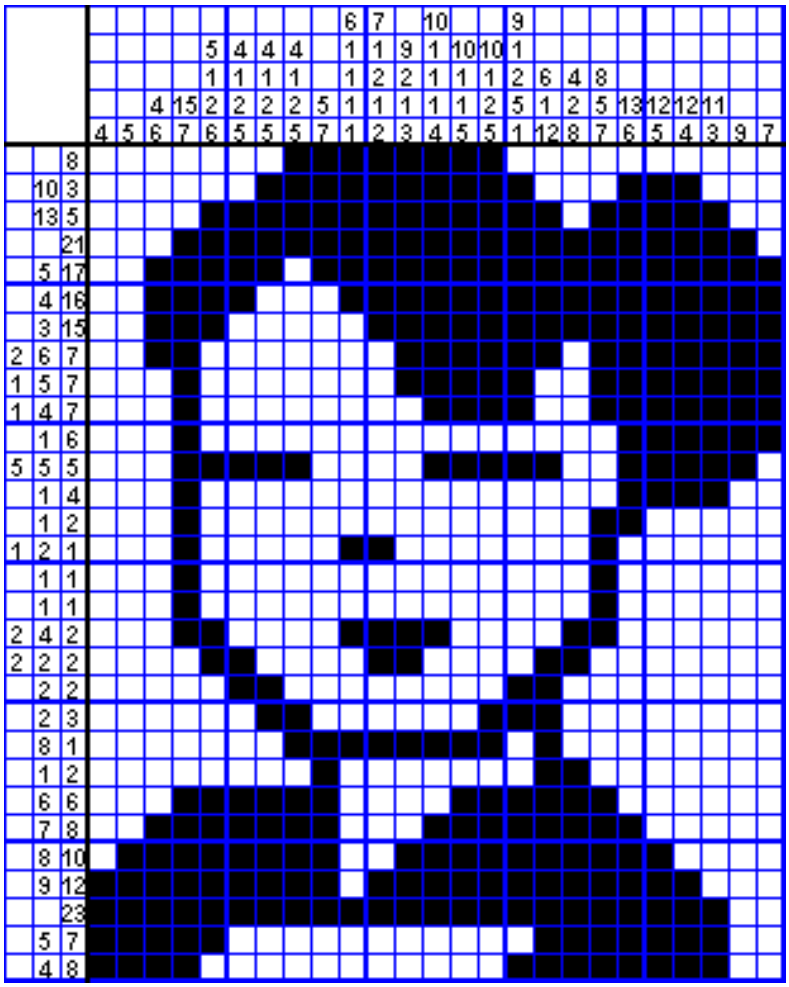
Japanse puzzels.

Wiskunde en informatica in Japanse beeldzoekers.



Een Japanse puzzel met de afbeelding van Non Ishida.

Willem van der Vegt

21 maart 2007.

Uitgave voor de alumnidag Windesheim lerarenopleiding VO.

Vooraf.

Al meer dan tien jaar ben ik, via mijn werk voor de Informatica Olympiade, maar ook door mijn belangstelling voor puzzels en de wiskundige aspecten ervan, geconfronteerd met zogenaamde Japanse puzzels. Voor mij begon het met de opgave van de eerste dag van de Internationale Informatica Olympiade in Bonn in 1992. De verwijzing naar Japan kwam in de opdrachtformulering overigens niet voor. Kort daarna publiceerde Jan de Geus in de puzzelrubriek van Euclides, het tijdschrift van de Nederlandse Vereniging van wiskundeleraren, een Japanse puzzel uit The Sunday Telegraph. Ik heb het kader van de puzzel zorgvuldig overgenomen als tabel in Wordperfect en heel wat exemplaren uitgedraaid om een goede oplossing te vinden. Naast het maken van de puzzel zelf heb ik ook nog wel even zitten puzzelen op de vraag of ik een formule kon vinden voor het aantal mogelijke oplossingen voor één regel van zo'n puzzel.
In de voorbereiding van de Internationale Informatica Olympiade van Eindhoven dook het puzzeltype opnieuw op; helaas konden we het niet zo snel al weer gebruiken. Ook andere puzzels werden onder de loep genomen, en uiteindelijk werd een letterspel gekozen als één van de opgaven.
Vlak voor de zomervakantie van 1998 trof ik bij de boekhandel een voor mij nieuw type puzzelboekje aan, in het geheel gewijd aan Japanse puzzels. Bijna het hele gezin Van der Vegt heeft die zomer enthousiast aan deze puzzels gewerkt, en sinds die tijd worden de nieuwe uitgaven van deze boekjes regelmatig aangeschaft.
In het voorjaar van 2001 liep ik op een heel andere manier weer tegen de Japanse puzzels op. Na een nascholingscursus over kunstmatige intelligentie had ik het op me genomen om het materiaal van die dag te bewerken voor gebruik in een afsluitend college van de opleiding docent informatica. Ik werd al een tijdje gefascineerd door het gebruik van principes uit de biologie in het heuristisch zoeken, dat is het zoeken naar relatief goede oplossingen voor problemen die onaangenaam veel mogelijke oplossingen hebben. Op zoek naar leesbare teksten, liefst in het Nederlands, stuitte ik op een opdracht die Walter Kosters van de Universiteit Leiden aan zijn studenten gaf. Ze moesten een oplossing vinden voor een kleine Japanse puzzel door gebruik te maken van een genetisch algoritme. Ik nam zijn opdracht als bijlage op in ons cursusmateriaal; het jaar erop maakte ik in de voorbereiding op hetzelfde college enkele kleine programma's om te laten zien hoe de betreffende algoritmes werken. Naast twee programma's voor het tonen van de werking van een genetisch algoritme modelleerde ik ook het concept van het voedsel zoeken door een mierenkolonie. De context van de Japanse puzzels, waar bij iedere kandidaatoplossing een plaatje te tonen is, werkte wat mij betreft heel plezierig en in heb veel van het schrijven van deze programma's geleerd. De programma's horen bij het tweede gedeelte van dit boekje; ze zijn te vinden op http://www.willemvandervegt.nl/japans.zip
Inmiddels was de tijd rijp voor een nieuwe opgave over Japanse puzzels. Opgave 1 van de eerste ronde van de Nederlandse Informatica Olympiade 2002-2003 werd, na een veel te moeilijke eerste formulering uiteindelijk de opdracht om, gegeven een afbeelding, de omschrijving als Japanse puzzel erbij te vinden. Deze opgave is door vrijwel alle 130 deelnemers tot een goed einde gebracht.
In dit boekje heb ik mijn eigen avonturen met Japanse puzzels op een rijtje gezet. In eerste instantie voor de deelnemers aan de prijsuitreiking van de Nederlandse Informatica Olympiade 2002-2003 en voor de nascholingsconferentie van de UT; wellicht vindt het later zijn weg ook op een andere manier, naar mijn hbo-studenten of naar leerlingen uit de tweede fase van het voortgezet onderwijs. Zowel voor wiskunde als voor informatica zijn er wellicht toepassingen mogelijk.
Al schrijvend doemden er nog weer enkele nieuwe vragen op; ik heb er met plezier aan gewerkt en ik hoop uiteraard dat je dat eraan af zult kunnen zien.

Willem van der Vegt
Christelijke Hogeschool Windesheim
Lerarenopleiding Voortgezet Onderwijs
Postbus 10090
8000 GB Zwolle
Email: willem@informaticaolympiade.nl

Inhoud:

Inleiding.

Wat zijn Japanse puzzels?

In de tweede helft van de jaren negentig verschenen puzzelboekjes met een nieuw type puzzels. In Nederland worden ze aangeduid als Japanse puzzels of Japanse beeldzoekers. Beide grote Nederlandse puzzelleveranciers, Denksport en Puzzelsport, geven enkele malen per jaar een boekje uit met uitsluitend dergelijke puzzels. En ook in boekjes met verschillende soorten logische puzzels staan vaak Japanse puzzels vermeld.

Afbeelding 1. De omslag van een recent boekje van Denksport met Japanse puzzels.

Een Japanse beeldzoeker is een puzzel waarvan de oplossing bestaat uit een rechthoek met zwarte en witte vierkantjes. Bij iedere rij en kolom staan getallen aangegeven die vastleggen hoeveel aaneengesloten zwarte vierkantjes er te vinden zijn. Zo betekent '4 3 1' dat de betreffende rij of kolom bestaat uit eerst een groep van vier aaneengesloten zwarte vierkantjes, gevolgd door een groep van drie, met tenslotte nog een los zwart vierkantje. Groepen worden gescheiden door minstens één wit vierkantje. Voor de eerste groep kan eventueel een groep voorkomen met witte vierkantjes; ook na de laatste groep kan dat het geval zijn.

In Engeland publiceert The Sunday Telegraph al vanaf 1991 wekelijks een Japanse puzzel. In het tijdschrift Euclides nam puzzelredacteur Jan de Geus de volgende puzzel eruit over als opgave Recreatie 652:

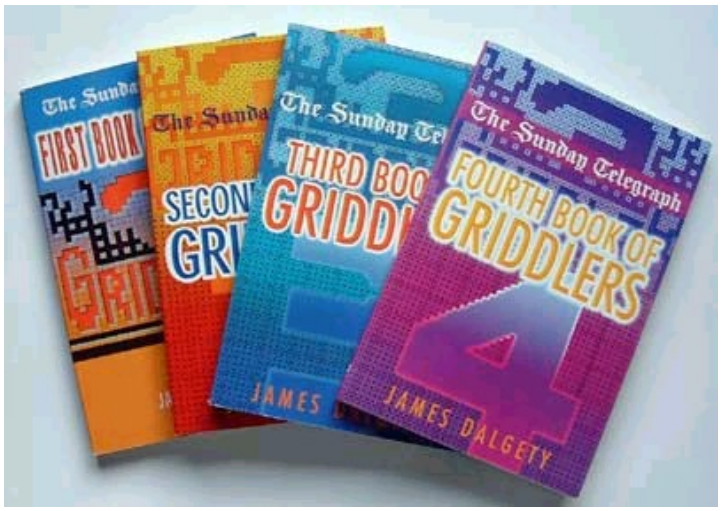
											1														
										3	4														
					2	2	2	2	3	2	3	9					2								
					1	2	3	4	5	3	4	4					6	2	3	2	3	2
				3	5	3	2	3	1	1	2	5	3	15	15	15	15	3	2	3	2	3	5
			3																						
			4																						
			5																						
			4																						
			8																						
		2	11																						
	3	8	3																						
		10	2																						
	2	5	2																						
			4																						
			5																						
			7																						
			11																						
			11																						
		3	7																						
		3	4																						
	3	3	2																						
2	2	2	3																						
3	1	2	5																						
	5	3	4																						
	3	5	4																						
			2																						
			2																						
			1																						
			1																						

Afbeelding 2. Opgave recreatie 652 van Euclides.

Er zijn ook Japanse puzzels met meerdere kleuren. In dit boekje beperken we ons tot Japanse zwart-wit puzzels.

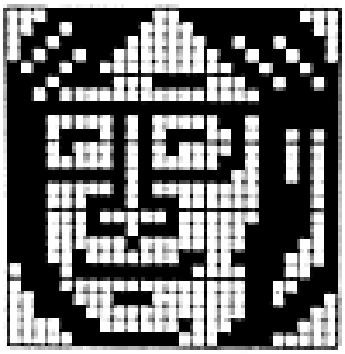
Non Ishida en de ramen van de flats.

In 1987 was er in Tokio een wedstrijd voor ontwerpers van "Window Art"; bij deze wedstrijd ging het er om figuren te maken door van een wolkenkrabber een deel van de ramen te verlichten. Het winnende ontwerp kwam van Non Ishida. Het toonde prinses Taketori Monogatari, een heldin uit de achtste eeuw, die op een lichtstraal van de maan kwam en in een soort voertuig van licht terugkeerde (Voor de geschiedenis van Taketori Monogatari zie http://home.clara.co.uk/wabei/xlation/kaguya/kagtoc.htm) Voor haar ontwerp maakte ze uiteraard een aantal afbeeldingen van de wolkenkrabber, en in de kantlijn legde ze per rij en kolom het aantal verlichte ramen vast. Dat bracht haar op de vraag of je, zonder het plaatje te tekenen, de afbeelding uit die informatie zou kunnen afleiden. Met dat idee maakte ze drie puzzels die ze in 1988 publiceerde.
In 1989 toonde ze James Dalgety haar puzzels. Die zag meteen de mogelijkheden ervan en hij slaagde erin om de puzzels gepubliceerd te krijgen in The Sunday Telegraph. Hij noemde de puzzels Nonograms, naar de ontwerpster.
Later wilde Non Ishida deze naam uitsluitend laten gebruiken voor puzzels die ze zelf had ontworpen. Na een prijsvraag werd de nieuwe naam van deze puzzels in The Sunday Telegraph "Griddlers".



Afbeelding 3. De jaarlijkse puzzelboekjes van The Sunday Telegraph.

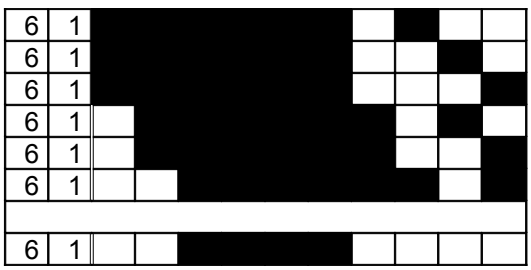
Een andere Japanse puzzelmaker die waarschijnlijk onafhankelijk van Non Ishida het idee van de Japanse beeldzoeker heeft ontwikkeld is Tetsuya Nishio. Hij spreekt over Oekaki Logic; Oekaki is een Japanse tekentechniek.



Afbeelding 4. Zelfportret van Tetsuya Nishio.

Oplossingsmethoden.

Om een Japanse beeldzoeker op te lossen maak je gebruik van het feit dat alle mogelijke invullingen van een regel (een rij of een kolom) elkaar soms overlappen. Door gebruik te maken van deze overlap is het mogelijk steeds meer vakjes te kleuren; ook is het mogelijk uit te sluiten dat bepaalde vakjes gekleurd moeten worden.
Een voorbeeld:



Afbeelding 3. 6 mogelijke oplossingen en hun overlap.

Voor een regel van 10 vakjes geldt als omschrijving 6 1. Er zijn zes verschillende manieren waarop de zwarte vakjes kunnen worden ingevuld. Op de onderste regel van de figuur zie je de overlap van die zes manieren; van deze drie vakjes ben je zeker dat ze zwart moeten worden.

Stel nu eens dat al bekend is dat het negende vakje vanaf links zwart moet zijn (informatie van een regel die hier loodrecht op staat). Van de zes mogelijkheden blijven er nu nog maar twee over.

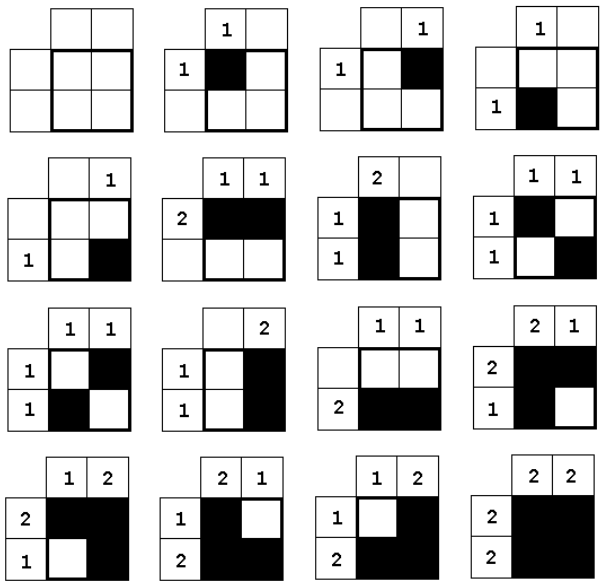
Afbeelding 4. 2 mogelijke resterende oplossingen en hun overlap.

Nu zijn er maar liefst zeven vakjes waarvan de kleur bekend is; vijf zijn er zwart, twee blijven zeker wit.

Tellen aan Japanse puzzels.

Hoeveel puzzels zijn er?

Als we ons beperken tot puzzels die uit zwarte en witte vakjes bestaan, zijn er voor ieder vakje twee mogelijkheden. Hieronder staan alle mogelijke Japanse puzzels van 2 bij 2 aangegeven.



Afbeelding 5. Alle verschillende Japanse puzzels van 2 bij 2 vakjes.

Vraag 1. Waarom zijn er 16 verschillende puzzels van 2 bij 2 vakjes?

Vraag 2. Hoeveel verschillende puzzels zijn er van 5 bij 5 vakjes?

Puzzels van 2 bij 2 of van 5 bij 5 zijn vierkant. Uiteraard zijn ook andere verhoudingen mogelijk. Daarom de volgende algemenere vraag:

Vraag 3. Hoeveel verschillende puzzels zijn er van N bij M vakjes?

Zoals je ziet neemt het aantal mogelijkheden dramatisch toe. Bij een puzzel van een beetje formaat is het aantal mogelijkheden gigantisch. Om dat aantal in de goede verhoudingen te kunnen zien de volgende opgave.

Vraag 4. Hoeveel vakjes telt een kader van een Japanse puzzel die tenminste 10^{125} mogelijke invullingen kent (dit is volgens sommige schattingen het aantal deeltjes in het heelal)?

Hoeveel verschillende omschrijvingen van één regel zijn er?

Bij een regel kunnen verschillende omschrijvingen horen. Is te bedenken of uit te rekenen hoeveel verschillende omschrijvingen horen bij een regel van N vakjes?
Meestal is het wel handig om eerst eens een paar kleine gevallen op te schrijven.

Vraag 5. Geef voor N=1 tot en met N=5 alle mogelijke omschrijvingen van een regel van N vakjes, en tel voor elke waarde van N het aantal verschillende mogelijke omschrijvingen O(N).

Het zou mooi zijn als je uit de waarde van het aantal mogelijke omschrijvingen van N, uit O(N), O(N +1) kon uitrekenen. Helaas lijkt dat hier nog niet zo makkelijk.
Het eerste houvast zou je kunnen vinden in de al gevonden getallen bij vraag 5. Deze getallen komen ook voor bij een bekende rij uit de geschiedenis van de wiskunde.

Vraag 6. Ga na welke getallenrij de getallen bevat uit het antwoord op vraag 5. Als je zelf geen idee hebt kun je misschien op zoek naar een pagina op Internet met allerlei rijen van gehele getallen. Twee voorbeelden van dergelijke sites zijn
http://www.research.att.com/~njas/sequences/ en
http://research.interface.co.uk/gus/index.htm

Eén observatie die je het verband tussen de gevonden getallenrijen en het probleem van de aantallen omschrijvingen kan helpen vinden, is de volgende: Alle omschrijvingen van een regel met N vakjes zijn ook bruikbaar voor een regel met N+1 vakjes. De enige omschrijvingen extra zijn die waarbij alle N+1 vakjes al vastliggen.

Vraag 7. Geef nu een formule voor het aantal mogelijke omschrijvingen O(N) van een regel met N vakjes, uitgedrukt in O(N-1) en O(N-2) en bewijs dat deze formule voor iedere N voor N>2 opgaat. Bedenk dat uit het antwoord op vraag 5 al geldt dat O(1)=2 en O(2)=3.

Hoeveel verschillende oplossingen horen er bij de omschrijving voor één regel?

Bij een specifieke omschrijving van één regel hoort, als je geen verdere informatie hebt, een bepaald aantal oplossingen. Dit aantal hangt er van af hoeveel "speling" je hebt, hoeveel witte hokjes je nu vrij kunt kiezen.
In het voorbeeld was voor een regel van 10 vakjes de omschrijving 6 1. Er zijn nu al 8 vakjes bekend; een groep van 6 zwarte, 1 witte en 1 zwarte. Er zijn nu nog 2 witte vakjes vrij te kiezen.

Vraag 8: Op hoeveel manieren kunnen deze witte vakjes worden geplaatst?

Vraag 9: Bepaal bij tenminste 5 regels van de puzzel uit voorbeeld 1 het aantal mogelijke invullingen van de betreffende regel.

Misschien komen de getallen die je hebt gevonden je bekend voor; het zijn zogenaamde binomiaalcoëfficiënten, die onder meer in de waarschijnlijkheidsrekening worden gebruikt. Ga zo nodig eerst op zoek naar wat je al over deze getallen hebt geleerd.
Nu moeten we nog zien uit te vinden wat de relatie is tussen deze binomiaalcoëfficiënten en de vraag uit deze paragraaf.

Vraag 10: Geef een algemene formule voor het aantal mogelijke oplossingen voor een regel van N vakjes met M groepen en in totaal Z zwarte vakjes.

Hoeveel puzzels zijn uniek?

Eén van de lastigste vragen die puzzelmakers moeten zien op te lossen is de vraag in hoeverre ze een unieke puzzel hebben gemaakt. Als een willekeurig patroon van zwarte en witte vakjes wordt gemaakt, hoe groot is dan de kans dat de puzzel, die ontstaat door de beschrijving van het patroon te geven, precies één oplossing heeft?

Vraag 11: Hoeveel verschillende omschrijvingen zijn er voor Japanse puzzels van 2 bij 2 vakjes?

Omdat er minder omschrijvingen zijn dan mogelijke invullingen moeten er verschillende invullingen zijn die bij dezelfde omschrijving horen. In het geval van 2 bij 2 vakjes geldt dat voor de omschrijving met bij elke rij en elke kolom een 1; deze heeft twee oplossingen.

Vraag 12: Zoek een omschrijving van een Japanse puzzel van 5 bij 5 waarbij bij deze omschrijving tenminste 16 verschillende oplossingen horen.

Zijn er puzzels die uniek zijn maar niet op te lossen?

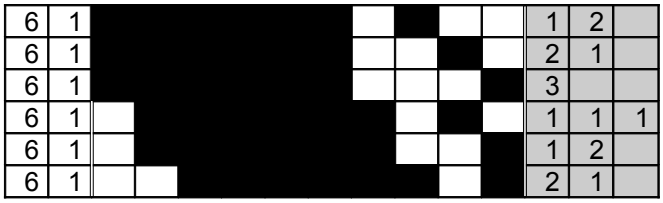
Een omschrijving van een Japanse puzzel is uniek als er maar één afbeelding hoort bij die omschrijving. Voor puzzelontwerpers is de vraag van belang of iedere unieke omschrijving ook leidt tot een oplossing. Zijn er puzzels waarbij je vastloopt, terwijl ze toch maar één oplossing hebben?

Het negatief van een Japanse puzzel.

Voor een puzzel in zwart-wit wordt normaal gesproken de informatie over de zwarte vakjes gegeven. Het negatief van een Japanse puzzel krijg je door juist de informatie over de witte vakjes te geven. In deze paragraaf komen enkele vragen over puzzels en hun negatief aan de orde.

Hoeveel getallen heeft de omschrijving van een regel van het negatief?

In afbeelding 3 is een voorbeeld gegeven van de mogelijke invullingen van een regel van 10 vakjes met als omschrijving 6 1.
Hieronder zie je dezelfde figuur, met nu rechts de omschrijving van de negatieve puzzel.



Afbeelding 6. Links de omschrijving, rechts de omschrijving van het negatief.

Vraag 13: Als een regel uit N vakjes bestaat, met M groepen zwarte vakjes en in totaal Z zwarte vakjes, hoeveel witte vakjes zijn er dan?

Vraag 14: Hoeveel groepen witte vakjes zijn er op een regel zoals beschreven in vraag 13?

Is het negatief van een unieke puzzel ook uniek?

Het antwoord op de vraag of het negatief van een unieke puzzel ook uniek is, is nog niet zo makkelijk te geven.
Bij de puzzels van 2 bij 2 vakjes geldt dat voor een puzzel en zijn negatief hetzelfde te zeggen valt: Beiden zijn uniek, of beiden zijn niet uniek. Voor grotere puzzels hoeft dat niet het geval te zijn.

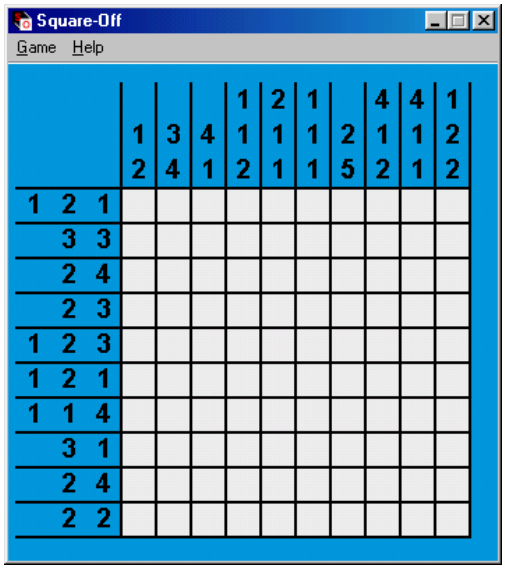
Vraag 15: Zoek een voorbeeld van een puzzel en zijn negatief waarvan één van beide uniek is en de andere niet.

Programma's voor Japanse puzzels.

Enkele voorbeelden.

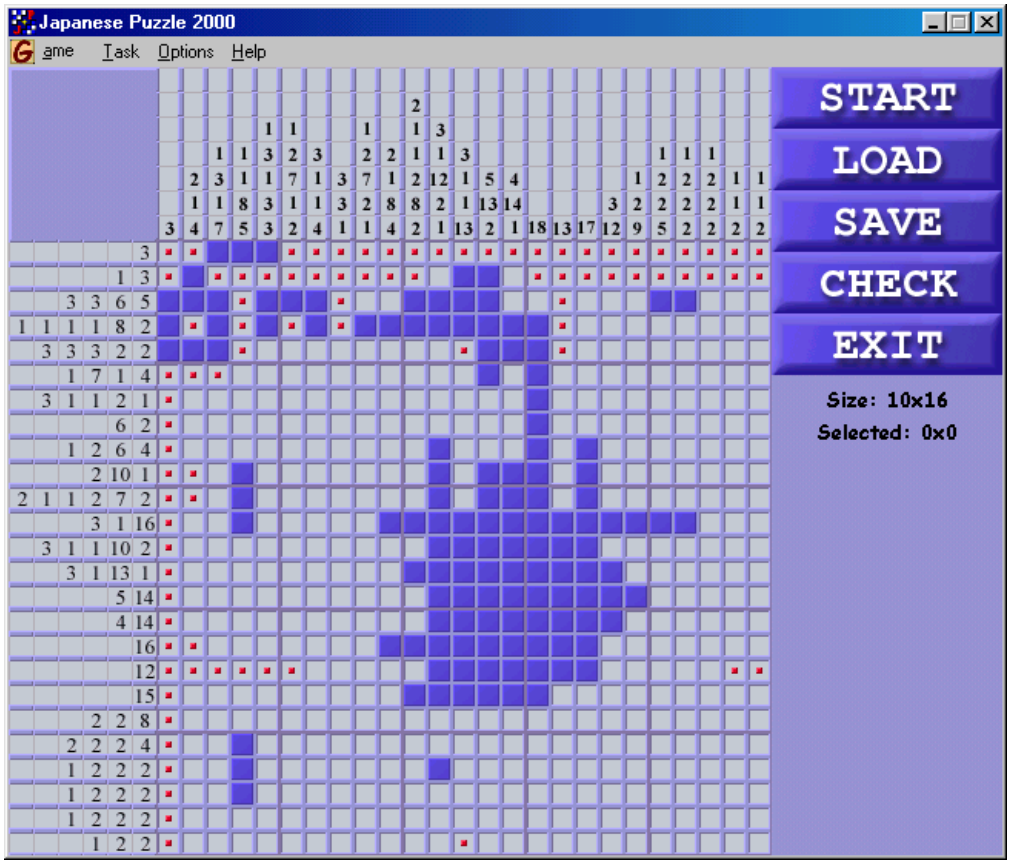
Er zijn veel programma's beschikbaar die je kunt gebruiken om Japanse puzzels op te lossen. Het programma is dan te zien als een vervanging van je puzzelboekje, met een diagram waarop je de

informatie kunt verwerken, en vaak een aantal handige hulpmiddelen. Zoek bijvoorbeeld op "+nonograms +program +download" en er verschijnt een hele collectie.
Hieronder enkele voorbeelden:



Afbeelding 7. Square-Off, te vinden op www.metalgrass.com

Het programma Square-Off geeft toevallig gegenereerde puzzels van drie verschillende niveaus Een vakje als zwart markeren gaat door er op te staan en op de linkermuisknop te klikken; voor herstellen nogmaals klikken. Als wit markeren gaat net zo, maar dan met de rechtermuisknop; er verschijnt een stipje. Wanneer je het niet meer ziet kun je het programma een hint laten geven; het programma maakt je niet duidelijk waarom deze zet een goed idee is.
Wanneer de hele puzzel is opgelost kun je hem laten controleren. Helaas wordt de controle uitgevoerd op grond van de gegenereerde afbeelding; wanneer de puzzel niet uniek was worden alternatieve oplossingen afgekeurd, hoewel ze wel aan de omschrijvingen voldoen.



Afbeelding 8. Het pogramma Japanese Puzzle 2000, van Gunn-soft, www.puzzle.kiev.ua

Het programma Japanese Puzzle 2000 komt met vijftien voorbeeldpuzzels. Je hebt de mogelijkheid deze puzzels tussentijds op te slaan en weer in te lezen. Ook is het mogelijk door de linker- of rechtermuisknop ingedrukt te houden een gebied te selecteren dat in zin geheel moet worden gemarkeerd. Een foutieve markering moet echter hokje voor hokje ongedaan worden gemaakt.
Ook dit programma geeft de mogelijkheid de oplossing te laten controleren; als je een fout hebt gemaakt komt er echter geen enkele aanwijzing waar de fout zit. Wel is het mogelijk de puzzel door het computerprogramma te laten oplossen, maar of dan de al opgeslagen oplossing wordt getoond of alsnog wordt berekend wordt nergens duidelijk.

Vraag 16: Zoek zelf een ander programma waarmee Japanse puzzels kunnen worden opgelost en geef aan waar het programma overeenkomt of verschilt van de twee programma's die hierboven zijn besproken.

Eisen aan een hulpprogramma.

De meeste programma's hebben een eenvoudige userinterface, met een menu en gebruik van de muis om vakjes te selecteren en te kleuren. Als de gebruiker bekend is met andere grid-spelletjes (zoals bijvoorbeeld "mijnenveger") zal bediening van de programma's geen problemen opleveren. Toch zijn er grote verschillen in de mogelijkheden van de programma's voor het werken met Japanse puzzels.

Vraag 17: Stel een lijst met eisen samen die je zou willen stellen aan een goed programma waarmee je Japanse puzzels wilt oplossen. De lijst moet zo zijn opgezet dat je hem kunt gebruiken om verschillende programma's met elkaar te vergelijken.

Oplossen van een Japanse puzzel met een computerprogramma.

Enkele gemeenschappelijke problemen voor alle programma's.

Alle programma's zijn geschreven om de oplossing van een Japanse puzzel te vinden of te benaderen van 5 bij 5 vakjes. Een afbeelding kan dan worden voorgesteld door een vierkant van 5 bits, of door een regel van 25 bits. Een regel van 25 bits is ook op te vatten als een getal met een waarde tussen 0 en 33554431. Dat is dan ook het achterliggende variabelentype dat wordt gebruikt. Om van een regel naar een vierkant te gaan kan gebruik worden gemaakt van de van de bewerkingen mod en div (de rest bij deling door een getal en de gehele uitkomst bij deling door een getal). Als je de rijen en kolommen nummert van 1 tot en met 5 en de bits van 1 tot en met 25 krijg je allerlei nare berekeningen om bij een bitnummer het rijnummer en kolomnummer te vinden. Maar als je begint te tellen bij 0 geeft `bitnummer mod 5` het rijnummer en `bitnummer div 5` het kolomnummer.

Vraag 18: De formules hierboven staan geformuleerd in Pascal. Zoek uit voor de programmeertaal die je gewend bent te gebruiken hoe je de rest van bitnummer gedeeld door 5 en de gehele uitkomst van bitnummer gedeeld door 5 kunt krijgen.

Vraag 19: Hoe zouden de formules hierboven zijn als je kolomnummer, rijnummer en bitnummer wel bij 1 begint te tellen?

Bij dit probleem kan het handig zijn om een maat te hebben voor de kwaliteit van een afbeelding bij een omschrijving. Bij iedere afbeelding wordt daarom geteld hoeveel rijen en kolommen aan de omschrijving voldoen. Dat levert een fitheidwaarde (minstens 0, bij een goede oplossing 10). Veel van de in dit hoofdstuk gepresenteerde programma's worden op het scherm afgebeeld in een kleur die al een beeld geeft van de fitheidwaarde. De volgende kleuren worden gebruikt:

Fitheidwaarde(n)	Kleurcode	Kleur
0, 1, 2, 3	7	Grijs
4	9	Blauw
5	10	Groen
6	11	Cyaan
7	12	Rood
8	13	Paars
9	14	Geel
10	15	Wit

Vraag 20: Dit soort kleurcodes is opgebouwd uit vier bits; één bit zet blauw aan of uit, één bit doet dat met groen, één bit met rood en het vierde bit maakt het verschil tussen gewone en heldere weergave. Schrijf de kleurcodes in het tweetallig stelsel en probeer de opbouw van de kleuren te achterhalen.

In een deel van de programma's worden de onderzochte afbeeldingen afgebeeld op een positie die overeenkomt met hun fitheidwaarde. Als er een nieuwe oplossing met dezelfde waarde wordt gevonden, wordt de vorige overschreven.

Brute force aanpak.

Het programma **`JAP-ALL`** genereert een willekeurige afbeelding van 5 bij 5 vakjes. Er wordt vervolgens een omschrijving van een Japanse puzzel van 5 bij 5 van gemaakt. Nu worden alle mogelijke afbeeldingen één voor één gegenereerd. Midden op het scherm worden de onderzochte afbeeldingen getoond; de fitheidwaarde bepaalt de plaats op het scherm, zodat altijd de meest recent gevonden afbeelding met deze waarde te zien is. Ook wordt een teller getoond, in dit geval is dat het

volgnummer van de onderzochte afbeelding; als je dit getal schrijft in het tweetallig stelsel staan de enen voor de gekleurde vakjes en nullen voor de niet gekleurde.
Als een oplossing is gevonden stopt het programma. Ook als tijdens het zoeken op de X wordt gedrukt stopt het. Wanneer het programma is gestopt, dien je nogmaals op X te drukken om het af te sluiten. Drukken op de D zorgt er voor dat het programma alsnog doorgaat (bijvoorbeeld om te kijken of er nog andere oplossingen zijn bij dezelfde omschrijving).

Vraag 21: Voer het programma uit en laat het een minuut werken. Zet het dan stil door op X te drukken. Hoeveel afbeeldingen heeft het programma nu onderzocht? Bereken hoe lang het in dit tempo duurt om alle 33.554.432 mogelijke afbeeldingen te onderzoeken.

Het doorzoeken van alle mogelijkheden lijkt een langdurige kwestie. Gelukkig zijn er manieren om dit efficiënter te organiseren. Belangrijk daarbij is dat zo snel mogelijk rekening wordt gehouden met zoveel mogelijk informatie.

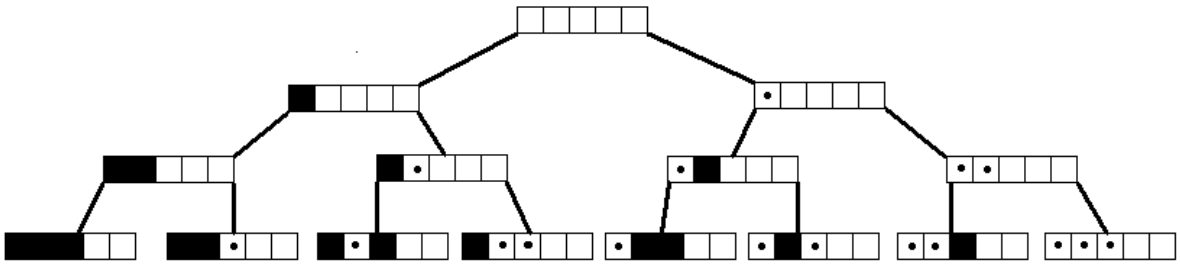
Backtracking.

In het programma **`JAP-TAK`** wordt begonnen met een leeg veld van 5 bij 5 vakjes. De lege hokjes van dit veld worden na elkaar gevuld met het volgende voorschrift:

Is het veld al gevuld, dan ben je klaar en is de oplossing gevonden. Zo niet:
- Onderzoek of het vakje kan worden gekleurd.
- Zo ja, kleur het vakje en ga door met de rest van de puzzel.
- Maak het vakje weer leeg.
- Onderzoek of het vakje mag worden leeg gelaten.
- Zo ja, laat het leeg en ga door met de rest van de puzzel.
- Beëindig je onderzoek en ga verder met dat van het vorige vakje.

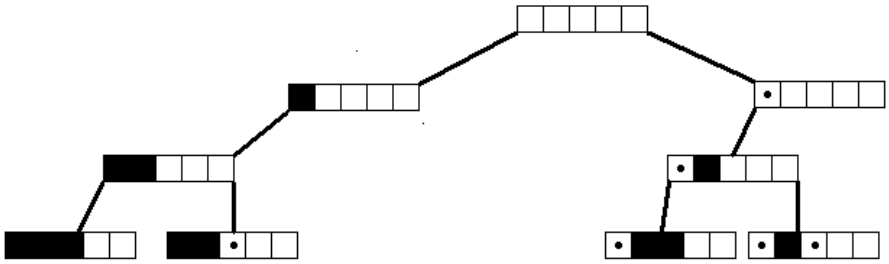
Iedere volgende stap in dit zoekproces wordt getoond door in een vakje een balk of een rondje te plaatsen. Na iedere stap moet op een toets worden gedrukt.
Iedere keer dat een vakje wordt gevuld, wordt de teller die links onder in beeld verschijnt één opgehoogd. Als de puzzel rechtstreeks door het programma kan worden ingevuld, loopt deze teller door tot 25; wanneer het programma een aantal keren op zijn schreden moet terugkeren, wordt dit aantal uiteraard hoger. In een uurtje uitproberen kwam deze waarde nooit boven de 85.

Het algoritme dat hier wordt gebruikt heet backtracking. Voor ieder vakje kun je in principe kiezen tussen kleuren en niet kleuren; maak een keuze, en ga verder. De keuzes die je kunt maken zijn weer te geven als een boom; iedere knoop van niveau N is een bepaalde kleuring van de eerste N vakjes; voor vakje N+1 zijn er twee mogelijkheden. Zoeken naar oplossingen is van het vak boven (leeg veld) naar beneden deze boom doorlopen tot je beneden bij een oplossing bent aangekomen.



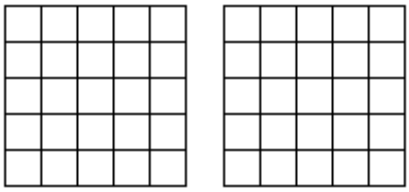
Afbeelding 9. Begin van een boom met alle mogelijke afbeeldingen.

Wanneer het programma uit de omschrijving weet dat een bepaalde voorwaarde geldt (bijvoorbeeld dat de tweede kolom moet beginnen met een gekleurd vakje), hoeven die takken van de boom waarbij niet aan deze voorwaarde wordt voldaan niet verder te worden doorzocht.



Afbeelding 10. Boom waarin de takken met op de tweede plaats geen gekleurd vakje zijn weggelaten.

Vraag 22: Laat het programma een aantal keren werken. Teken in het diagram hieronder een afbeelding die in één keer wordt gevonden (tellerstand 25) en één waarbij het programma zo vaak terug moet dat de teller boven de 50 uitkomt. Geef een verklaring voor het verschil.



Als een puzzelaar.

Het programma **JAP-PUZ** maakt gebruikt van dezelfde manier van redeneren als gewone puzzelaars. Op de manier zoals aan het begin van dit boekje beschreven wordt, afwisselend rij voor rij en kolom voor kolom, nagegaan of er meer informatie bekend is. Dit programma werkt stap voor stap; als de omschrijving van de puzzel niet uniek blijkt, de puzzel niet eenduidig is op te lossen, dan wordt dat door het programma gemeld.

Vraag 23: Ga bij alle omschrijvingen van een regel van 5 vakjes na welke informatie al meteen bekend is.

Lokaal zoeken.

Wanneer de zoekruimte, de verzameling van alle mogelijke oplossingen, erg groot wordt is het vaak noodzakelijk je toevlucht te nemen tot zoekmethodes waarbij niet alle mogelijke oplossingen worden bekeken. Er wordt dan wel gesproken over lokaal zoeken, lokaal in de zin van plaatselijk, in de omgeving van meestal door het toeval bepaalde beginoplossingen.

Het programma **JAP-RAND** zoekt wel op een hele simpele manier: Er wordt voortdurend een nieuwe, door het toeval bepaalde afbeelding gekozen, en deze wordt vergeleken met de omschrijving van de te vinden afbeelding. De uitvoer en de werking komen verder overeen met programma **JAP-ALL**.

Vraag 24: Voor iedere afbeelding wordt nu een toevalsgetal gekozen. De meeste talen geven de mogelijkheid een toevalsgetal te bepalen met behulp van een functie of methode `random`. De aanroep van `random(n)` geeft een geheel getal, minstens 0, kleiner dan n. Helaas gaat de aanroep `random(33554432)` niet op, omdat n een integer moet zijn en dit getal groter is dan het grootste integer. Verzin een manier om toch een goed toevalsgetal te nemen.

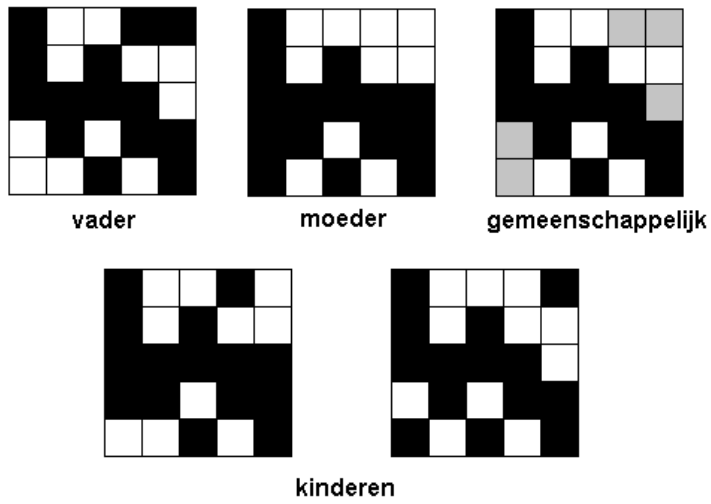
Het programma **JAP-HILL** geeft een beter voorbeeld van een lokale zoekmethode, die bekend staat als hill climbing. Er wordt telkens een door het toeval bepaalde afbeelding gekozen. Vervolgens wordt gekeken of er in de buurt van deze afbeeldingen een verbetering optreedt. Wat is in de buurt? In dit geval zijn twee afbeeldingen buren als ze maar één vakje van elkaar verschillen. Ieder afbeelding heeft dus 25 buren. Van al deze buren wordt geteld wat de fitheidwaarde ervan is; als er een

buurafbeelding is met een hogere fitheidwaarde, wordt degene gekozen waarvan de fitheid het grootst is. Dit wordt herhaald tot je bij een afbeelding bent die geen buren met een hogere waarde heeft. Als je de fitheid als een maat voor de hoogte van de afbeelding beschouwt, dan ga je omhoog tot je niet verder kunt, dus tot je de top van de heuvel bereikt hebt. Dat hoeft uiteraard niet te betekenen dat je niet ergens anders nog hoger kunt komen; maar in de omgeving heb je de top bereikt.

Vraag 25: Er zijn andere algoritmes die zoeken in de omgeving van een door toeval (of anderszins) bepaalde beginwaarde. Twee bekende algoritmes zijn simulated annealing en tabu search. Zoek op Internet een korte beschrijving van deze algoritmes.

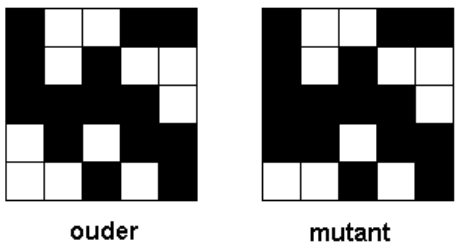
Een oplossing met een genetisch algoritme.

In het midden van de jaren zeventig ontwikkelde John Holland een hele nieuwe methode van lokaal zoeken: Het genetisch algoritme. Bij een genetisch algoritme begin je met een populatie mogelijke oplossingen; in het geval van de Japanse puzzels zijn dat door het toeval bepaalde afbeeldingen. Iedere afbeelding heeft een bepaalde fitheidwaarde. Er is nu een ronde van selectie; de beste exemplaren hebben meer kans om te overleven dan de slechtere. Van de tussenpopulatie die overblijft ontstaat een nieuwe generatie op een drietal manieren: Door voortplanting, door mutaties en door klonen.
In het programma **`JAP-GEN1`** wordt van dit mechanisme een demonstratie gegeven. Er worden in eerste instantie 16 afbeeldingen gekozen; bij deze afbeeldingen wordt de fitheidwaarde berekend. Afbeeldingen met de laagste fitheid vallen af; van de overige wordt door gewogen loting bepaald hoeveel exemplaren ervan in de tussengeneratie terecht komen; deze tussengeneratie wordt op het rechterdeel van het scherm getoond. Vervolgens worden er door loting zeven paren ouders uit deze tussengeneratie gekozen, die elk twee kinderen krijgen voor de nieuwe generatie.



Afbeelding 11. Voortplanting.

Als vader en moeder voor een vakje dezelfde inhoud hebben, krijgen beide kinderen die ook. Als vader gekleurd is en moeder niet (of andersom) krijgt het ene kind een gekleurd vak, het andere een leeg vakje.



Afbeelding 12. Mutatie.

Eén of twee van de afbeeldingen van de tussengeneratie worden gemuteerd. Dat betekent dat één van de vakjes verandert. Nul of één afbeeldingen worden gekloond, dat wil zeggen ongewijzigd overgenomen. Dit gebeurt in ieder geval als de beste vertegenwoordiger uit de tussengeneratie een betere fitheid heeft dan de afbeeldingen uit de nieuwe generatie.
Wanneer dit proces wordt herhaald neemt de gemiddelde fitheid van de populatie gestadig toe. In dit programma worden maximaal 20 generaties doorgerekend. Omdat er maar een kleine populatie is gaan de afbeeldingen al snel erg op elkaar lijken; de echte oplossing zal doorgaans niet worden gevonden.
In het programma **JAP-GEN2** wordt gewerkt met een populatie van 160 afbeeldingen. Deze wordt met hetzelfde mechanisme vernieuwd. Het linkerscherm geeft nu een beeld van de afbeeldingen van de huidige generatie; rechts staat de gemiddelde fitheid van de populatie aangegeven. Na iedere generatieovergang moet op een toets worden gedrukt. Als er 25 overgangen zijn geweest en de oplossing is nog niet gevonden gaat het programma automatisch nog maximaal 100 generaties verder.
Wanneer het programma stopt, omdat de oplossing is gevonden, of omdat er 125 generaties zijn berekend, komt de vraag of je met het programma wilt stoppen. Er zijn nu de volgende mogelijkheden:

J	Stoppen met het programma.
N	Niet stoppen, verder met een andere puzzel.
O	Niet stoppen, opnieuw beginnen met dezelfde puzzel.

Het programma wordt afgesloten door op de Enter-toets te drukken.

Vraag 26: Hoe verandert de fitheid van een afbeelding als gevolg van een mutatie?

Vraag 27: Als twee ouders N vakjes gemeenschappelijk hebben, hoeveel verschillende kinderen kunnen ze dan krijgen?

Het mierenkolonie algoritme.

Een ander algoritme dat gebaseerd is op een analogie met de natuur is bedacht door Marco Dorigo. Hij simuleert het gedrag van een kolonie mieren die op zoek is naar voedsel. Iedere mier onderneemt een tocht vanaf het nest, vindt meer of minder voedsel en loopt langs dezelfde weg terug naar het nest. Op de terugweg laat hij een spoor van feromonen achter, dat sterker is naar mate er meer voedsel is gevonden. De volgende mieren kunnen op verschillende plaatsen kiezen welke richting ze inslaan. Ze laten die keuze beïnvloeden door de door de collega's achtergelaten reukstoffen.
In het programma **JAP-MIER** gaat telkens een groep van 16 mieren op pad; ze produceren elk een afbeelding op basis van een toevalsproces. De fitheid van deze afbeelding is bepalend voor de manier waarop ze hun geurstof achterlaten bij het betreffende vakje. De kans dat mieren uit een volgende groep het vakje wel of niet kleuren wordt beïnvloed door deze geurstoffen.
Voor ieder vakje zijn er getallen *w* en *z*. Hiervoor geldt altijd dat $w + z = 400$. Aan het begin zijn *w* en *z* beiden 200. De kans op het kleuren van een vakje door een mier is $\frac{z}{z + w}$ en net zo kun je de kans op het leeglaten van een vakje bepalen.
Aan het eind van een serie van 16 worden de resultaten bijgewerkt. Iedere mier heeft een afbeelding met een fitheid *f* gevonden. Voor ieder vakje wordt bij *w* of *z* nu de waarde *f* opgeteld. In totaal kunnen er maximaal 16 waarden bij een *w* of *z* worden opgeteld. Gevolg is uiteraard dat $w + z$ nu groter is dan 400. Voor ieder vakje geldt dat de verhouding tussen *w* en *z* nu zo veel mogelijk gehandhaafd blijft, maar dat $w + z$ weer gelijk wordt aan 400.

Vraag 28: Geef formules om de nieuwe waardes van z en w te bepalen uit de oude.

Ook op deze manier komen na verloop van tijd steeds hogere fitheidwaardes te voorschijn.

Antwoorden bij de vragen.

Vraag 1.

16 = 2 x 2 x 2 x 2

Vraag 2.

2 x 2 x 2 x … x 2 met 25 keer een 2.
Dit is 2^{25} en dat is 33.554.432

Vraag 3.

2^{NM}.

Vraag 4.

10^{125} is ongeveer 2^{415}.
Als een puzzel meer dan 415 vakjes heeft zijn er meer dan 10^{125} verschillende invullingen.

Vraag 5.

N	Omschrijvingen	Aantal
1	- 1	2
2	- 1 2	3
3	- 1 2 3 1 1	5
4	- 1 2 3 4 1 1 1 2 2 1	8
5	- 1 2 3 4 5 1 1 1 2 1 3 2 1 2 2 3 1 1 1 1	13

Vraag 6.

Dit zijn getallen uit de rij van Fibonacci. Als je meer wilt weten over hem en zijn rij, zijn er legio sites op Internet te vinden.

Vraag 7.

O(N) = O(N-1) + O(N-2).
Waarom is dit het geval?
Alle omschrijvingen van een regel van N-1 zijn ook bruikbaar voor een regel van lengte N.
Dit verklaart het aandeel van O(N-1).
Wat nog overblijft zijn alle omschrijvingen waar geen overbodige witte vakjes bijhoren. Deze kun je krijgen door alle mogelijke omschrijvingen van een regel van N-2 vakjes te nemen en die rechts aan te vullen met zonodig een wit vakje en dan zoveel mogelijk zwarte. Bekijk maar eens de omschrijvingen voor een regel van 3 vakjes:

Mogelijkheden voor 3 vakjes	Mogelijkheden zonder extra wit voor 5 vakjes
0	5
1	1 3
2	2 2
3	3 1
1 1	1 1 1

Het aantal extra omschrijvingen is daarom O(N-2).

Vraag 8.

Er zijn twee groepen zwarte vakjes (inclusief een verplicht wit tussenvakje) en twee vrije witte vakjes. Je zoekt dus het aantal manieren waarop je die twee witte vakjes kunt plaatsen als je deze vier zaken op een rij zet. Dat kan op $\binom{4}{2}$ manieren of 6 manieren (zie de paragraaf over het oplossen van Japanse puzzels voor een overzicht van deze zes mogelijkheden).

Vraag 9.

Zie de formule uit vraag 10.

Vraag 10.

In vraag 8 heb je de redenering al gezien. Het aantal groepen dat een plek krijgt is M. Het aantal vrij te plaatsen witte vakjes is afhankelijk van vele zaken. Er zijn N vakjes. Z ervan zijn zwart, dat wil zeggen er zijn N-Z wit. Maar de M groepen veroorzaken dat er M-1 witte vakjes verplicht zijn. Er zijn dus nog N-Z-M+1 witte vakjes vrij te kiezen. Het aantal zaken dat op een rij moet worden gezet is M zwarte groepen en N-Z-M+1 vrije witte vakjes, samen N-Z+1. Het aantal mogelijkheden wordt daarom:

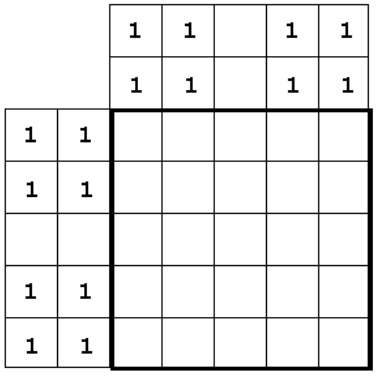
$$\binom{N - Z + 1}{N - Z - M + 1}$$

Vraag 11.

Er zijn 15 verschillende omschrijvingen (zie afbeelding 3).

Vraag 12.

Maak een onderverdeling, waarbij in alle hoeken een niet-unieke 2 bij 2 puzzel zit. Voor iedere hoek zijn twee mogelijkheden, dus in totaal zijn er 16.



Afbeelding 12. Een mogelijke oplossing van vraag 12.

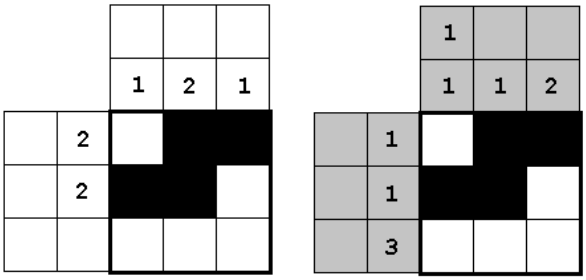
Vraag 13.

N-Z

Vraag 14.

Minstens M-1, hoogstens M+1.
M-1 als beide uiteinden een zwart vakje hebben.
M+1 als beide uiteinden een wit vakje hebben.
M als de beide uiteinden verschillend gekleurd zijn.

Vraag 15.



Afbeelding 13. Links een niet-unieke puzzel, rechts een uniek negatief.

Vraag 16.

Het antwoord is afhankelijk van het gevonden programma. Zie de antwoorden op vraag 17 voor punten die een rol zouden kunnen spelen.

Vraag 17.

De volgende lijst is niet uitputtend.

- Is het de gebruiker duidelijk welke mogelijkheden het programma biedt?
- Bevat het programma een duidelijke hulpfunctie?
- Is een beschrijving van de manier waarop Japanse puzzels worden opgelost beschikbaar?
- Kan het uiterlijk van het programma aan de wensen van de gebruiker worden aangepast?
- Is het duidelijk hoe vakjes moeten worden gekleurd?
- Kan de kleuring van een vakje weer gemakkelijk ongedaan worden gemaakt?
- Kan een groep vakjes worden geselecteerd om zo snel alle vakjes ervan te kleuren?
- Geeft het programma desgevraagd hints? Met uitleg?
- Is er de mogelijkheid de gevonden oplossing te controleren?
- Zo ja, gebeurt deze controle op basis van de afbeelding of van de omschrijving?
- Houdt het programma een vorm van score bij, zoals de verstreken tijd?

- Worden bijgehouden scores getoond in de vorm van een top tien of iets dergelijks?
- Kan een tussentijdse oplossing worden weggeschreven en weer worden ingelezen?
- Kunnen er puzzels aan het programma worden toegevoegd?

Vraag 18.

In Java is het
```
bitnummer % 5
bitnummer \ 5
```

In Visual Basic
```
bitnummer Mod 5
bitnummer \ 5
```

Vraag 19.

In Pascal:
```
rijnummer   := 1 + ((bitnummer -1) mod 5)
kolomnummer := 1 + ((bitnummer-1) div 5)
```

In andere talen vergelijkbaar.

Vraag 20.

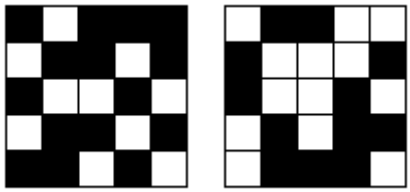
Kleurcode	Binair	Betekenis	Resultaat
7	0111	Rood Groen Blauw	Grijs
9	1001	Helder Blauw	Blauw
10	1010	Helder Groen	Groen
11	1011	Helder Groen Blauw	Cyaan
12	1100	Helder Rood	Rood
13	1101	Helder Rood Blauw	Paars
14	1110	Helder Rood Groen	Geel
15	1111	Helder Rood Groen Blauw	Wit

Vraag 21.

Dit hangt uiteraard af van de snelheid van je computer. Op mijn Pentium II, 350 MHz, worden in een minuut ongeveer 150.000 afbeeldingen onderzocht; alles doorzoeken duurt dan bijna vier uur.

Vraag 22.

Hieronder een voorbeeld. Teller bij linkerafbeelding 25, bij die van rechts 61.



De figuur links heeft meer gekleurde vakjes (15 tegen 12), daardoor ligt van veel regels de invulling al vast (eerste horizontaal en eerste en vierde verticaal). Bij de rechterfiguur liggen nog maar twee vakjes vast (zie ook de volgende opgave).

Vraag 23.

Omschrijving	Bekend
0	ooooo
1	?????
2	?????
3	??X??
4	?XXX?
5	XXXXX

1 1	?????
1 2	???X?
1 3	XoXXX
2 1	?X???
2 2	XXoXX
3 1	XXXoX
1 1 1	XoXoX

Vraag 24.

33554432 is te schrijven als 4096 * 8192.
Daarom gaat het bijvoorbeeld goed met `random(4096)*8192 + random(8192)`

Vraag 25.

Bij simulated annealing wordt, vanuit de huidige afbeelding, één van de buren onderzocht. Heeft die een hogere fitheidwaarde, dan wordt dat de nieuwe afbeelding. Als de fitheidwaarde kleiner of gelijk is aan de huidige, bepaalt een toevalsproces of deze buurafbeelding het nieuwe uitgangspunt wordt. De kans daarop wordt kleiner als de fitheid minder wordt; bovendien neemt de kans ook af naarmate het zoekproces langer duurt.
Bij tabu search wordt net als bij hill climbing de hele omgeving doorzocht. De beste afbeelding in de omgeving wordt het nieuwe uitgangspunt; maar er is een lijst met afbeeldingen die dat recentelijk al eens geweest zijn. Deze mogen niet te snel weer aan de beurt komen, omdat je op voldoende verschillende plaatsen moet zoeken.

Vraag 26.

Er veranderen twee regels, een rij en een kolom. Acht regels blijven onveranderd. De fitheid verandert dus hoogstens 2 ten opzichte van de oorspronkelijke waarde.

Vraag 27.

2^{25-N}.

Vraag 28.

$$w_{nieuw} = round(\frac{w_{oud} * 400}{w_{oud} + z_{oud}})$$

Bijlagen.

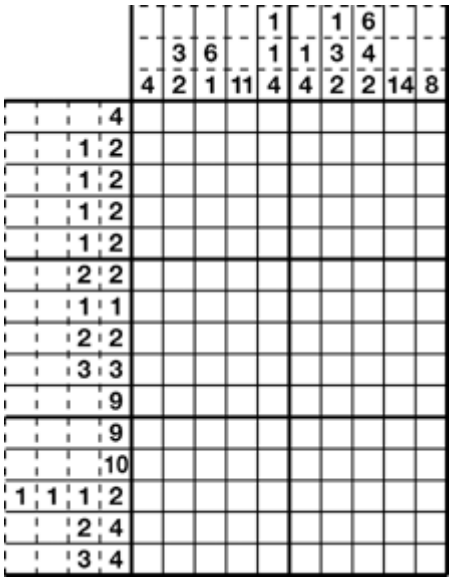
Bijlage 1. Uitleg van Puzzelsport.

http://www.puzzelsport.nl/uitleg/japans.html

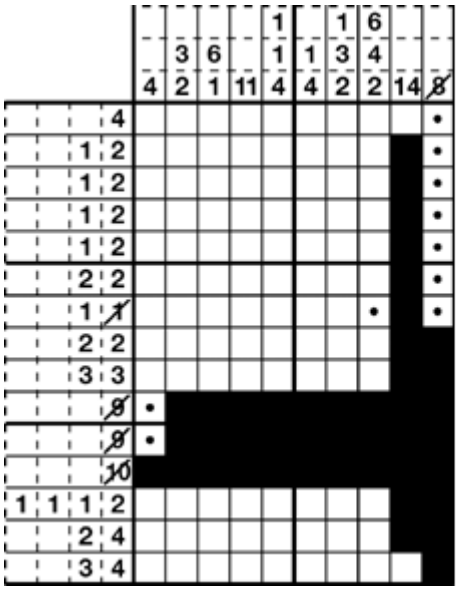
Japanse Puzzels, logisch tekenen

Zo los je een Japanse puzzel op:

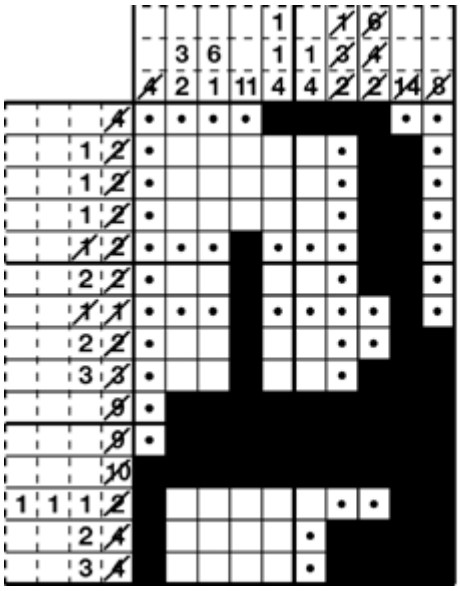
Bij Japanse puzzels kun je door logisch na te denken een tekening reconstrueren. De cijfers naast en boven het diagram geven aan hoeveel vakjes er in de betreffende rij of kolom moeten worden ingekleurd. Staat er bijvoorbeeld 3-2 dan betekent dat dat er na eventueel een of meer witte (lege) vakjes eerst drie vakjes achtereen gekleurd zijn; daarna volgen er een of meer witte vakjes, dan twee gekleurde vakjes en ten slotte eventueel weer een of meer witte vakjes. Een voorbeeld.



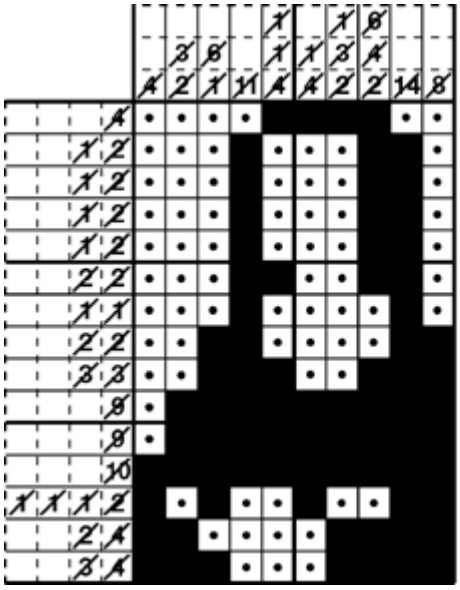
Links van de twaalfde regel staat een 10. Dat betekent dat alle tien vakjes op die regel gekleurd worden. Op elk van de twee regels daarboven worden negen vakjes gekleurd. Van de tien beschikbare vakjes kunnen er dus al acht worden ingekleurd; alleen de buitenste vakjes blijven nog even leeg. Hetzelfde geldt voor de negende kolom, die met 14. Links van de zevende regel staat 1-1. Dat betekent dat het gekleurde vakje op die regel een losstaand vakje moet zijn; de vakjes links en rechts van dat gekleurde vakje blijven leeg. Om dat aan te geven zet je een stip in die vakjes. In de laatste kolom, die met de 8, staat nu een stip en de acht aaneengesloten gekleurde vakjes passen er nog maar op één manier in. Ook de twee rijen met 9 zijn daarmee bekend.



De volgende stap is de achtste kolom, die met 6-4-2. Deze kolom kan nog maar op één manier worden ingevuld. Ook is het duidelijk waar de vier gekleurde vakjes in de eerste kolom komen. Wanneer we nu kijken naar de rijen 2 t/m 6, dan zien we dat we de meest rechter blokjes van twee achter elkaar gekleurde vakjes al hebben ingevuld. De zevende kolom is daarmee compleet. De middelste acht van de 11 vakjes van de vierde kolom liggen nu ook vast.

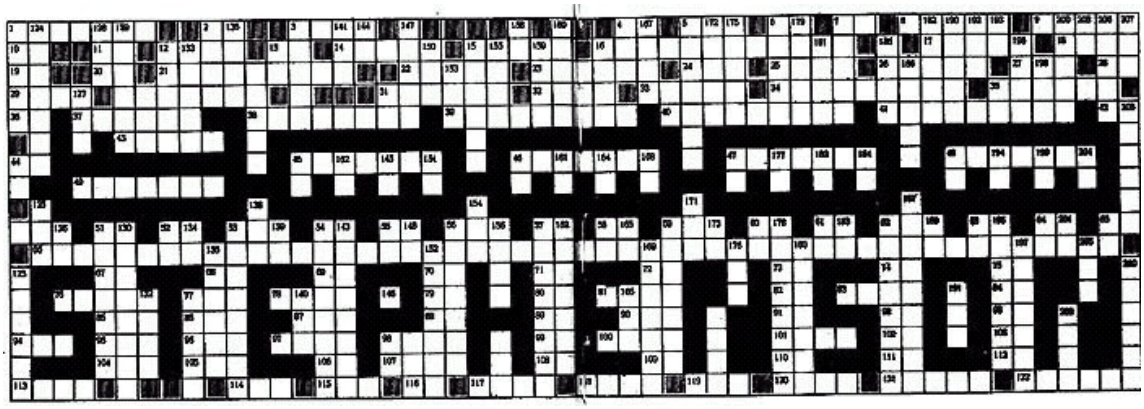


Vanaf hier kan het best verder gegaan worden met de hoek linksonder. Door de overige kolommen en rijen met elkaar te vergelijken kan de tekening worden afgemaakt.



Alle puzzels hebben een titel die verwijst naar de tekening die ontstaat. Dit voorbeeld heet 'Da Vinci's vrouw', omdat het plaatje de Mona Lisa van Leonardo da Vinci voorstelt.

Bijlage 2. De oorsprong van beeldzoekers.



Roosters met afbeeldingen worden al vele jaren in puzzels gebruikt. Vroege kruiswoordpuzzels maakten soms gebruik van afbeeldingen als onderdeel van hun lay-out; de Italiaanse puzzel Stephenson's Train uit 1925 is daar een fraai voorbeeld van. Ook het gebruik van roosters met aanwijzingen om de inhoud te achterhalen heeft een lange geschiedenis; Trevor Truran's "Whittleword" van het tijdschrift Games & Puzzles Magazine, Issue 73 van de zomer van 1979 is er een voorbeeld van. Het type beeldzoeker is nieuw in die zin dat de informatie bij het rooster wordt gebruikt om de afbeelding op het rooster te berekenen.
In 1987 won de grafisch ontwerpster Non Ishida een wedstrijd in Tokio om een afbeelding te maken door bepaalde lichten van een wolkenkrabber aan en uit te zetten. Dit bracht haar op het idee voor een puzzel waarbij een deel van de vakjes in het rooster moest worden gekleurd. In 1988 publiceerde ze drie van deze puzzels in Japan onder de naam "Window Art Puzzles". Tetsuya Nishio werkte in dezelfde periode onafhankelijk van haar aan hetzelfde idee en publiceerde dit type puzzels in een ander tijdschrift.
In 1989 toonde Non Ishida haar "Window Art Puzzles" aan James Dalgety. James vond het geweldige puzzels en hij sprak met Non Ishida af om haar puzzels buiten Japan op de markt te brengen. In 1990 bedacht James Dalgety de naam Nonogram naar "Non" Ishida en Dia"gram". Hij haalde de grootste Britse kwaliteitskrant The Telegraph over om de puzzels wekelijks te publiceren in hun zondagsuitgave. Dit gebeurt nog steeds.
In 1993 werden de Nonogrammen, meestal van de hand van Non Ishida, uit de Telegraph overgenomen door Mainich (één van de grootste kranten in Japan). In dat jaar werd ook het eerste boek met Nonogrammen door Non Ishida uitgebracht in Japan; later in dat jaar werd "The Sunday Telegraph Book of Nonograms" in Engeland gepubliceerd door Pan Books. Nonogrammen verschenen in Zweden, de Verenigde Staten, Zuid Afrika en andere landen.
In 1995 bracht Pan Books het vierde Sunday Telegraph Book of Nonograms uit. Non Ishida wilde de naam Nonogram uitsluitend voor haar eigen ontwerpen in Japan gebruiken en de samenwerking met James Dalgety werd in 1996 beëindigd In de zomer van 1998 organiseerde The Sunday Telegraph een prijsvraag voor een eigen naam voor deze puzzels; **Griddler** was de winnende naam..

In de herfst van 1999 bracht The Sunday Telegraph het eerste **Griddler Book uit** en sindsdien verschijnt er één per jaar. In 2002 is de puzzel in deze vorm elf jaar oud; puzzels van dit type verschijnen als elektronisch speelgoed, als kunststof puzzels, en alleen al in Japan zijn er meer dan 10 tijdschriften en vele boeken over dit onderwerp. The Sunday Telegraph was de eerste krant die de mogelijkheden onderkende van dit type puzzel en de puzzels wekelijks publiceerde, al meer dan elf jaar; langer dan ieder ander.

GRIDDLERS en COMPUTERS.
Vanaf het begin hebben liefhebbers, docenten en studenten ontdekt dat het schrijven van computerprogramma's om deze puzzels op te lossen een fascinerende bezigheid is. Ieder aspect van programmeren komt er bij kijken: Werken met bestanden, graphics, data opslag, printen en uiteraard vooral algoritmes. Discussies over dat programmeren, Java applets met oplossingen, en vele zaken meer, kunnen op veel plaatsen op het World Wide Web worden gevonden; zoek op namen en merknamen als CrossPix, Descarte's Enigma, FigurePic, Griddler, IllustLogic, Nonograms, Oekaki-Logic, Paint by Numbers, Picross, PictureLogic, and StarPic.

James Dalgety en Bill Stanton schreven misschien wel het eerste programma dat puzzels kon oplossen in 1990. Merkwaardigerwijs was het zo ontworpen dat het alleen maar menselijke logica gebruikte. Latere programma's konden veel ingewikkelder puzzels oplossen, maar het oorspronkelijke algoritme wordt nog steeds gebruikt om alle puzzels voor The Sunday Telegraph te controleren. Puzzelaars moeten er altijd op bedacht zijn dat sommige ontwerpers puzzels publiceren die niet uitsluitend door gebruik te maken van logica zijn op te lossen. Bijvoorbeeld omdat je moet raden, zeer ingewikkelde aannames moet doen of gebruik moet maken van kennis over de afbeelding. James en zijn team van puzzelontwerpers proberen zich er altijd van te verzekeren dat hoe moeilijk de puzzel ook mag zijn, deze altijd kan worden opgelost door logisch te redeneren; hoe moeilijk dat soms ook is te realiseren, het staat garant voor het meeste puzzelplezier.

Bijlage 3. Opgave 1 Internationale Informatica Olympiade Bonn 1992.

```
TASK 4.1.3 "ISLANDS IN THE SEA"
===============================
The SEA is represented by an N times N grid. Each ISLAND is a "*" on
that grid. The task is to reconstruct a MAP of islands only from some
CODED INFORMATION about the horizontal and vertical distribution of
the islands. To illustrate this code, consider the following map:

*   * *       1 2
  * * *   *   3 1
*   *   *     1 1 1
  * * * * *   5
* *   *   *   2 1 1
      *       1

1 1 4 2 2 1
1 2   3   2
1
```

The numbers on the right of each row represent the order and size of the groups of islands in that rows. For example, "1 2" in the first row means that this row contains a group of one island followed by a group of two islands; with sea of arbitrary length to the left and right of each island group. Similarly, the sequence "1 1 1" below the first column means that this column contains three groups with one island each, etc.

PROBLEM STATEMENT
=================
Implement a program which repeats the following steps until a given input file containing several information blocks has been read completely:

1. Read the next information block from an ASCII input file
   (for the data structure of that file see also the examples below)
   and display it on the screen.
   Each information block consists of the size of the square grid,
   followed by the row constraints and the column constraints. Each
   constraint for a single row or column appears on a single line as
   a sequence of numbers separated by spaces and terminated by 0.
2. Reconstruct the map (or all of the maps, if more then one solution
   is possible, see Example-4) and display it/them on the screen.
3. Write the map(s) to the end of an ASCII output file. Each blank
   must be represented by a pair of spaces. Each island should be
   represented by a '*' followed by a space. Different maps satisfying
   the same constraints should be separated by a blank line. If there
   is no map satisfying the constraints, indicate it by a line saying
   "no map". The solutions to the different information blocks must be
   separated by a line saying "next problem".

```
TECHNICAL CONSTRAINTS
=====================
Constraint-1: N must be not less than 1 and not larger than 8.
Constraint-2: Put your solution program into an ASCII text file named
              "C:\IOI\DAY-1\413-PROG.xxx". Extension .xxx is:
              - .BAS for BASIC programs, .C   for C      programs,
              - .LCN for LOGO  programs, .PAS for PASCAL programs.
Constraint-3: The name of the ASCII input file for reading the coded
              information from must be "C:\IOI\DAY-1\413-SEAS.IN".
Constraint-4: The name of the ASCII output file for writing the
              map(s) to must be "C:\IOI\DAY-1\413-SEAS.OU".

EXAMPLE(S)
==========
6             Example-1 (the problem above): 6 is the size of the grid.
1 2 0         <-- The start of the first line constraint
3 1 0
1 1 1 0
5 0
2 1 1 0
1 0
1 1 1 0       <-- The start of the first column constraint
1 2 0
4 0
2 3 0
2 0
1 2 0

4             Example-2. Solution: columns: 1 2 3 4
0                                  row 1:
1 0                                row 2:     *
2 0                                row 3:   * *
0                                  row 4:
0
1 0
2 0
0

2             Example-3. Note that there is no map
0                        satisfying the constraints.
0
2 0
2 0

2             Example-4. Note that there are two different maps
1 0                      satisfying the constraints.
1 0
1 0
1 0
```

SAMPLE FILES
============
We provided these correct example files for your convenience:
"C:\IOI\DAY-1\413-SEAS.IN" and "C:\IOI\DAY-1\413-SEAS.OU".

WARNING: Successful execution of your program with these examples
does not necessarily guarantee that your program is correct !!!

CREDITS
=======
Read an information block from
the input file and display it .... 5 points
Process all information blocks one by one
until the input file is read completely .... 10 points
Reconstruct one map for each information
block (if it has a solution) and display it .... 35 points
Write the solution map to the output file .... 5 points
Reconstruct all possible maps (if there
are several solutions) and display them .... 20 points
Write all solution maps correctly
separated to the output file .... 10 points
Identify information blocks having no solution .... 5 points
Technical constraints completely obeyed .... 10 points
----
                                                  maximal 100 points

Bijlage 4. Japanse puzzels via een genetische algoritme.

Dit is een opgave van Walter Kosters van de Universiteit van Leiden.

Schrijf een genetisch algoritme voor het oplossen van Japanse puzzels. Zie ook hieronder voor meer uitleg. Beoogde lengte van het programma: ruim tweehonderd regels.
Neem aan dat we een klein vierkant, zeg een 5-bij-5 array hebben, waarvoor een "Japanse puzzel" gespecificeerd is. Oplossingen zijn strings met 25 bits die aan de eisen voldoen - leg de rijen van het array maar achter elkaar; de oplossingen hoeven overigens niet altijd uniek te zijn. De fitnessfunctie is bijvoorbeeld het aantal kolommen en rijen dat precies klopt met de specificatie, of iets beters. Gebruik mutatie en crossover; je kunt bijvoorbeeld crossover doen met de twee "fitste" strings uit de populatie, of met de twee "fitste" van een tiental random getrokken strings uit de populatie, of met twee willekeurige strings. Redelijke waarden zijn: mutatiekans (per bit) 0.01, populatiegrootte 100, 5000 generaties. Neem bijvoorbeeld de 5 beste steeds mee naar de volgende generatie (elitair).

Japanse puzzels:

```
5 5
0
1 1
0
1 1
3
1
1 1
1
1 1
1
```

Uitleg aan de hand van bovenstaand voorbeeld. De puzzel heeft 5 rijen en 5 kolommen, en dus 25 vakjes, die wit of zwart zijn. In de eerste rij staan 0 zwarte vakjes, oftewel: alle vakjes zijn wit - die 0 staat dus in de tweede regel van deze file. In de tweede rij staan (ergens) 1 los zwart vakje en nog 1 los zwart vakje - de cijfers uit de derde regel van deze file. In de vijfde rij staan 3 zwarte vakjes direct naast elkaar. In de eerste kolom staat 1 los zwart vakje - dat is dus de 1 uit de zevende regel van deze file. In het algemeen betekent a_1,a_2,...,a_n dat er -in die volgorde- in de betreffende rij of kolom eerst een x-tal (eventueel 0) witte vakjes zijn, dan aaneengesloten a_1 zwarte, dan een y-tal witte (minstens EEN), dan aaneengesloten a_2 zwarte, dan weer ..., en tot slot aaneengesloten a_n zwarte, en dan nog z witte (eventueel 0). Je kunt overigens deze file letterlijk als voorbeeldfile nemen; doordat de eerste regel van de file de dimensies onthult weet je waar het "commentaar" begint. Bovenstaande voorbeeldpuzzel is in wellicht wat meer bekende notatie:

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

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

Voorbeeld-opgave met oplossing.

Bijlage 5. Op mieren rekenen

De Financieel-Economische Tijd, Zaterdag 14 september 2002

Sociale insecten zoals mieren komen op basis van een stel eenvoudige regels tot doeltreffende collectieve prestaties. De kolonie is intelligenter dan de afzonderlijke individuen die er deel van uit maken, met als resultaat een robuust en flexibel geheel. Computerwetenschappers en ingenieurs zijn erin geslaagd deze prehistorische strategie te vertalen naar nuttige hedendaagse toepassingen op het gebied van dataverkeer, logistiek en distributie. En mieren hebben wellicht nog meer in petto. Dit weekend vergaderen de mierenrekenaars in Brussel.

Kris De Decker

De ijver en kracht van mieren is bekend, maar de manier waarop ze op zoek gaan naar voedsel in minstens even opmerkelijk. Deze oeroude insecten communiceren niet rechtstreeks met elkaar, maar via de omgeving. Dat doen ze door het afscheiden van feromonen, geursporen die slechts langzaam weer verdwijnen. Deze vorm van indirecte communicatie berust op twee eenvoudige gegevens : elke mier zet feromonen af terwijl ze zich voortbeweegt en elke mier verkiest het spoor te volgen dat het rijkst is aan deze geurstoffen. Daarom vinden mieren altijd hun weg tussen een voedselbron en het nest en daarom gaan mieren na een tijd allemaal via dezelfde, kortste route wandelen. Op deze manier kunnen ze zich ook aanpassen aan veranderingen in die omgeving, als er bijvoorbeeld een obstakel op hun weg terechtkomt. Aangezien het feromoonspoor dan onderbroken wordt, zal een deel van de mieren linksom gaan, terwijl een ander deel langs rechts het obstakel probeert te omzeilen. De insecten die per toeval de kortste weg kiezen, zullen eerder het onderbroken pad herstellen. Bijgevolg ontvangt het kortere pad meer feromonen in dezelfde tijdspanne, wat er op zijn beurt toe leidt dat meer en meer mieren dat pad zullen volgen. Het is de kolonie die in staat is om steeds weer de kortste weg te vinden, niet de afzonderlijke mieren.

Rekenkracht.

Deze manier van communiceren, realiseerde Marco Dorigo zich een decennium geleden, kan ook van nut zijn voor het oplossen van combinatorische optimaliseringsproblemen, zoals het handelsreizigersprobleem. Bij dit bekende wiskundige vraagstuk is het de bedoeling de kortste weg te vinden tussen een gegeven aantal steden, terwijl je elke stad maar één keer mag bezoeken en waarbij je weer moet aankomen in de stad waar je vertrokken bent. Professor Marco Dorigo, nu verbonden aan de Université Libre de Bruxelles (ULB) maar toen nog student kunstmatige intelligentie in Milaan, had meteen een onderwerp voor zijn doctoraat. De grondlegger van het rekenen met mieren stuurde zijn revolutionaire idee ook naar een gereputeerd vakblad voor ingenieurs, waar het meer dan vijf jaar bleef liggen. Maar toen het uiteindelijk werd gepubliceerd, ging de bal aan het rollen. Dit weekend zitten tientallen biologen en ingenieurs uit de gehele wereld bij elkaar in Brussel voor de conferentie Ants2002. De tweejaarlijkse internationale workshop, voorgezeten door Dorigo, vindt al voor de derde keer plaats.
Het handelsreizigersprobleem is van het type ‘NP-hard’ (NP-moeilijk), een wiskundige term die erop duidt dat de tijd die nodig is om de optimale oplossing te vinden, exponentieel groeit met de omvang

van het probleem. De berekening loopt dan ook al gauw stuk op de beschikbare rekenkracht van de computer. Om de optimale oplossing te vinden, moet je alle mogelijkheden verkennen. Dorigo : 'Afhankelijk van de rekenkracht vraagt het bijvoorbeeld 1 uur computertijd om het probleem op te lossen met 10 steden. Wil je dezelfde berekening doen met 11 steden, dan heb je 10 uur rekentijd nodig. Met 12 steden vraagt de berekening al honderd uur tijd. Stel dat je een algoritme ontwikkelt dat dit probleem oplost met 10.000 steden : datzelfde algoritme is waardeloos als je dezelfde berekening wil maken met 10.005 steden.'
Er is een andere mogelijkheid, weet Dorigo : 'In plaats van twintig jaar lang op de absoluut optimale oplossing te wachten, kun je beter in twintig seconden verschillende goede oplossingen bekomen.'
Software gebaseerd op de communicatie bij mieren blijkt daar erg goed in te zijn. In plaats van alle mogelijke routes te berekenen, laat het programma een hoop virtuele mieren los, die vervolgens zelf hun weg zoeken door het probleem heen. De software imiteert de eenvoudige gedragsregels op basis van feromonen. Ook deze abstracte insecten laten denkbeeldige geursporen achter op hun pad en ze volgen het liefst die routes die het sterkst geuren. De artificiële mieren kregen ook karakteristieken mee die echte mieren niet hebben, zoals een geheugen en sneller verdampende feromonen, wat veel betere algoritmes opleverde. Dankzij het geheugen kunnen ze de hoeveelheid achtergelaten geurstoffen laten afhangen van de kwaliteit van de gevolgde route. De keuze van een volgende stad wordt bepaald door de hoeveelheid feromonen op de verschillende routes. 'Goede routes worden zo versterkt en op die manier komt vrij snel een oplossing naar boven, die waarschijnlijk wel niet de beste, maar wel een goede oplossing is', zegt Dorigo.
De mierenalgoritmes werden oorspronkelijk uitgedacht als een benaderende oplossing voor het handelreizigersprobleem, maar dezelfde methodologie – Ant Colony Optimization (ACO) – wordt nu ook met succes toegepast op andere optimaliseringsproblemen. Het handelsreizigersprobleem kent vele varianten, die allemaal NP-moeilijk zijn. Dorigo : 'Voor het handelreizigersprobleem kan je met een zeer krachtige computer op een redelijke tijd, zeg een dag, een optimale oplossing vinden voor 20.000 steden. Maar voor andere, zoals het Quadratic Assignment Problem, kan dat maar voor 50 items.' Bij dit probleem moet je een gegeven aantal items (bijvoorbeeld fabrieken) in een even groot aantal locaties plaatsen. Tussen de fabrieken vindt een uitwisseling plaats, maar de hoeveelheid varieert. Waar plaats je dus welke fabrieken opdat je de materialen over zo'n kort mogelijke afstand moet vervoeren ?
Nog lastiger is het Vehicle Routing Problem. Elke knoop stelt bijvoorbeeld een klant voor die op de bus wacht. Eén knoop is het busdepot, waar een aantal bussen klaar staan om te vertrekken. Het is de bedoeling de routes van de bussen zodanig uit te stippelen dat de totaal afgelegde reisafstand minimaal is, dat elke bus maar zoveel passagiers ophaalt als zijn maximum capaciteit toelaat, dat er zo weinig mogelijk bussen nodig zijn, dat de lengte van de route van elke bus onder een gegeven maximum blijft én dat alle bussen op het einde van hun route terug in het depot aankomen. Een ingewikkelder variant houdt ook rekening met een bepaalde tijdspanne waarbinnen een klant opgehaald moet worden.
Logistiek combinatorische optimaliseringsproblemen doen zich voor in de logistiek, distributie, catering, bij het opmaken van tijdstabellen of bij het plannen van distributie of productie. Het Zwiterse bedrijf AntOptima, dat ook de conferentie sponsort, heeft de mierenalgoritmes intussen gecommercialiseerd om logistieke problemen op te lossen. Het gaat om benaderende toepassingen van het Vehicle Routing Problem en het Sequential Ordering Problem. Die mierensoftware wordt al gebruikt door de Italiaanse oliemaatschappij Agip en door de Zwitserse distributieketen Migros.
Hoewel ze dus erg verschillende toepassingen hebben, zijn al deze probleemstellingen op wiskundig vlak gelijk. Het gaat om problemen die herleidbaar zijn tot het zoeken in een graaf, dat is een abstracte wiskundige figuur die bestaat uit een aantal punten of knopen, verbonden door lijnen of bogen. Een graaf kan statisch zijn, zoals in de vermelde probleemstellingen ; de afstand tussen de 'steden' of knooppunten blijft immers altijd gelijk. Maar de graaf kan ook dynamisch zijn. En zo stuiten de mierenrekenaars op nieuwe toepassingen. Het probleem van routing – het rondsturen van datapakketjes in netwerken – kan worden herleid tot het zoeken van het kortste pad in een dynamische graaf. Ook daarvoor is een methodologie ontworpen : Ant Colony Routing (ACR). 'De mierenalgoritmes blijken ook uitermate geschikt te zijn voor het controleren van pakketgeschakelde netwerken zoals internet', zegt Dorigo. 'Maar er zijn nog geen toepassingen. Daar is de techniek nog niet klaar voor. Er is wel interesse van netwerkbouwers.'
Er zijn nog veel meer algoritmes die geïnspireerd zijn door mieren en door andere sociale insecten, maar die zitten nog in de fase van fundamenteel onderzoek. Als groep worden ze ook aangeduid met de term 'swarm intelligence'. Dorigo en zijn medewerkers doen bijvoorbeeld ook onderzoek naar het besturen van modulaire robots, die van nut kunnen zijn bij reddingsoperaties of rampen. 'Het idee is een groep robots te laten samenwerken om problemen op te lossen, ook hier weer zonder centrale

controle. Ook in dit geval blijkt de mierenkolonie een voorbeeld te zijn : "Mieren bouwen clusters van larven. Ze volgen daarbij zeer eenvoudige regels, bijvoorbeeld : als ik geen larf draag en ik vind er één, dan til ik ze op. Maar ligt ze dichtbij andere larven, dan laat ik ze liggen. En als ik een larve draag, leg ik ze neer bij andere larven. Zie ik geen andere larven, dan blijf ik de larve dragen. We proberen de robots te laten clusteren, geïnspireerd door deze eenvoudige regels."
Ook de werkverdeling zorgt voor inspiratie. Dorigo : "Elke mier kan alle mogelijke taken uitvoeren, maar voor elke taak hebben ze een bepaalde ingebouwde drempel. Dat blijkt bijvoorbeeld uit een experiment met werkers en soldaten. Normaal gezien houden de soldaten zich niet bezig met larven. Zij hebben dus een hoge drempel voor deze stimulus, in dit geval de geur die de kolonie verspreidt wanneer de larven aandacht nodig hebben. Maar zodra je werkers gaat verwijderen, gaat die stimulus omhoog en op een bepaald punt, wanneer je voldoende werkers hebt verwijderd, beginnen sommige soldaten hun taak over te nemen. Verwijder je alle werkers, dan doen de soldaten al het werk. Het enige verschil is dat ze er iets later aan beginnen, omdat de drempel voor die bepaalde stimulus hoger is. Ook dit model zou je op robots kunnen toepassen."

Rekenen met mieren is niet de enige oplossing.

Er bestaan tal van andere andere technieken voor het oplossen van ‘NP-moeilijke’ problemen die gebouwd zijn rond enkele basisprincipes ontleend aan een natuurlijk fenomeen. Zo wordt ook gewerkt met neurale netwerken, waarbij de oplossing wordt gezocht via de werking van de hersenen. De mierenalgoritmes scoren het best voor enkele problemen, maar moeten de duimen leggen bij de andere.

Bijlage 6. Opgave 1 eerste ronde N.I.O. 2002-2003.

Opgave 1. Japanse beeldzoekers.

Een Japanse beeldzoeker is een puzzel waarbij de oplossing bestaat uit een rechthoek met zwarte en witte vierkantjes. Bij iedere rij en iedere kolom staat een aantal getallen aangegeven die vastleggen hoeveel aanliggende zwarte vierkantjes er te vinden zijn. Als er bij een rij of kolom staat '4 3 1' betekent dat dat er eerst een groep van vier aaneengesloten zwarte vierkantjes is in de oplossing, dan een groep van drie en dan nog een los zwart vierkantje. De groepen worden gescheiden door minstens één wit vierkantje. Voor de eerste groep kan eventueel een groep voorkomen van witte vierkantjes; achter de laatste zwarte groep is dat ook het geval.

			1	1	1	1	1		2	3			1
			1	3	2	1	1	5	3	1	7	7	4
		8	■	■	■	■	■	■	■	■			
		5						■	■	■	■	■	
	3	4				■	■	■		■	■	■	■
	2	2						■	■		■	■	
2	2	3	■	■				■	■		■	■	■
	1	5		■					■	■	■	■	■
	2	3		■	■						■	■	■
	1	3			■						■	■	■

Voor meer over Japanse beeldzoekers zie: http://www.puzzelsport.nl/uitleg/japans.html
In deze opgave ga je een programma schrijven waarmee je bij een figuur van zwarte en witte vakjes een omschrijving gaat geven van de puzzel. Je programma zoekt voor iedere rij en iedere kolom de getallen die de groepjes zwarte vakjes aangeven.

Opdracht:

Schrijf een programma **`nio1`**. Invoer is een bestand **`puzzel.in`**. Het bestand bestaat uit eerst twee regels met een getal, daarna wordt het patroon van de gegeven figuur vastgelegd. Op de eerste regel staat een getal r met $0 < r < 31$, dat staat voor het aantal rijen met vakjes. Op de tweede regel staat een getal k met $0 < k < 31$ dat staat voor het aantal kolommen met vakjes. Vervolgens komen er r regels van elk k letters 'Z' of 'W', waarbij de Z staat voor een zwart vakje en de W voor een wit vakje.
Uitvoer is een bestand **`puzzel.uit`** van $r + k$ regels. De eerste r regels bevatten informatie over de rijen van de figuur; de k regels erna over de kolommen. Telkens staat aangegeven: Eerst het aantal g aan groepjes met zwarte vakjes op de betreffende rij of kolom, vervolgens g getallen met de aantallen zwarte vakjes in de opeenvolgende groepen (zie het voorbeeld, dat hoort bij de afbeelding hier boven). Getallen worden gescheiden door spaties.
De tijdlimiet voor deze opgave is 5 seconden.

Voorbeeld:

puzzel.in

```
8
11
ZZZZZZZZWWW
WWWWWZZZZZW
WWWZZZWZZZZ
WWWWWZZWZZW
ZZWWWZZWZZZ
WZWWWWZZZZZ
WZZWWWWWZZZ
WWZWWWWWZZZ
```

puzzel.uit

```
1 8
1 5
2 3 4
2 2 2
3 2 2 3
2 1 5
2 2 3
2 1 3
2 1 1
2 1 3
2 1 2
2 1 1
2 1 1
1 5
2 2 3
2 3 1
1 7
1 7
2 1 4
```

Bijlage 7. Loten met Japanse puzzels.

De snelle inzendersprijs van de Nederlandse Informatica Olympiade 2002-2003 is bedoeld voor die leerlingen die opgave 1 van de eerste ronde voor 1 december 2002 hebben ingestuurd. Zes deelnemers zijn uit de stapel inzendingen getrokken; op 9 mei worden de prijzen tussen hen verdeeld. Eén van deze zes deelnemers wint de hoofdprijs van 500 euro; de anderen winnen elk 100 euro.

Door het lotingsprogramma wordt de afbeelding van een Japanse puzzel van 9 bij 9 vakjes gegenereerd. Vervolgens wordt voor de deelnemers om beurten een getal tussen 0 en 31 getrokken. Dit getal hoort bij een verdeling van een aantal gekleurde en lege vakjes voor een regel van 5 vakjes (zie de tabel onderaan de bladzijde).
Als er een vrij fragment in de afbeelding voorkomt met deze samenstelling, mag dit door de deelnemer worden geblokkeerd. Als dat niet het geval is, gaat de beurt voorbij en is de deelnemer uitgeschakeld.

Als er nog maar één deelnemer over is is dat degene die de hoofdprijs van de snelle inzendersprijzen wint. Als er nog meer deelnemers in de strijd zijn wint degene die het laatste fragment heeft geblokkeerd.

0	O	O	O	O	O
1	O	O	O	O	X
2	O	O	O	X	O
3	O	O	O	X	X
4	O	O	X	O	O
5	O	O	X	O	X
6	O	O	X	X	O
7	O	O	X	X	X
8	O	X	O	O	O
9	O	X	O	O	X
10	O	X	O	X	O
11	O	X	O	X	X
12	O	X	X	O	O
13	O	X	X	O	X
14	O	X	X	X	O
15	O	X	X	X	X
16	X	O	O	O	O
17	X	O	O	O	X
18	X	O	O	X	O
19	X	O	O	X	X
20	X	O	X	O	O
21	X	O	X	O	X
22	X	O	X	X	O
23	X	O	X	X	X
24	X	X	O	O	O
25	X	X	O	O	X
26	X	X	O	X	O
27	X	X	O	X	X
28	X	X	X	O	O
29	X	X	X	O	X
30	X	X	X	X	O
31	X	X	X	X	X

Bijlage 8. Links.

http://www.geocities.com/activityworkshop/puzzlesgames/nonograms/index.html
Een site met informatie over Japanse puzzels.

http://puzzle.geoweb.ge/index.htm
Een Georgische site waar puzzels kunnen worden gevonden en waar ook de geschiedenis van Japanse puzzels wordt verteld.

http://www.08033.com/Nonograms.nsf/Welcome/Welcome?OpenDocument
De geschiedenis van Japanse puzzels.

http://www.research.att.com/cgi-bin/access.cgi/as/njas/sequences/eisA.cgi?Anum=A000045
Dit is een link over de rij van Fibonacci.

http://www-users.cs.york.ac.uk/~tw/Teaching/nonogram.html
Een programmeeropdracht met veel links.

http://www.liacs.nl/home/kosters/AI/jaar2001.html
De site met de opgave van Walter Kosters voor een genetisch algoritme.

http://www.esat.kuleuven.ac.be/education/options/da/iroptiondna/knipsels/tijd20020914.html
Het artikel over "Op mieren rekenen".