**Number Crunching**

**Artikel 42**

**Jord Rood**

Universiteit van Amsterdam

**Arent Stienstra**

Universiteit van Amsterdam

**Matthias Talens**

Universiteit van Amsterdam



**1. Inleiding**

De hypothese bestaat dat vanuit het begingetal 4 alle natuurlijke getallen zijn te maken door slechts gebruik te maken van drie operatoren, te weten de vierkantswortel, faculteit en floor-functie. Het doel van dit onderzoek is deze hypothese te onderbouwen door zoveel mogelijk getallen onder de 10.000 te vinden die op deze wijze kunnen worden gemaakt.

Het getal 5 wordt dan gevonden door 4(!) 24 (!) 620448401733239000000000 (√) 787685471322(√) 887516(√) 942(√) 31(√) 5,5 (f) 5   
Waarbij != faculteit, √ = wortel en f = naar beneden afronden. Het is al snel te zien dat er enorm grote getallen gebruikt worden en dat de wortel vaak voorkomt. Uit het resultaat van 5 kan al snel 26 worden gevonden 5! 120 (√) 10,95 (√) 3,31 (f) 3 (1) 6 (!) 720 (√) 26,8 (f) 26

Voor dit paper leek het in eerste instantie nuttig om opdracht a van Number Crunching uit het hoofd op te schrijven, met als doel om de eventuele valkuilen die het oplossen van het probleem zouden vermoeilijken te ontdekken. Het kernprobleem bij de onderbouwing van de hypothese bleek als snel het feit dat iedere stap extra die dient te worden gemaakt bij het vinden van een gewenst getal een vergroting met factor 3 oplevert, bij een oneindige toestandsruimte van positieve getallen. Hierna is gebruik gemaakt van achtereenvolgens Brute Force. Dit paper sluit af met de redenen om het zelfontwikkeld algoritme te gebruiken voor het oplossen van de opdracht en de conclusies die uit deze oplossing vallen te trekken.

De onderzoeksvraag van dit paper is: Kunnen alle getallen gevonden worden door met begin getal 4 te manipuleren met slechts gebruik te maken van de vierkantswortel, faculteit en floor functie?

**2. Het Algoritme**

Resultaat: Een getal dat gevonden moet worden door 4 te manipuleren met de 3 operatoren

Sequentie: De reeks operatoren die nodig is om het resultaat te berekenen

Voorbeeld: Resultaat 5 kan berekend worden door de sequentie: !! √√√√√f toe te passen op begin getal 4.

In deze sectie wordt het algoritme beschreven dat gebruikt wordt voor het vinden van oplossingen voor *Number Crunching*. . Het basisidee voor het algoritme luidt als volgt:

1. Trek net zolang wortels totdat met behulp van een *floor* een resultaat gevonden kan worden waar naar gezocht wordt. Ga bij een gevonden resultaat door met worteltrekken om mogelijke nieuwe resultaten te vinden.
2. Indien er geen nieuwe resultaten meer worden gevonden, neem dan de faculteit van een al eerder gevonden resultaat en begin opnieuw bij stap 1.

Het algoritme beschouwd het getal ***c*** (*current number*). Door middel van een heuristiek wordt bepaald welke operator op dit getal wordt toegepast waardoor ***c*** verandert. De volgorde van operaties wordt bijgehouden en de gevonden resultaten met de bijbehorende sequenties worden opgeslagen. Voor het uitvoeren van het algoritme worden de volgende variabelen en verzamelingen gedefinieerd:

|  |  |  |
| --- | --- | --- |
|  | *Current Number* | Op dit getal worden de operaties toegepast. |
|  | *Resultaten* | Verzameling van gevonden resultaten. |
|  | *Faculteiten* | Verzameling van getallen waarvan de faculteit al een keer is genomen. |
|  | *Kleine f* | Ofwel het kleinste getal dat al gevonden is, waar nog niet de faculteit van is genomen. |
|  | *Streak* | Het hoogste resultaat in de langste streak vanaf 1. Wanneer bij voorbeeld de volgende resultaten gevonden zijn:  [ 1, 2, 3, 4, 6, 24, 30 ]. Dan is de langste streak vanaf 1 te definiëren als [1, 2, 3, 4]. Het hoogste resultaat in deze streak, ofwel de waarde van ***s***, is dan gelijk aan 4. |
| ­­­­­­ | *Maximum* | Het grootste getal waar naar gezocht wordt. Deze bepaald het stopcriterium; Als alle of een aangegeven deel van de getallen onder gevonden is, dan zal gestopt worden met zoeken. |
|  | *Fraction* | Deel van de getallen onder wat gevonden moet worden. Bijvoorbeeld 0.75 |
|  | *Stopcriterium* | *True* als het beoogde deel () van alle getallen onder is gevonden. |

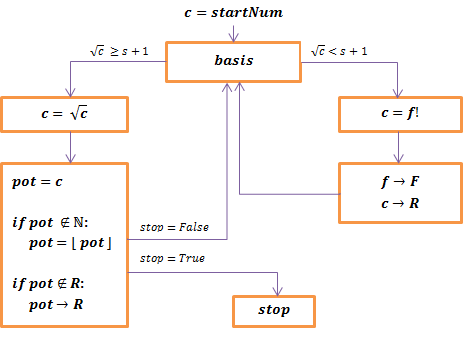
Het algoritme heeft een ***basis*** waar de beslissing wordt genomen welke operator op het huidige getal ***c*** moet worden toegepast. Na deze beslissing volgen een aantal stappen die worden uitgevoerd waarna, indien het stopcriterium nog niet is bereikt, weer naar de ***basis*** wordt teruggekeerd. De beslissing die gemaakt wordt, berust op het feit dat de mogelijkheid bestaat dat verder wortel trekken geen nieuwe resultaten zal opleveren. Dit gebeurt als de wortel van het huidige getal kleiner is dan ***s*** + 1. Wanneer dit zo is, is verder worteltrekken zinloos, en dient er een faculteit genomen te worden om tot nieuwe resultaten te kunnen komen.

**Mogelijkheid 1: √c ≥s+1**

De wortel dient genomen te worden dus ***c*** wordt aangepast tot de wortel van ***c***. Dit nieuwe huidige getal vormt een potentieel nieuw resultaat (). Wanneer niet integer is, dient een floor te worden toegepast en wanneer dit potentiele nieuwe resultaat nog niet is gevonden, wordt deze opgenomen in de verzameling resultaten. Als het stopcriterium nog niet is bereikt, zal opnieuw de keuze van operatie gemaakt moeten worden in de basis van dit algoritme.

**Mogelijkheid 2: √c<s+1**

Er dient een faculteit genomen te worden om tot nieuwe resultaten te kunnen komen. Het meest voor de hand liggend is het om de faculteit te nemen van het laagste gevonden resultaat waar nog niet de faculteit van is genomen (**)**. Het nieuwe huidige getal ***c*** zal dan de waarde van aannemen. Vervolgens zal ***f*** aan de verzameling ***F***, en ***c*** aan de resultatenverzameling ***R*** worden toegevoegd. Omdat het nieuwe resultaat, gevonden door de faculteit te nemen, naar grote waarschijnlijkheid niet het laatste gezochte resultaat is, zal er ten alle tijden teruggekeerd worden naar het basisblok. Mocht het voorkomen dat dit laatste niet zo is, dan zal er na het trekken van een wortel in de volgende stap geconstateerd worden dat aan het stopcriterium is voldaan en het algoritme alsnog worden beëindigd.



Tijdens het uitvoeren van dit algoritme dient de sequentie van operaties bijgehouden te worden. Dit is bij de uitleg achterwegen gelaten om de duidelijkheid te vergroten. Verder zijn er twee aannamen gemaakt die nader onderzocht moeten worden:

1. Wanneer er meerdere wortels achter elkaar worden getrokken dan maakt het voor het eindresultaat (getal wat aan het einde van deze sequentie wordt gevonden) niet uit als hier tussendoor een aantal keer wordt gefloord. Het gevolg van deze aanname is dat de floor operatie alleen gebruikt hoeft te worden bij het vinden van een nieuw resultaat en dat niet van elk gevonden resultaat ook nog de wortel getrokken hoeft te worden.
2. Wanneer een nieuw resultaat wordt gevonden dan gebeurt dit gelijk met de kortste sequentie. Het gevolg hiervan is dat bij het opnieuw vinden van een resultaat niet gecheckt hoeft te worden of deze een kortere sequentie heeft dan de huidige sequentie waarmee dit resultaat gevonden is. De heuristiek is ook op deze aanname gebaseerd.

Er is kort aangetoond dat de eerste aanname klopt. Voor de getallen 1 tm 30 000 is het verschil tussen (√√f) en (√f√f) berekend. Dit verschil is te allen tijde 0. Het verschil tussen deze twee manieren wordt alleen maar kleiner, dus het is een zeer logische aanname. Zie appendix voor de resultaten

De tweede aanname klopt niet. In de resultaten blijkt dat er zonder de tweede aanname kortere sequenties worden behaald.

**3. Resultaten**

Het algoritme berekend heel snel een groot deel van alle uitkomsten. Echter, de laatste uitkomsten duren onevenredig lang. Dit komt doordat een aantal getallen zeer moeilijk te vinden zijn, de sequentie is dan erg lang.

Uitkomsten <100

In de bovenstaande grafiek zijn voor de verschillende getallen (x as) de grootte van de sequentie geplot (y-as). Er is veel variatie in de uitkomsten, maar er is een duidelijke trend te zien dat hoe groter de uitkomst moet zijn hoe langer de sequenties worden. De langste sequentie is 347, deze hoort bij het getal 87.

De formule voor de trendlijn is als volgt: . Waarbij Y de lengte van de sequentie is en X de grootte van het resultaat.

Uitkomsten <200

Niet alle uitkomsten onder de 200 zijn gevonden. 182 en 197 bleken te lastig. De grootste gevonden sequentie blijft 347 bij getal 87. Verder is de formule van de trendlijn nu :   
Deze waarden zijn echter niet zuiver doordat er twee resultaten missen. Door de opzet van eht algoritme zijn dit resultaten met de langste sequentie. (Wanneer er voor deze waarden de sequentie van 347 wordt aangenomen wordt de vergelijking: (Y= )



Hiernaast is de volgorde van de gevonden getallen te zien. Het algoritme vind eerst 4, daarna 2 en 24 en eindigt bij 87. Er is weer trend dat hogere getallen later gevonden worden.

**4. Conclusie**

Er is geen antwoord gevonden op onderzoeksvraag. Echter er zijn wel zeer veel getallen gevonden en alles duid er op dat de rest van de getallen, gegeven genoeg tijd en rekenkracht, ook te vinden zijn. Verder bieden de oplossingen inzicht in het probleem en de getallen wereld. Een aantal oplossingen zijn vele male moeilijker te vinden dan andere en de lengte van de sequentie gaat omhoog naarmate het te vinden getal groter wordt.

**5. Discussie**

Een antwoord op de onderzoeksvraag kan niet worden bepaald met een algoritme, dit moet uit zuivere wiskunde komen. Hiervoor is Church-Rosser theorema interessant. Dit onderzoek zou uitgebreid kunnen worden door een diepe analyse uit te voeren op de resultaten. Er kan wellicht een verklaring gezocht worden voor de lengte van sequenties van bepaalde te vinden resultaten. Zo is het wellicht interessant om te kijken of de uitsplitsing naar priemgetallen meer informatie biedt. Verder kan er wellicht een optimaal algoritme gevonden worden, dit zou de zoektocht naar een verklaring simpeler maken.