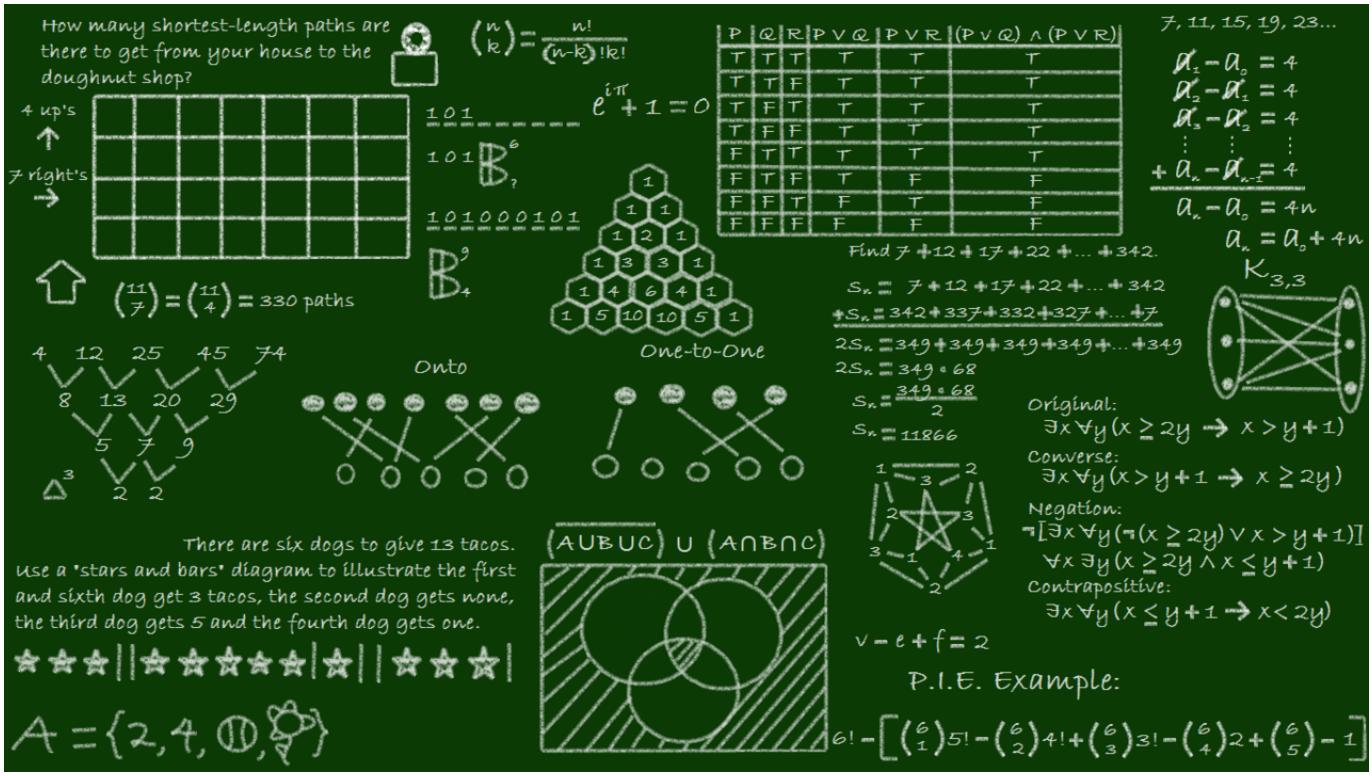
Het vinden van alle natuurlijke getallen met behulp van drie wiskundige operatoren



Do not worry about your difficulties in Mathematics. I can assure you mine are still greater.
~ Albert Einstein

Universiteit van Amsterdam

Heuristieken - 2014

Rick Slot - rickslot@live.nl

John Zwarthoed - john.zwarthoed@gmail.com

Tony Abidi - tony.abidi@student.uva.nl

1. Inleiding

In dit verslag beschrijven wij hoe we onze casus van het vak Heuristieken hebben opgelost. Tijdens het eerste college werden wij geacht om uit twaalf opgaven één casus te kiezen. Wij kozen voor Number Crunching. Donald Knuth heeft de hypothese gesteld dat alle natuurlijke getallen gemaakt kunnen worden door op het begingetal een reeks van wiskundige operaties uit te voeren. Deze operaties zijn: faculteit, vierkantswortel en naar beneden afronden (floor). De opdracht is om hiervoor een deelbewijs te vinden met behulp van twee cases. Het vinden van de getallen 1 tot en met 100 en het vinden van de getallen 1 tot en met 10000.

Wat dit probleem lastig maakt is het berekenen van grote getallen. Dit duurt namelijk met de processoren van nu nog erg lang. Het berekenen van 1000000! duurt bijvoorbeeld al enkele minuten.

De toestandsruimte van onze casus is oneindig. Dit komt doordat er geen limiet zit aan het aantal getallen dat gemaakt kan worden. Wel hebben wij in ons programma een limiet moeten stellen aan het grootste getal dat gemaakt kan worden. Dit omdat het programma er anders te lang over doet om het volgende getal te berekenen, simpelweg omdat de processor niet snel genoeg is. Wij hebben ervoor gekozen om de probleemstelling op te lossen in de taal Python. Dit omdat deze taal het gebruik van grote getallen eenvoudiger maakt.

```
value: 30 numbers left: 95
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO']
value: 5 numbers left: 94
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO']
value: 10 numbers left: 93
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO', 'FAC', 'SQR', 'FLO']
value: 3 numbers left: 92
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO', 'FAC', 'SQR', 'SQR', 'FLO']
value: 6 numbers left: 91
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO', 'FAC', 'SQR', 'SQR', 'FLO', 'FAC']
value: 43 numbers left: 90
Sequence: ['FAC', 'FAC', 'SQR', 'SQR', 'SQR', 'SQR', 'SQR', 'FLO', 'FAC', 'SQR', 'FLO', 'FAC', 'SQR', 'SQR', 'FLO']
```

Fig. 1. Een gedeelte van de output uit ons programma dat weergeeft welke volgorde van operaties moet worden uitgevoerd om een bepaald getal te vinden.

2. Mehodes

Om het probleem op te lossen maken wij gebruik van een boom datastructuur. We bouwen deze boom breadth-first op. Dit betekent dat we de boom laag voor laag opbouwen waardoor altijd het kortste pad naar een bepaalde node wordt gevonden. Dit betekent dat wanneer je één stap dieper de boom in gaat, er een operator bij komt. Om de boom te vullen maken we gebruik van een queue. Een queue is een datastructuur die volgens het First In First Out (FIFO) principe werkt. Een queue is dus eigenlijk niks anders dan een wachtrij.

De nodes in de boom hebben een aantal belangrijke waardes nodig:

- Het getal
- De operatie die is uitgevoerd om dat getal te krijgen
- De parent-node

De waarde van een node wordt berekend door de operatie van die node op de waarde van de parent-node uit te voeren.

In het begin van het programma wordt een node aangemaakt met het basisgetal als waarde en null pointers voor de operatie en de parent-node. Vervolgens wordt deze node in de queue geplaatst. Een andere functie zorgt ervoor dat zolang de queue niet leeg is er een algoritme wordt uitgevoerd: op de waarde van de node worden de drie gegeven operaties uitgevoerd. Voor elke operatie wordt een nieuwe node met de nieuwe waarde aangemaakt. Deze node wordt vervolgens in de queue gezet. Door steeds een node uit de queue te halen en deze sequentie af te lopen wordt de boom laag voor laag opgebouwd. Dit is het principe van breadth-first search.

Zodra er een node is met de waarde van een getal dat gezocht werd, kan de node worden opgeslagen. Vervolgens kan door het aflopen van de parents van die node een lijst van operaties worden gemaakt. Door deze om te draaien krijg je een lijst met alle operaties die nodig zijn om tot een getal te komen.

We hebben ons algoritme onder verschillende omstandigheden getest. Allereerst moesten wij een limiet stellen aan de maximale grootte van een getal. Dit omdat de getallen anders zo groot worden dat het al dagen duurt voordat het volgende getal wordt gevonden. Om tot een goed resultaat te komen hebben wij ons algoritme getest met drie maximale groottes. Dit zijn respectievelijk: 30000!,

3. Resultaten

Het vinden van 1 tot en met 100:

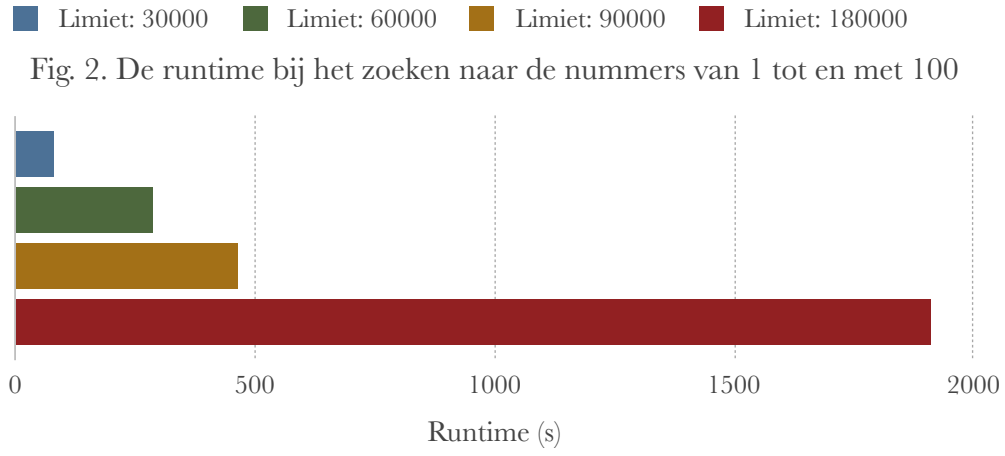


Fig. 2. De runtime bij het zoeken naar de nummers van 1 tot en met 100

In Fig. 2. is de runtime voor het vinden van de getallen 1 tot en met 100 te zien. Hoe groter de limiet van de maximale grote van het getal is, des te langer het programma erover doet. Dit komt doordat de processor er langer over doet om grotere getallen te berekenen. Als er grotere getallen berekent worden, kunnen er ook meer getallen gemaakt worden.

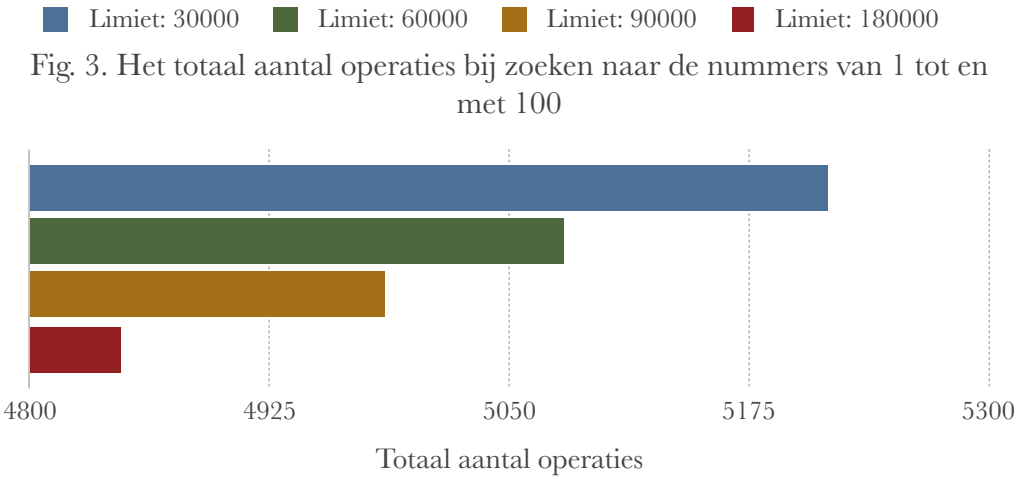


Fig. 3. Het totaal aantal operaties bij zoeken naar de nummers van 1 tot en met 100

In Fig. 3. is het totaal aantal wiskundige operaties te zien die nodig zijn om de getallen 1 tot en met 100 te vinden. Wat hierbij opvalt is dat wanneer het limiet hoger wordt er minder operaties nodig zijn om de getallen te vinden. Dit komt doordat er bij een hoger limiet grotere getallen gemaakt kunnen worden. Als er grotere getallen zijn kunnen de sequenties anders worden. Hierbij kan het voor komen dat de sequenties korter worden.

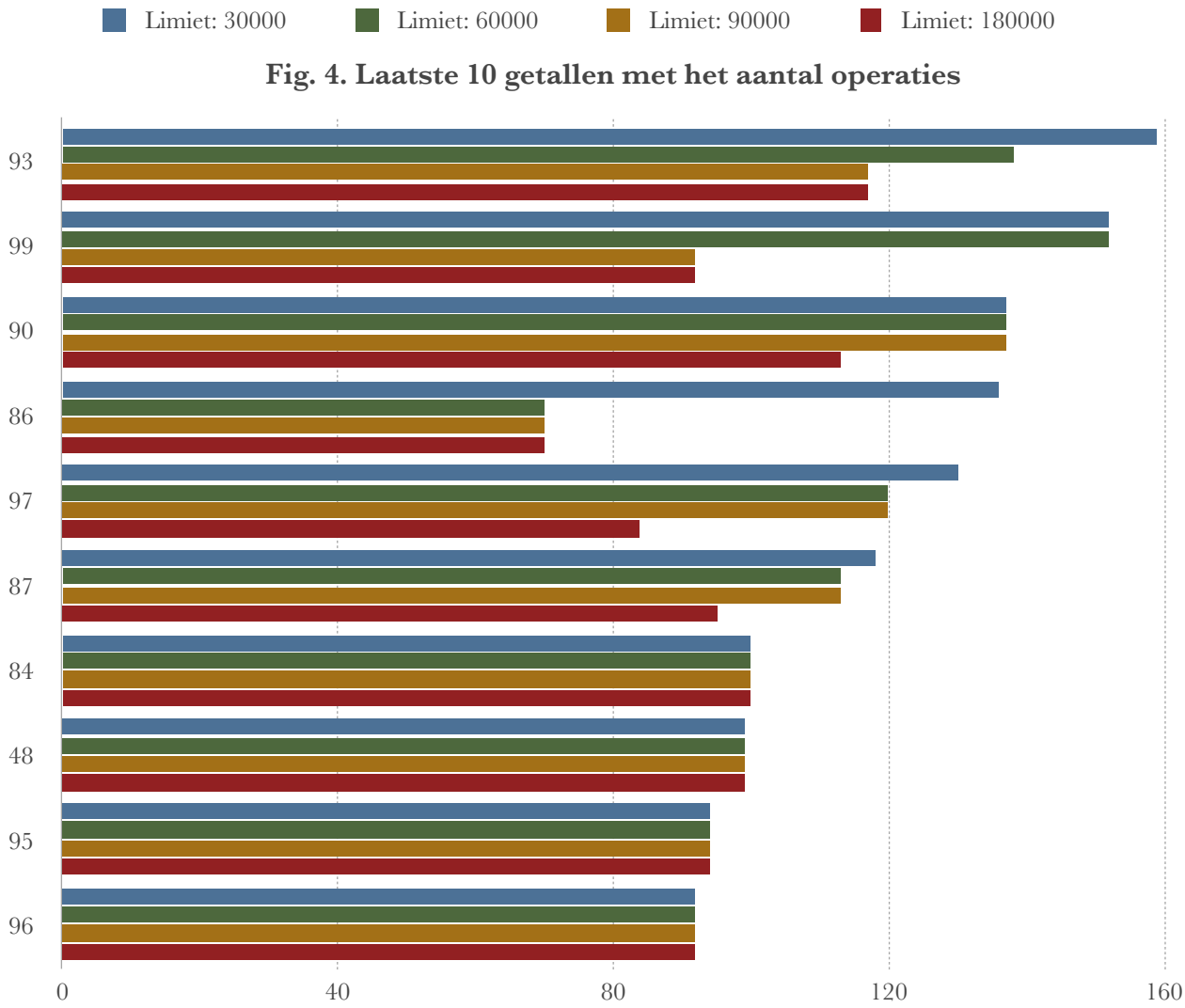


Fig. 4. Laatste 10 getallen met het aantal operaties

In fig. 4. is het aantal operaties te zien die nodig zijn om de laatste 10 getallen te maken die gevonden worden bij een limiet van 30000. Hierbij hebben wij weer met vier verschillende limieten getest. Wat opvalt is dat des te hoger het limiet, hoe lager het aantal operaties.

Er kan een verband worden gelegd tussen het aantal operaties die nodig zijn en de runtime van het programma. Als er een hoger limiet gesteld wordt zal de runtime langer worden, maar het aantal operaties neemt af.

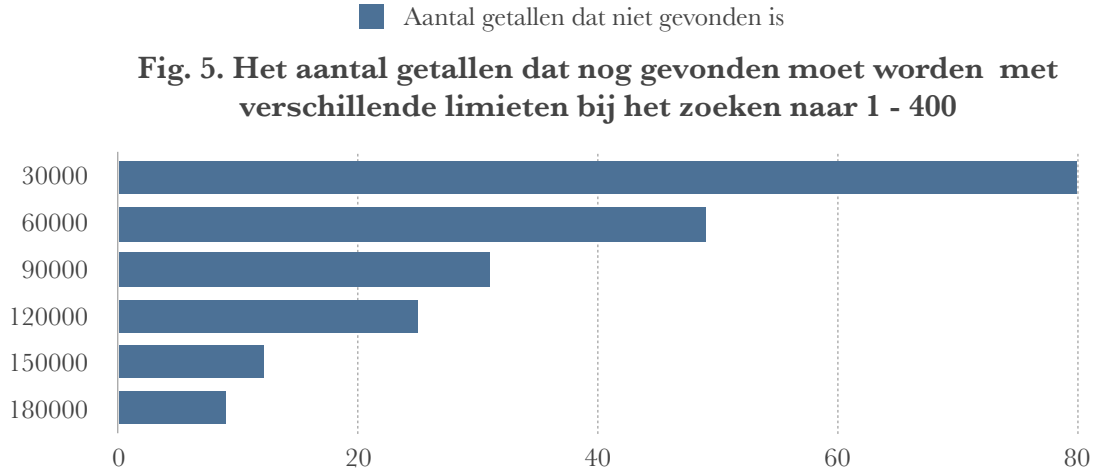


Fig. 5. Het aantal getallen dat nog gevonden moet worden met verschillende limieten bij het zoeken naar 1 - 400

Met een hoger limiet kunnen nieuwe reeksen van operaties gemaakt worden waardoor nieuwe nummers gevonden kunnen worden. Het kan hierbij voorkomen dat de reeksen korter worden. In Fig. 5. is dit te zien.

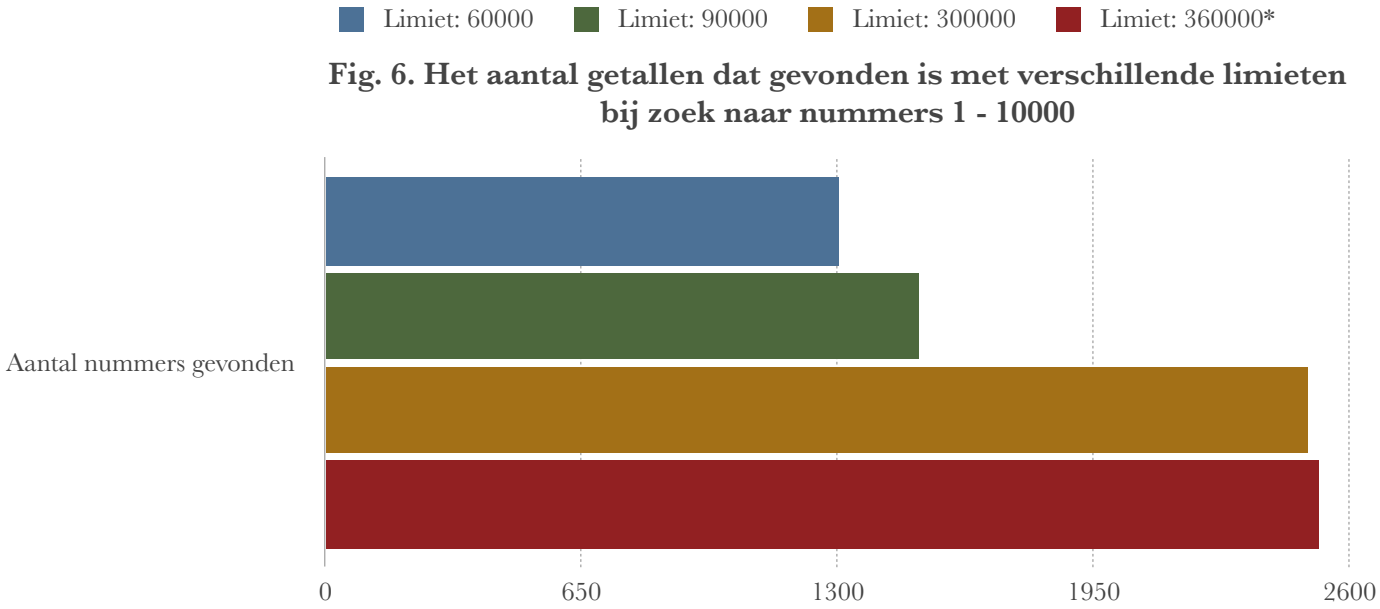


Fig. 6. Het aantal getallen dat gevonden is met verschillende limieten bij zoek naar nummers 1 - 10000

*Het programma werd na een dag stop gezet waardoor en kon waarschijnlijk nog meer dingen vinden.

Ook hebben we gezocht naar alle nummers van 1 tot en met 10000. We hebben dit geprobeerd met verschillende limieten en merkte dat we niet alle nummers konden vinden met deze limieten. Uiteindelijk hebben we het met een hoger limiet geprobeerd, maar dit duurde te lang en we moesten het programma stoppen. Hieruit bleek weer wel dat je met een hoger limiet meer nummers vind.

4. Conclusie

Onze intuïtie zegt dat alle natuurlijke getallen berekend kunnen worden met het begingetal 4 en drie wiskundige operaties. Het probleem is echter dat er een limiet moet worden gesteld aan het grootste getal dat gemaakt kan worden. Dit belemmert het vinden van hele grote getallen. Op het moment dat er een computer is die met grotere getallen kan werken zullen er meer getallen gevonden kunnen worden.