**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.

Volgens deze methode is de optimale sequentie om het getal 5 te vinden bijvoorbeeld !!√√√√√f en de optimale sequentie om het getal 26 te vinden !!√√√√√f!√√f!!.

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 en een zelfontwikkeld algoritme om de opdrachten b, c en d op te lossen. 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.

NOG WAT PARAGRAFEN

**x. Het Artikel42-algoritme**

**!! Belangrijk om eerder in het verslag de terminologie uit te leggen. Uitleg over wat er bedoeld wordt met *resultaat* en *sequentie* is noodzakelijk !!**

In deze sectie wordt het algoritme beschreven dat gebruikt wordt voor het vinden van oplossingen voor *Number Crunching*. Dit algoritme is gebaseerd op eerdere bevindingen zoals beschreven in sectie SECTIENR. 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 beschouwt 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. |

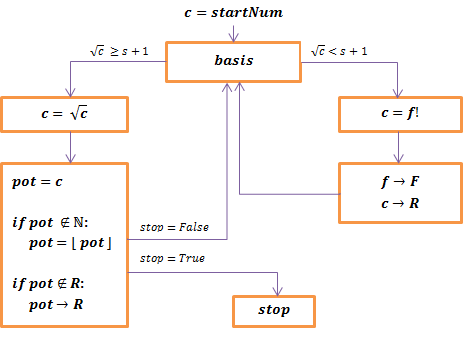
Het algoritme heeft een ***basis*** waar de beslissing wordt genomen welke operator op het huidige getal ***c*** moet worden toegepast. Na deze beslissing volgen een aantal stappen die worden uitgevoerd waarna, indien het stopcriterium nog niet is bereikt, weer naar de ***basis*** wordt teruggekeerd. 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:**

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:**

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.

Deze aannamen zorgen voor een efficiënter algoritme waardoor er snel korte sequenties gevonden kunnen worden om tot de resultaten te komen. Indien bewezen wordt dat deze aannamen kloppen, zal het algoritme ook de kortste sequenties opleveren.

STUKJE OVER HET ONDERZOEKEN VAN DE AANNAMEN