanuit wiskundig oogpunt een leukere puzzel is.



De bedoeling in een roosterpuzzel is om in de witte vakjes een 1, 2, 3 of 4 te zetten. Restrictie hierbij is wel dat ieder getal slechts eenmaal in elke rij of kolom voor mag komen. (Het is eigenlijk een Latijns vierkant waarbij alle getallen tussen de 5 en 9 al zijn ingevuld.)

De naam van de puzzel komt van de praktische toepassing ervan. Bij het maken van een schoolrooster is het de bedoeling de docenten aan de verschillende klassen te koppelen. De te roosteren combinaties zijn met een wit vakje aangegeven. Het is niet mogelijk dat een docent in een lesuur voor meerdere klassen staat. Het is evenmin de bedoeling dat er meerdere docenten voor dezelfde klas komen te staan. Men heeft vier lesuren tot de beschikking en elke docent beschikt over een eigen klaslokaal (het is dan ook een sterk vereenvoudigde versie van een werkelijke roosterpuzzel). Door in elk wit vakje de getallen 1, 2, 3 of 4 in te vullen zodanig dat ieder getal slechts eenmaal in iedere rij of kolom voorkomt, wordt het gewenste rooster verkregen. Wanneer bijvoorbeeld in het vakje (VIII, E) een 3 wordt ingevuld, moet klas VIII zich voor het derde uur melden in het lokaal van docent E.

Probeer zelf zoveel mogelijk getallen alvast in te vullen en lees verder als u bent vastgelopen of het heeft opgelost.

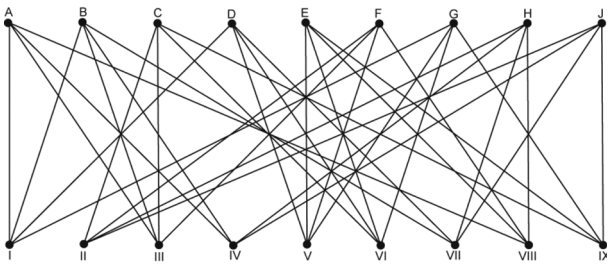
De stelling van König

Een methode voor het oplossen van een roosterpuzzel is gebaseerd op een bewijs van de stelling van König. Deze stelling zegt dat elke bipartiete graaf met maximum graad k k-lijnkleurbaar is.
Een paar begrippen op een rij:

- Een bipartiete graaf is een graaf waarvoor geldt dat de puntenverzameling in twee delen kan worden gesplitst, zodanig dat de punten in beide delen onderling niet verbonden zijn.
- De graad van een punt is het aantal lijnen dat in het punt samenkomt.
- De maximum graad van een graaf is de graad van het punt waar de meeste lijnen in samenkomen.
- In een geldige lijnkleuring moeten alle lijnen die samenkomen in een punt een verschillende kleur hebben.

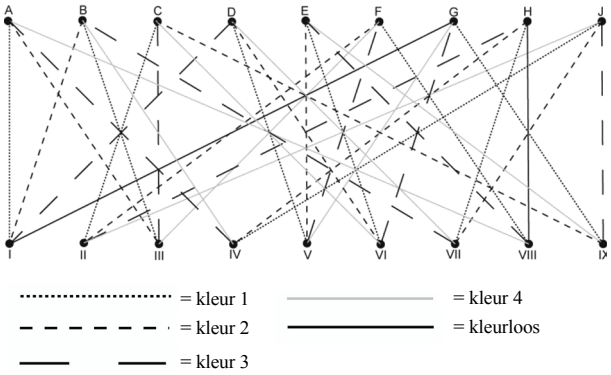
Het bewijs van deze stelling berust op een algoritme waarmee de lijnen van de bipartiete graaf $G = (V, E)$ met k kleuren gekleurd kunnen worden.

We gaan de stelling bewijzen door een manier te geven waarmee we elke bipartiete graaf G kunnen kleuren met k kleuren. We maken deze methode inzichtelijk door het gelijk toe te passen op de graaf van het bovenstaande voorbeeld. We kennen aan iedere kolom in de puzzel een punt toe in de ene partij en aan iedere rij een punt in de andere partij. Een wit vakje (X, Y) in de roosterpuzzel correspondeert met een lijn tussen de punten X en Y.



De bovenste rij punten stelt de kolommen van de roosterpuzzel voor, en de onderste rij punten de rijen. Tussen twee punten X en Y loopt een lijn als het vakje dat de doorsnede van kolom X en kolom Y vormt, wit is.

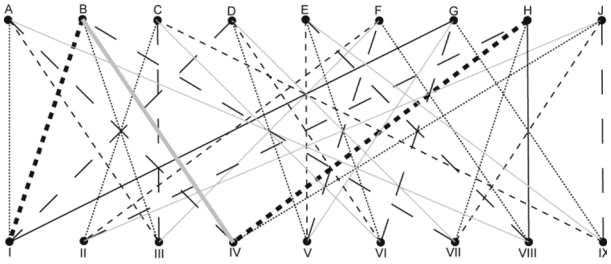
In geen van de punten in de graaf komen meer dan vier lijnen samen. De maximum graad van onze graaf is dus vier. Volgens de stelling van König zouden we de graaf met vier kleuren moeten kunnen kleuren. Deze kleuren kunnen worden gezien als de mogelijke uren die we in de roosterpuzzel moesten invullen. In principe is het mogelijk om met het algoritme de lijnen één voor één een kleur te geven. Dit is vrij omslachtig en we helpen het algoritme een eindje op weg door eerst zelf zo veel mogelijk lijnen te kleuren. Pas als we met onze kleuring zijn vastgelopen, passen we het algoritme toe.



We mogen maar vier kleuren gebruiken en de lijnen moeten wel correct gekleurd worden. Dat wil zeggen, geen twee lijnen die in een willekeurig punt v samenkomen mogen dezelfde kleur hebben. In de roosterpuzzel zou dit namelijk betekenen dat een docent in een lesuur aan twee klassen moet lesgeven of dat er twee docenten voor dezelfde klas komen te staan.

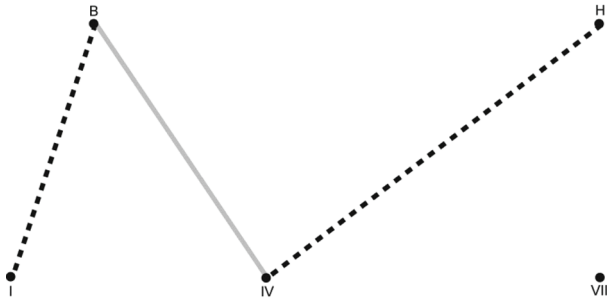
We mogen ervan uitgaan dat het niet lukt om alle lijnen te kleuren. Er blijven dus lijnen ongekleurd. In de tekening is bijvoorbeeld de lijn $e = \{H, VIII\}$ ongekleurd.
Er kunnen ten hoogste 4 lijnen in H samenkomen en de lijn e is nog niet gekleurd. Er moet dus nog een kleur zijn die niet gebruikt is voor de in H samenkomende lijnen. Dit is kleur 4. Evenzo bestaat er een kleur die niet gebruikt is voor de in VIII samenkomende lijnen, namelijk kleur 2.

We gaan nu vanuit H een pad maken dat afwisselend lijnen met de kleuren 2 en 4 doorloopt. We willen dat dit pad maximale lengte heeft. We zullen deze eigenschap later in het bewijs nodig hebben. Noem dit langste pad P. In de tekening hebben we P met dikke lijnen aangegeven.

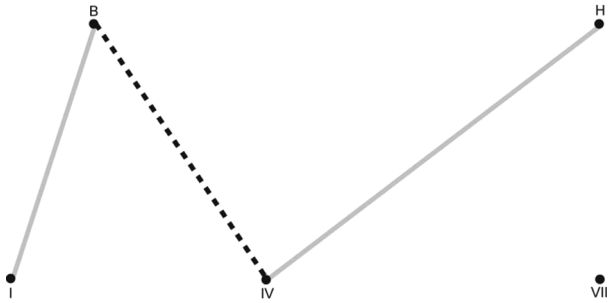


Dit pad kan nooit in het punt VIII eindigen, omdat we hadden aangenomen dat er geen lijnen met de kleur 2 in het punt VIII uit zouden komen.

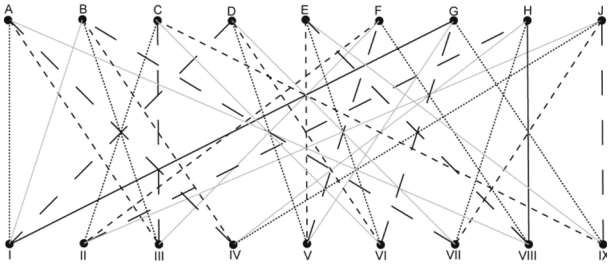
We lichten het pad P eruit.



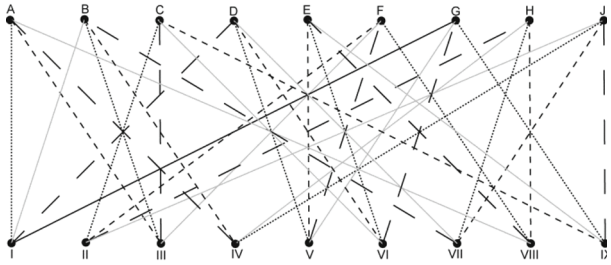
Het is nu mogelijk om de kleuren in P om te wisselen. De lijnen die eerst kleur 2 hadden, krijgen kleur 4 en wat kleur 4 had, krijgt kleur 2.



De kleuring van de graaf blijft op deze manier correct. Stel dat er een foute kleuring zou ontstaan door het verwisselen van de kleuren langs P. Er zou dan in de nieuwe kleuring in minstens één van de punten dus twee keer een lijn met dezelfde kleur moeten samenkomen. Dat zou echter betekenen dat het pad P met een lijn verlengd had kunnen worden en dat is in tegenspraak met de aanname dat P maximale lengte heeft.



We zien nu dat er een kleur vrij is gekomen om e te kleuren. We kunnen e de kleur 2 geven.



Op deze manier hebben we een lijn meer kunnen kleuren. Deze methode kunnen we gebruiken om alle ongekleurde

lijnen te kleuren. Uiteindelijk zullen we een correcte kleuring met vier kleuren vinden voor alle lijnen. In de puzzel kan dit algoritme direct worden toegepast. Hoe dat werkt, is ook nog wel een aardig puzzeltje.

Sudoku

De roosterpuzzel is een afgezwakte variant van het aanvullen van een Latijns vierkant. Een Latijns vierkant kan op zijn beurt weer worden herleid tot het aanvullen van een sudoku. In tegenstelling tot het aanvullen van sudoku's, bestaat er voor het oplossen van een roosterpuzzel wel een polynomiaal algoritme. Het invullen van een roosterpuzzel heeft dan ook een lagere complexiteit dan het aanvullen van een sudoku. Voor wiskundigen valt er wel meer te vertellen over een roosterpuzzel. Roosterpuzzels vormen eigenlijk een grafentheoretisch probleem waarover veel theorie is ontwikkeld en waarvoor een snelle methode bestaat om het op te lossen. Wiskundig gezien mooi, maar voor de puzzelindustrie niet erg interessant.

Hoewel wereldkampioene sudoku Jana Tylova waanzinnig snel een sudoku kan oplossen, is haar prestatie slechts marginaal. Ook Jana zal niet een echt slimme methode tot haar beschikking hebben om een sudoku op te lossen. Het oplossen van een sudoku is een NP-volledig probleem en de meeste wiskundigen geloven niet dat er ooit een polynomiaal algoritme zal worden gevonden om een sudoku op te lossen. Bewezen is dat echter nooit, waardoor de deur naar eeuwige roem en een miljoen dollar van het Clay Institute for Mathematics voor iedereen nog altijd wagenwijd openstaat.

Denk daar maar eens aan als u een sudoku aan het invullen bent. Een nieuwe polynomiale methode zou zomaar om de hoek kunnen liggen op een plek waar niemand het verwacht en u zou net zo goed als alle andere mensen die zich met dergelijke vraagstukken bezig houden, zomaar tegen de oplossing aan kunnen lopen. Een miljoen dollar zal de uwe worden.

Niels Oosterling,
de Praktijk, Amsterdam

Literatuur

Takayuki Yato. (2003). http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.pdf. University of Tokio.

Alexander Schrijver. (2006). http://homepages.cwi.nl/~lex/files/dict.pdf. Centrum voor Wiskunde en Informatica.

Alexander Schrijver. (2000). http://homepages.cwi.nl/~lex/files/graphs1_3.pdf. Centrum voor Wiskunde en Informatica.

Klaas Pieter Hart. (2006). http://dutiaw37.twi.tudelft.nl/~kp/nwd/nwd-handout.pdf. Technische Universiteit Delft.

Kerstpuzzel

Na alle theorie over het oplossen van sudoku's op de voorgaande bladzijden, nog even wat praktijk voor in de kerstvakantie....

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

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

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

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

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

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