Puzzelen

De puzzels in Java Magazine hebben een nieuw karakter en een nieuwe vorm. Wouter Klein Heerenbrink, die twee jaar lang de Puzzlers van Joshua Bloch en Neil Gafter bewerkte, neemt het heft nu in eigen handen. U bent gewaarschuwd: het wordt er niet eenvoudiger op. Voor deze puzzeltjes schrijf je een handig algoritme dat eens iets afwijkt van de standaard procedures. Ze gebruiken basis-methoden uit de programmeertaal. Het zijn programmeerpuzzels, die je (ook) met Java kunt oplossen. Antwoorden kunt u naar de redactie mailen. Onder de goede inzenders wordt telkens een IT-gerelateerd boek verloot.

Genetische algoritmen

In maart 2011 was er een historische gebeurtenis. Een grote publieke verkoop van spullen uit het Koninklijk Huis. Voorafgaand aan deze veiling moesten alle spullen van onder andere Paleis Soestdijk naar het veilinghuis Sotheby's worden gebracht. Er zijn meer dan 1725 kavels verplaatst, dat zijn veel vrachtwagenladingen.

Het bedrijf dat de verplaatsing voor haar rekening mocht nemen wilde natuurlijk zo efficiënt mogelijk alle spullen verplaatsen. Brandstof en belading moeten optimaal worden benut. Het aantal mogelijkheden waarop de vrachtwagens ingedeeld kunnen worden is eindeloos. Als we het aantal manieren (volgorden) uitrekenen, waarop de dozen naar de vrachtwagen gebracht kunnen worden, dan komen we uit op een getal van 1725 faculteit (*1): veel te veel. En we hebben nog niet eens meegerekend dat iedere doos die de vrachtwagen in gaat, op verschillende manieren kan worden neergezet: naast de vorige doos, op de vorige doos, voor de vorige doos etc.

Een veel gebruikte oplossing voor dit probleem wordt aangedragen door genetische algoritmen; geïnspireerd door de natuur, door de wijze waarop mensen, dieren en planten evolueren tot organismen die zo efficiënt mogelijk leven en zich aanpassen aan hun omgeving. Deze algoritmen worden toegepast voor indelingsproblemen zoals de verhuisdozen, maar ook voor complexe roosterproblemen zoals lesroosters. Ook worden de algoritmen toegepast op wiskundige problemen, zoals bijvoorbeeld het vinden van een maximum van een formule, of de beste trendlijn in een ingewikkelde scatterplot.

Om te laten zien hoe een genetisch algoritme werkt gaan we op zoek naar de trendlijn in een (simpele) scatterplot (*2), Voor het gemak spreken we af dat de trendlijn door de oorsprong moet gaan en dat de richtingscoëfficiënt een geheel getal is. Met andere woorden, we gaan op zoek naar het gehele getal a in de formule f(x) = ax; zodat f(x) de trendlijn het beste beschrijft.

De punten uit de scatterplot zijn gegeven in onderstaande tabel.

x	y
1	8
2	19
3	23
4	33
5	39
6	50

1. Bepaal de genen

De eigenschappen van een individu liggen vast in het DNA. DNA bestaat uit een aantal chromosomen met daarop genen. Ieder gen staat, eventueel in combinatie met andere genen, voor een eigenschap (bijvoorbeeld blauwe ogen).

In het algoritme gaan we straks een populatie maken van individuen met één chromosoom met daarin een aantal genen. In het geval van ons voorbeeld kiezen we ervoor een individu zijn chromosoom een waarde voor a te laten representeren. Als we deze waarde van een chromosoom binair uitdrukken, dan hebben we een prima representatie voor de genen. In onderstaand figuur zien we een representatie van het chromosoom dat voor a=10 staat:

Wanneer je werkt met getallen dan is een binaire representatie vaak handig. Je zou echter ook gewoon

getallen, cijfers of zelfs een Class kunnen gebruiken als gen.

2. Bepaal je start-populatie

Het algoritme vereist dat we beginnen met een startpopulatie. In tegenstelling tot de natuur, blijft de populatie gedurende het gehele algoritme constant. Voor dit voorbeeld nemen we een populatie van vijf.
De startpopulatie wordt geheel willekeurig samengesteld. Dit doen we door vijf willekeurige getallen te trekken. Een blik op de data leert ons dat de waarde van a ergens tussen 0 en 15 zal liggen. Op dit interval kiezen we de willekeurige punten. Bijvoorbeeld: 3, 7, 9, 13, 15. (*3)

Chromosoom	Gen 1	Gen 2	Gen 3	Gen 4
3	0	0	1	1
7	0	1	1	1
9	1	0	0	1
13	1	1	0	1
15	1	1	1	1

3. Fitness

Wie heeft de grootste overlevingskans? Welk getal beschrijft de beste trendlijn? Voor ieder individu moeten we bepalen hoe goed deze is. Om te spreken met de woorden van Darwin: hoe "fit" is ieder getal, welke is de "fittest"?
Om dit te bepalen gaan we een fitnessfunctie opstellen die beschrijft hoe goed de trendlijn is met de gekozen waarde. De beste trendlijn is de lijn waarbij de afwijkingen tot de datapunten zo klein mogelijk is. We kunnen hiervoor de standaardformule voor de standaardafwijking gebruiken.

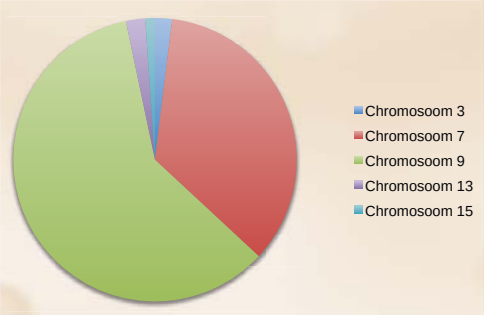
$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N}(x_i - \mu)^2$$

Hoe groter het getal van de standaardafwijking, hoe slechter het getal is. Voor de fitnessfunctie willen we natuurlijk het omgekeerde, dus als fitnessfunctie gebruiken we 1 gedeeld door de standaardafwijking. Hoe kleiner de afwijking, hoe groter het resultaat van de fitnessfunctie. In onderstaande tabel staat voor ieder chromosoom de fitnesswaarde in de kolom fitness.
Maar wie is het "fittest"? Hiervoor berekenen we de fitnessratio, de mate van fitness ten opzichten van de andere individuen uit de populatie. Deze fitnessratio kun je berekenen door alle fitnesswaarden van ieder individu op te tellen. Vervolgens deel je per individu zijn fitnesswaarde door de totale fitnesswaarde. Het resultaat staat in onderstaande tabel.

Chromosoom	Std. Afwijking	Fitnesswaarde	Fitnessratio
3	405.17	0.00247	0.019
7	22.50	0.04444	0.350
9	13.17	0.07594	0.598
13	385.50	0.00278	0.022
15	713.17	0.00140	0.011
Totaal	*1512.50*	*0.12705*	*1*

4. Paringstijd

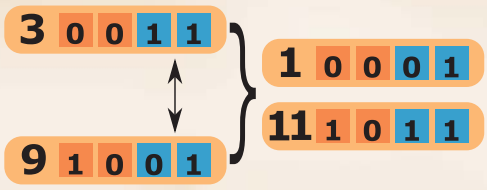
Nu we weten hoe goed ieder gekozen getal was, is het nu tijd om te bepalen hoe de nieuwe populatie eruit ziet. Bij een genetisch algoritme ontstaat er telkens een compleet nieuwe populatie op basis van de huidige populatie. Alle individuen uit de vorige ronde 'sterven' uit.
We gaan uit de bestaande populatie een nieuwe populatie selecteren via een 'roulettewiel' selectie. Op basis van de fitnessratio uit de vorige stap kunnen we de individuen in een pie-chart zetten. Hoe groter de fitnessratio, hoe meer oppervlakte ze innemen op de pie-chart.
Op de pie-chart plaatsen we een roulettewiel en we draaien die vijf maal rond (met andere woorden: we doen een uniforme trekking tussen 0 en 1). De individu die aangewezen wordt door het roulettewiel wordt ingevoegd in de nieuwe populatie. Wordt een individu vaker aangewezen dan mag deze vaker mee. Het is duidelijk dat de waarde 9 het grootste deel van de pie-chart voor zijn rekening neemt. De kans dat we deze het vaakst kiezen is zeer groot.



5. Cross-over

Als we de nieuwe populatie gewoon opbouwen uit exact dezelfde getallen als de vorige ronde maar dan in een andere verhouding, komen we natuurlijk nooit op een beter antwoord. Daarom gaan we de populatie onderling kruisen.
We pakken willekeurig twee chromosomen van individuen en daarvan wisselen we een paar genen om. Bijvoorbeeld de eerste twee genen van het chromosoom. In figuur zien we de wisseling van de genen van chromosoom 3 en 9. Er ontstaan nu nieuwe chromosomen, namelijk 11 en 1.
Door deze cross-over combineer je in veel gevallen twee goede eigenschappen met elkaar. Dit is het beste te visualiseren met het voorbeeld van de verhuisdozen. Stel je hebt twee individuen die beide een indeling van drie redelijk efficiënt gevulde vrachtwagens beschrijven. Als we nu de indeling van vrachtwagen 1 uit het ene individu verwisselen met de indeling van vrachtwagen 1 uit het andere individu, dan combineren 'het beste van de een' met 'het beste van de ander' en dan kan het zo zijn dat er in totaliteit een beter indeling ontstaat.
Belangrijk bij de cross-over is dat het iedere keer wordt

toegepast omdat de beste optie hierdoor mogelijk te snel wordt veranderd in een minder goede optie. Op hoeveel procent van de populatie een cross-over moet worden toegepast is per casus verschillend.



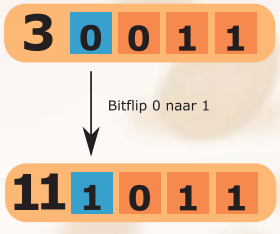
6. Mutaties

Wanneer je zoekt naar het maximum van een grafiek met meerdere toppen dan kan het voorkomen dat het algoritme een andere top aanziet als het maximum van de grafiek omdat hij bij het bepalen van de beste optie de echte hoogste top niet meegenomen heeft omdat deze nooit voorkwam in de populatie. We werken dan naar een lokaal optimum.

In ons voorbeeld zal dat niet snel gebeuren. Om het toch te illustreren zou je de ondenkbare situatie kunnen schetsen waarbij de willekeurig gekozen startpopulatie bestaat uit allemaal individuen met de waarde 15. Bepalen we nu de fitnessratio dan zal deze bij ieder individu gelijk zijn aan 1/6. Bij de cross-over wisselen we genen uit, maar die zijn voor beide individuen gelijk. Dit zou betekenen dat de nieuwe populatie ook weer alleen bestaat uit individuen met de waarde 15. Dit zou betekenen dat we concluderen dat a=15; maar dit is natuurlijk niet de beste trendlijn.

Daarom zijn er mutaties, net als in de echte wereld. Bij mutaties veranderen we de waarde van een willekeurig gen. In ons geval zou dat een bitflip betekenen. Doen we een bitflip op het eerste gen van de chromosoom die 3 representeert (=0011), dan representeert deze opeens 11 (=1011). Daardoor onderzoeken we opeens een heel andere plek in de oplossingsruimte.

Mutaties moeten niet al te vaak voor komen (orde van grootte: 0.1% van de gevallen), omdat het algoritme anders nooit convergeert naar de beste oplossing.



7. Herhalen maar

Met het genereren van één generatie hebben we nog geen beste oplossing. Met de nieuwe populatie kunnen we stap drie tot en met zes vele malen opnieuw uitvoeren. Het gevonden antwoord zal steeds beter en beter worden.

Stoppen kan na een vooraf vastgesteld aantal iteraties (bijvoorbeeld 1000) of nadat de populatie nauwelijks nog verandert, wat suggereert dat het optimum bereikt is. De allerbeste trendlijn zal nooit gevonden worden in dit voorbeeld, omdat de beste a geen geheel getal is. Wel zal er geconvergeerd worden naar een getal dat het dichtst in de buurt ligt van het gebroken getal. Dit voorbeeld is op papier een aantal keer te herhalen en dan zul je zien dat het antwoord 8 is.

Met dit algoritme zorg je ervoor dat het convergeert naar het goede antwoord waardoor je niet alle mogelijkheden langs hoeft. Op basis van de probeersels bepaal je hoe goed iets is en vervolgens ga je in die richting verder zoeken. Een zeer krachtige strategie!

De puzzel

Ook deze keer kun je aan de slag. Natuurlijk omhelst de puzzel dit keer een genetisch algoritme. De opdracht is als volgt: Schrijf een genetisch algoritme dat het spelletje MasterMind kan spelen. Niet de standaardvariant met 4 pinnetjes en 6 kleuren; maar de uitgebreide variant met 30 pinnetjes en 9 kleuren. Dat maakt dat er 9^30 = 2.099139643e+46 mogelijkheden zijn.

Op de website van JavaMagazine staat de Java code en documentatie van een klein MasterMind spelletje waarmee wij gaan testen. Het spel laat je oneindig veel pogingen maken en geeft enkel terug hoeveel pinnetjes op de goede plaats stonden en hoeveel pinnetjes wel in de code zaten maar niet op de goede plaats.

*1 1725 x 1724 x 1723 x 1722 x 1721 x ... x 1: Heel erg groot. Ter illustratie, 92! = 1.24384140546413e+142

*2 Dit voorbeeld is natuurlijk makkelijk uit te rekenen met de hand / andere formules; maar dat is gedurende de uitleg wel zo gemakkelijk.

*3 Alle punten in de dataset liggen onder de lijn 15x; 15 is gekozen omdat dit het maximale getal is dat het gekozen chromosoom kan representeren (binair: 1111)

Winnaar Puzzel 'Als jij het kan'

Het goede antwoord op de puzzel in JM1 is : "watwindjewandezepuzzelmmmmmm", zo schrijft één van de vele inzenders. Hij suggereert dat de spelfout in de gecodeerde tekst vast is om dictonary attacks te dwarsbomen. Welnu, het is geen spelfout. Het is een bij-effect van de codeer methode. Doordat de i/j samengevoegd zijn in de Polybius-square, heb je nog maar 25 letters in het alfabet. Daardoor krijg je of het antwoord 'wat wind je wan deze puzzel" of "vat vind je van deze puzzel". Die rekenen we dus allebei goed, evenals het tot rechtgeaard Nederlands vertaalde "wat vind je van deze puzzel". Eigenlijk zijn alle inzenders winnaar, maar we hebben één boek in het vooruitzicht gesteld. Dat gaat naar Ruth Alkema