# De WeetNiet-WistAl-WeetWel-WeetOok puzzel

Maarten Fokkinga

Aan twee personen zijn het product en de som van twee verschillende positieve gehele getallen onder de honderd gegeven. Op de vraag wat die getallen zijn, zeggen zij achtereenvolgens en om beurten: “Ik weet het niet”, “Dat wist ik al”, “Nu weet ik het wel”, “Nu weet ik het ook”. Wij construeren een elegant programma dat het getallenpaar oplevert.

**Inleiding** Van twee gehele getallen $x$ en $y$ met $2 \leq x < y \leq 100$ kent Pim het Product en Sim de Som. Op de vraag aan Pim en Sim of zij $x$ en $y$ zelf kennen, ontstaat de volgende dialoog:

> Pim: Ik weet het niet.
> Sim: Dat wist ik al.
> Pim: Nu weet ik het wel.
> Sim: Nu weet ik het ook.

Aan de lezer gevraagd: wat zijn $x$ en $y$?

Dit probleem is afkomstig van prof. H. Freudenthal. De conditie $y \leq 100$ mag ook vervangen worden door $x + y \leq 100$; dat maakt het makkelijker om met de hand een oplossing te vinden. Het getal 100 kan trouwens veel groter gekozen worden; de extra bewijsverplichtingen zijn dan om te bewijzen dat de dialoog van Pim en Sim niet past bij andere paren getallen (dan het paar dat al met de grens 100 gevonden is). Toen ik dit lang geleden met pen en papier oploste, speelden priemgetallen een grote rol: ik had een tabel voor me van alle getallen tot honderd waarin de priemgetallen omcirkeld waren. Aan de hand van die tabel kon ik steeds meer getallen uitsluiten.

Hieronder volgt een berekening van de oplossing met behulp van Miranda. Het programma is een directe weergave van de dialoog. De structuur komt met de dialoog overeen, en iedere zin is, na letterlijke vertaling in Miranda, direct te herkennen. Het begrip priemgetal speelt niet expliciet een rol.

Om de efficiëntie te verbeteren (door inruil van berekeningstijd tegen opslagruimte) passen we ‘memoïsatie’ toe: een kleine syntactische wijziging die de semantiek niet wijzigt maar wel een groot operationeel gevolg heeft.

**Hulp-definities** Om het verhaal te stroomlijnen geven we nu alvast een paar hulpdefinities. Die zouden we straks toch tegenkomen. Allereerst namen voor de constanten:

$$laag \;=\; 2$$
$$hoog \;=\; 100$$

Dan een functie om efficiënt te testen of een lijst precies 1 lang is:

$$is1\ [x] \;=\; True$$
$$is1\ xs \;=\; False$$

En als laatste hulpdefinitie de ontbinding van een getal in twee summanden $(a, b)$ en in twee factoren $(a, b)$ met $laag \leq a < b \leq hoog$ (aangenomen dat argument $s$ hoogstens $hoog{-}1{+}hoog$ is):

$$ontbindingenS\ s \;=\; [(a,\ s-a) \mid a \leftarrow [laag\ ..\ (s-1)\ div\ 2]]$$
$$ontbindingenP\ p$$
$$=\; [(a,b) \mid a \leftarrow [laag\,..\,sqrt(p-1)];\ p\ mod\ a \;=\; 0;\ b \leftarrow [p\ div\ a];\ b \;\leq\; hoog]$$

**De dialoog** We zullen nu een reeks verzamelingen *over* definiëren van paren "die nog over zijn", dat wil zeggen het gezochte paar zit in die verzamelingen. Met iedere zin uit de dialoog correspondeert een inkrimping van *over*. In het begin is alles "nog over":

$$over0 \;=\; [(x,y) \mid x \leftarrow [laag\ ..\ hoog-1];\ y \leftarrow [x+1\ ..\ hoog]]$$

Pim zegt "Ik weet het niet".
Dus het product dat Pim kent is op meer dan één manier in twee factoren te ontbinden. Deze eigenschap noemen we *pimWeetNiet*:

$$pimWeetNiet\ p \;=\; \neg\ is1\ (ontbindingenP\ p)$$

Blijft over: alleen die paren uit $over0$ waarvan het product aan *pimWeetNiet* voldoet:

$$over1 \;=\; [(x,y) \mid (x,y) \leftarrow over0;\ pimWeetNiet\ (x{\times}y)]$$

Sim zegt "Dat wist ik al".
Dus de som die Sim kent is alleen maar te schrijven is als $a + b$ met $pimWeetNiet\ (a{\times}b)$. Deze eigenschap noemen we *simWistAl*:

$$simWistAl\ s \;=\; and\ [pimWeetNiet\ (a{\times}b) \mid (a,b) \leftarrow ontbindingenS\ s]$$

Blijft over: alleen die paren uit $over1$ waarvan de som aan *simWistAl* voldoet:

$$over2 \;=\; [(x,y) \mid (x,y) \leftarrow over1;\ simWistAl\ (x+y)]$$

Pim zegt "Nu weet ik het wel".
Dus het product dat Pim kent is op één manier te schrijven als $a{\times}b$ met $simWistAl\ (a+b)$.
Deze eigenschap noemen we $pimWeetWel$:

$$pimWeetWel\ p\ =\ is1\ [(a,b)\mid (a,b)\leftarrow ontbindingenP\ p;\ simWistAl\ (a+b)]$$

Blijft over: alleen die paren uit $over2$ waarvan het product aan $pimWeetWel$ voldoet:

$$over3\ =\ [(x,y)\mid (x,y)\leftarrow over2;\ pimWeetWel\ (x{\times}y)]$$

Sim zegt: "Nu weet ik het ook".
Dus de som die Sim kent is op één manier te schrijven als $a+b$ met $pimWeetWel\ (a{\times}b)$.
Deze eigenschap noemen we $simWeetOok$:

$$simWeetOok\ s\ =\ is1\ [(a,b)\mid (a,b)\leftarrow ontbindingenS\ s;\ pimWeetWel\ (a{\times}b)]$$

Blijft over: alleen die paren uit $over3$ waarvan de som aan $simWeetOok$ voldoet:

$$over4\ =\ [(x,y)\mid (x,y)\leftarrow over3;\ simWeetOok\ (x+y)]$$

**De oplossing** De evaluatie van $over4$ geeft een lijst te zien met precies één paar. Dat is dus het gevraagde getallenpaar. Om de lezer het plezier van puzzelen niet te ontnemen, noemen we dat paar niet. De kosten van de evaluatie zijn:

$$reductions\ =\ 4794276,\ cells\ claimed\ =\ 6831967,\ no\ of\ gc's\ =\ 6,\ cpu\ =\ 45.77$$

**Memoïsatie** Ik zie maar één mogelijkheid tot echte efficiëntie-verbetering. Berekeningen van $simWeetAl\ s$ worden veelvuldig herhaald met hetzelfde argument $s$. We kunnen eenvoudig herhaalde berekeningen voorkomen door *memoïsatie*: een lijst *memo* van functieresultaten aanleggen, en daarin de het functieresultaat opzoeken. Doordat Miranda lazy geëvalueerd wordt, wordt de lijst *memo* alleen dáár uitgerekend waar de berekening dat echt noodzakelijk maakt. Dus de definitie

$$simWistAl\ s\ =\ and\ [pimWeetNiet\ (a{\times}b)\mid (a,b)\leftarrow ontbindingenS\ s]$$

wijzigen we in:

$$simWistAl\ s$$
$$=\ memo!s$$

$$
\begin{array}{l}
\quad \mathit{where} \\
\quad \mathit{memo} \;=\; [\mathit{and}\ [\mathit{pimWeetNiet}\ (a{\times}b) \mid (a,b) \leftarrow \mathit{ontbindingenS}\ s] \mid s{\leftarrow}[0..]]
\end{array}
$$

De locale lijst memo mag ook een globale lijst zijn; dat is even efficiënt! Het blijkt dat dit een factor 3 in de kosten scheelt; de kosten van de evaluatie van *over*4 zijn nu:

$$\mathit{reductions} \;=\; 1565012,\ \mathit{cells\ claimed} \;=\; 2280625,\ \mathit{no\ of}\ gc's \;=\; 2,\ \mathit{cpu} \;=\; 14.22$$

(Met de voorwaarde $x + y \leq 100$ in plaats van $y \leq 100$ waren de kosten al veel minder en is de versnelling zelfs een factor 6.) Eenzelfde memoïsatie van *simWeetOok* levert maar een kleine besparing op. Het blijkt *niet* efficiënter te zijn om functies *pimWeet*... te memoïseren. De reden is dat de benodigde memo-lijst van *simWistAl* hoogstens 199 (= *hoog*−1+*hoog*) lang is, terwijl die van *pimWeet*... mogelijk heel lang zijn: (*hoog*−1)×*hoog*. Door die grote lengte ontstaat er al gauw een tekort aan ruimte.

Voor de grap schrijven we *over*4 uit, zonder *over*3, ..., *over*0 expliciet te definiëren:

$$
\begin{array}{l}
\mathit{over} \\
=\ [(x,y) \mid \\
\quad x \leftarrow [\mathit{laag}\ ..\ \mathit{hoog}-1];\ y \leftarrow [x+1\ ..\ \mathit{hoog}]; \\
\quad \mathit{pimWeetNiet}\ (x{\times}y); \\
\quad \mathit{simWistAl}\ (x+y); \\
\quad \mathit{pimWeetWel}\ (x{\times}y); \\
\quad \mathit{simWeetOok}\ (x+y)]
\end{array}
$$

Tezamen met de definities van *pimWeetNiet*, ..., *simWeetOok* is dit het hele programma.

**Naschrift** Stel dat er géén bovengrens aan de getallen gegeven was (dus $\mathit{hoog} = \infty$). Er zijn dan twee wijzigingen nodig in bovenstaande definities. Laat in *ontbindingenP* de conditie $b \leq \mathit{hoog}$ vervallen, en wijzig het begin van *over* als volgt:

$$\mathit{over} \;=\; [(x,y)\ //\ x \leftarrow [\mathit{laag}\ ..];\ y \leftarrow [x+1\ ..];\ \ldots$$

De diagonalisatie '//' zorgt er voor dat *alle* paren $(x, y)$ aan bod komen. Er blijken nu *drie* getallenparen $(x, y)$ te zijn met $x + y \leq 100$, en *vier* met $y \leq 100$, waarvoor Pim en Sim de gegeven dialoog terecht kunnen houden. Kennis van de bovengrens is dus een essentieel gegeven voor Pim en Sim, want mét kennis van de bovengrens $\mathit{hoog} = 100$ is er maar één zo'n paar. Van de bovengrens wordt essentieel gebruik gemaakt in *ontbindingenP*.