MachtenAlgoritme

June 3, 2022

1 Machten Algoritme

Door Matthijs Coster

Zie Pythagoras 61/6 Machten Algoritme en Vanggetallen, door Jaap Klouwen.

Wij zijn bijzonder geinteresseerd mocht je nog andere resultaten vinden. Stuur je resultaten naar matthijs at pyth punt eu.

In dit programma bekijken we het volgende probleem: - We splitsen een getal n in afzonderlijke cijfers. (nToCijfers) - We heffen deze cijfers tot zekere machten en tellen deze getallen bij elkaar op. (SomVanMachten) - Hier maken we een iteratief proces van. We eindigen als een getal voor de tweede keer in het rijtje dreigt te verschijnen.

In het artikel wordt geeindigd met een variant, waarbij de macht niet vooraf wordt gekozen, maar gelijk is aan het aantal cijfers van het getal.

We proberen antwoord te geven op de volgende vragen: - Wat is het gedrag van deze rijtjes? - Kunnen deze rijtjes naar "oneindig" gaan? - Zijn er getallen die zelf de som van de machten van de cijfers zijn? - Zijn er eindig veel lussen? - Hoe lang zijn de lussen? - Wat kan er worden gezegd over de bijzondere variant, waarbij de lengte van het getal de macht bepaalt? - Tenslotte kijken we naar literatuur over deze getallen.

2 Stukje wiskunde

Vanaf nu definieren we de functie $M_m(n)$ als de som van de m^{de}-machten van de cijfers van het getal n.

Getallen $10^{k-1} < n < 10^k$ bestaan uit k cijfers. We kiezen macht m. We schatten $M_m(n)$ af. De cijfers zijn kleiner dan 10 dus geldt $M_m(n) < k\dot{1}0^m$. Als we getallen tot 10 cijfers bekijken (dus $k \leq 10$) dan is $M_m(n) < k\dot{1}0^m < 10^{m+1}$.

In een volgende iteratie komt het resultaat evenmin boven 10^{m+1} uit (tenminste als $m < 10$. Na eindig veel iteraties moet er een eerder getal herhaald worden. Er ontstaat een lus.

Nu willen we een nauwkeurigere bovengrens uitrekenen. We gaan uit van getallen $10^{k-1} < n < 10^k$, met $k, m < 10$. Na de eerste iteratie hebben we $M_m(n) \leq k\dot{9}^m \leq 9^{m+1}$. We gaan de consequenties na voor de afzonderlijke machten m:

macht m	9^{m+1}	iteratie2	iteratie3
2	729	198	162

macht m	9^{m+1}	iteratie2	iteratie3
3	6561	2312	2188
4	59049	26500	26245
5	531441	296269	295246
6	4782969	3189375	3188710
7	43046721	33482970	33480911
8	387420489	344374024	
9	3486784401		

Het programma dat is gebruikt staat hier onder. Om alle lussen te bepalen moet je tot ongeveer deze grens. Je kunt er nog wel wat van afhalen, maar dat is best wel lastig! Probeer zelf maar eens!

Als $m \geq 10$, dan kan de boel anders gaan verlopen.

Met deze kennis gewapend gaan we de vragen beantwoorden.

2.0.1 Wat is het gedrag van deze rijtjes?

Zolang $m < 10$ zullen de rijtjes eindigen in een lus (of een enkel getal)

2.0.2 Kunnen deze rijtjes naar "oneindig" gaan?

Dat blijft hier nog een open vraag. Als $m < 10$ in ieder geval niet!

2.0.3 Zijn er getallen die zelf de som van de machten van de cijfers zijn?

Jaap klouwen geeft al uitgebreid antwoord. Voor het complete antwoord (voor $m < 10$) moet je zelf je computer laten rekenen.

2.0.4 Zijn er eindig veel lussen?

Voor $m < 10$ is het antwoord Ja! Dat volgt uit het feit dat alle rijtjes eindigen onder 10^{m+1}.

2.0.5 Hoe lang zijn de lussen?

Hiervoor moet je het programma uitvoeren.

2.0.6 Wat kan er worden gezegd over de bijzondere variant, waarbij de lengte van het getal de macht bepaalt?

Naar mate n groter wordt wordt het gedrag van de iteraties onvoorspelbaarder.

2.0.7 Tenslotte kijken we naar literatuur over deze getallen.

Na de programmatuur staat de literatuur.

[11]:
```
# We bepalen steeds de maximale iteratie van alle getallen kleiner dan 10^(m+1)
# En voeren een aantal iteraties uit, totdat geen verbetering meer optreedt.
# De maximale iteratie van alle getallen kleiner dan k is een getal kleiner dan␣
 ↪k
```

```
# met zoveel mogelijk negens. Daarom de procedure NegenGetal.

def nToCijfers(n):
    S = []
    while n > 0:
        S.append(n % 10)
        n = n // 10
    return S

def NegenGetalSet(n):
    S = nToCijfers(n)
    T = [S[-1]-1]
    ls = len(S)
    for i in range(ls-1):
        T.append(9)
    return T

def SetToN(T):
    som = 0
    lt = len(T)
    for i in range(lt):
        j = lt-i-1
        som = 10*som + T[i]
    return som

Machten = [0,1,2,3,4,5,6,7,8,9]
Bovengrens = []
for ii in range(8):
    i = ii+2
    for j in range(10):
        Machten[j] *= j
    s = 9*Machten[9]
    t = s-1
    S = [s]
    while t < s:
        T = NegenGetalSet(s)
        t = 0
        for k in T:
            t += Machten[k]
        S.append(t)
        if t < s:
            s = t
            t = s - 1
    print(i, S)
    Bovengrens.append(SetToN(T)+1)
print(Bovengrens)
```

```
2 [729, 198, 162, 162]
3 [6561, 2312, 2188, 2188]
4 [59049, 26500, 26245, 26245]
5 [531441, 296269, 295246, 295246]
6 [4782969, 3189375, 3188710, 3188710]
7 [43046721, 33482970, 33480911, 33480911]
8 [387420489, 344374024, 344374024]
9 [3486784401, 3486784913]
[100, 2000, 20000, 200000, 3000000, 30000000, 300000000, 3000000000]
```

[37]:
```python
def nToCijfers(n):
    S = []
    while n > 0:
        S.append(n % 10)
        n = n // 10
    return S

def SomVanMachten(n,m):
    S = nToCijfers(n)
    som = 0
    for s in S:
        som += Machten[m][s]
    return som

def IteratieSomVanMachten(n,m):
    T = [n]
    x = SomVanMachten(n,m)
    while x not in T:
        T.append(x)
        x = SomVanMachten(x,m)
    return T

def Lus(T,m): # bepaalt voor de lus de lengte en het kleinste element
    t = T[-1]
    U = [t]
    x = SomVanMachten(t,m)
    while x not in U:
        U.append(x)
        x = SomVanMachten(x,m)
    return (len(U), min(U))

BOUND = 20
Machten = [[1,1,1,1,1,1,1,1,1,1]] # Hier komen alle machten t/m 20 in te staan
for i in range(BOUND):
    X = []
    for j in range(10):
        X.append(Machten[i][j] * j)
```

```
    Machten.append(X)

MACHT = 20 # Kijk uit! Voor 20 duurt de berekening erg lang!
RANGE = 100000
Z = []
Z1 = []
LT = 0
li = 0
print("MACHT =", MACHT)
if MACHT < 5:
    RANGE = Bovengrens[MACHT-2]
for ii in range(RANGE-1):
    i = ii+2
    if (i % 100 == 0) & (i - li > 100000): # Voor het geval het niet zou werken...
        print(i)
    T = IteratieSomVanMachten(i,MACHT)
    lt = len(T)
    L = Lus(T,MACHT)
    if L[1] not in Z:
        Z.append(L[1])
        if L[0] == 1:
            Z1.append(L[1])
    if lt > LT:
        LT = lt
        print(i, len(T), L)
        li = i
Z1.sort()
print("Aantal lussen:", len(Z))
print("Aantal 1-lussen:", len(Z1))
print(Z1)
```

```
MACHT = 20
2 518 (210, 11257886613493064)
3 653 (210, 11257886613493064)
4 1101 (605, 14820873159435140)
12 1850 (605, 14820873159435140)
88 1858 (605, 14820873159435140)
112 1930 (605, 14820873159435140)
117 2433 (605, 14820873159435140)
566 2439 (605, 14820873159435140)
5556 2525 (605, 14820873159435140)
6789 2533 (605, 14820873159435140)
Aantal lussen: 6
Aantal 1-lussen: 1
[1]
```

Hieronder volgen twee programma's: - In een keer alle gevallen 2 t/m 10. - De bijzondere variant

[30]:
```
# In een keer alle gevallen 2 t/m 10.

def nToCijfers(n):
    S = []
    while n > 0:
        S.append(n % 10)
        n = n // 10
    return S

def SomVanMachten(n,m):
    S = nToCijfers(n)
    som = 0
    for s in S:
        som += Machten[m][s]
    return som

def IteratieSomVanMachten(n,m):
    T = [n]
    x = SomVanMachten(n,m)
    while x not in T:
        T.append(x)
        x = SomVanMachten(x,m)
    return T

def Lus(T,m): # bepaalt voor de lus de lengte en het kleinste element
    t = T[-1]
    U = [t]
    x = SomVanMachten(t,m)
    while x not in U:
        U.append(x)
        x = SomVanMachten(x,m)
    return (len(U), min(U))

BOUND = 20
Machten = [[1,1,1,1,1,1,1,1,1,1]] # Hier komen alle machten t/m 20 in te staan
for i in range(BOUND):
    X = []
    for j in range(10):
        X.append(Machten[i][j] * j)
    Machten.append(X)

for MACHTm in range(9):
    MACHT = MACHTm+2
    RANGE = 100000
    Z = []
```

```
    Z1 = []
    LT = 0
    print("MACHT =", MACHT)
    if MACHT < 5:
        RANGE = Bovengrens[MACHT-2]
    for ii in range(RANGE-1):
        i = ii+2
        T = IteratieSomVanMachten(i,MACHT)
        lt = len(T)
        L = Lus(T,MACHT)
        if L[1] not in Z:
            Z.append(L[1])
            if L[0] == 1:
                Z1.append(L[1])
        if lt > LT:
            LT = lt
            print(i, len(T), L)
    Z1.sort()
    print("Aantal lussen:", len(Z))
    print("Aantal 1-lussen:", len(Z1))
    print(Z1)
```

```
MACHT = 2
2 9 (8, 4)
3 13 (8, 4)
6 17 (8, 4)
Aantal lussen: 2
Aantal 1-lussen: 1
[1]
MACHT = 3
2 8 (1, 371)
6 11 (1, 153)
68 12 (1, 371)
177 14 (1, 153)
Aantal lussen: 9
Aantal 1-lussen: 5
[1, 153, 370, 371, 407]
MACHT = 4
2 25 (7, 1138)
3 36 (7, 1138)
14 54 (7, 1138)
457 55 (7, 1138)
2477 56 (7, 1138)
2557 57 (7, 1138)
2899 59 (7, 1138)
Aantal lussen: 6
Aantal 1-lussen: 4
[1, 1634, 8208, 9474]
```

```
MACHT = 5
2 46 (12, 24584)
7 62 (28, 244)
39 63 (22, 9045)
48 71 (22, 9045)
69 72 (22, 9045)
222 73 (22, 9045)
228 74 (22, 9045)
2337 76 (22, 9045)
Aantal lussen: 15
Aantal 1-lussen: 6
[1, 4150, 4151, 54748, 92727, 93084]
MACHT = 6
2 40 (10, 239459)
3 50 (30, 17148)
5 58 (30, 17148)
7 87 (10, 239459)
17 89 (10, 239459)
333 96 (10, 239459)
11237 99 (10, 239459)
25678 101 (10, 239459)
Aantal lussen: 6
Aantal 1-lussen: 1
[1]
MACHT = 7
2 110 (92, 80441)
4 123 (30, 376762)
29 127 (92, 80441)
179 148 (92, 80441)
278 151 (92, 80441)
1166 157 (92, 80441)
1244 160 (92, 80441)
Aantal lussen: 14
Aantal 1-lussen: 4
[1, 1741725, 9800817, 9926315]
MACHT = 8
2 78 (25, 8616804)
3 172 (154, 6822)
5 199 (154, 6822)
12 244 (154, 6822)
469 245 (154, 6822)
1126 250 (154, 6822)
1237 259 (154, 6822)
13444 260 (154, 6822)
Aantal lussen: 5
Aantal 1-lussen: 3
[1, 24678050, 24678051]
MACHT = 9
```

```
2 166 (30, 41179919)
9 182 (80, 32972646)
1335 184 (80, 32972646)
11124 186 (80, 32972646)
33588 187 (80, 32972646)
Aantal lussen: 18
Aantal 1-lussen: 5
[1, 146511208, 472335975, 534494836, 912985153]
MACHT = 10
2 58 (17, 62681428)
3 217 (123, 192215803)
12 252 (123, 192215803)
456 273 (123, 192215803)
Aantal lussen: 7
Aantal 1-lussen: 1
[1]
```

[36]:
```python
def nToCijfers(n):
    S = []
    while n > 0:
        S.append(n % 10)
        n = n // 10
    return S

def SomVanMachten(n,m):
    S = nToCijfers(n)
    som = 0
    for s in S:
        if m < 0:
            ls = len(S)
            som += Machten[ls][s]
        else:
            som += Machten[m][s]
    return som

def IteratieSomVanMachten(n,m):
    T = [n]
    x = SomVanMachten(n,m)
    while x not in T:
        T.append(x)
        x = SomVanMachten(x,m)
    return T

def Lus(T,m): # bepaalt voor de lus de lengte en het kleinste element
    t = T[-1]
    U = [t]
    x = SomVanMachten(t,m)
```

```
    while x not in U:
        U.append(x)
        x = SomVanMachten(x,m)
    return (len(U), min(U))

BOUND = 40
Machten = [[1,1,1,1,1,1,1,1,1,1]] # Hier komen alle machten t/m 20 in te staan
for i in range(BOUND):
    X = []
    for j in range(10):
        X.append(Machten[i][j] * j)
    Machten.append(X)

RANGE = [13557470126783088492, 6011488480156723240, 25711381864433939692,␣
 ↪15783424399270166217, 15943103192811834266, 1, 29010658302141123242,␣
 ↪13477675654490749289, 24485886132031337769, 25635438676514741290,␣
 ↪1555731942369692941, 50026711686521905344, 51020050847096430668,␣
 ↪26864398719037456714, 24558465846921732668, 1243971657514806314,␣
 ↪1320013497453391065, 62028544128396009117, 13473924139076253166,␣
 ↪2472839631142861591, 12476931008628761518, 27868607699430194519,␣
 ↪12487901672509755641, 12408203670648517818, 27853985268174739687,␣
 ↪48717866195346055242, 14554366783317937240, 26635991501948957642,␣
 ↪39021967732071823519, 39032840822526449817, 15779865803797166443,␣
 ↪51259425436506706492, 41095938076212745865, 26715878039654415467,␣
 ↪26639645457881077817, 15703729688969194343, 1323668545924618791,␣
 ↪4938551076221348716, 36650861708941991243, 15798143290490077766,␣
 ↪26712127609807555242, 37869046217005671917, 28017418893095514892,␣
 ↪40182201543097394669, 16932594378006358167, 13557469020302086217,␣
 ↪62180911715027435042, 3792560418770162343, 14474670983964642215,␣
 ↪15866871724496213418, 38869697701431794842, 15870622150856289242,␣
 ↪27944936741170086040, 1156772806418104090, 26791823409160850269,␣
 ↪1399613932862878442, 13477490413700919668, 38793564892109129143,␣
 ↪19321981172414750344, 52172972355189013469, 12400985628659876363,␣
 ↪12176233460039459288, 15634999073371627991, 14626941011115789342,␣
 ↪37724088510219337991, 25638906299114778568, 51027456335871726093,␣
 ↪13629855804795874441, 1327514353660327115]
MACHT = -1 # Kijk uit! Voor 20 duurt de berekening erg lang!
Z = []
Z1 = []
LT = 0
li = 0
print("MACHT =", MACHT)
for i in RANGE:
    for j in range(14):
        i = i * 10
        T = IteratieSomVanMachten(i,MACHT)
        lt = len(T)
```

```
        L = Lus(T,MACHT)
        if L[1] not in Z:
            Z.append(L[1])
            if L[0] == 1:
                Z1.append(L[1])
        if lt > LT:
            LT = lt
            print(i, len(T), L)
            li = i
Z1.sort()
print("Aantal lussen:", len(Z))
print("Aantal 1-lussen:", len(Z1))
print(Z1)
```

```
MACHT = -1
135574701267830884920 242 (10, 259)
13557470126783088492000000000000 257 (14, 18829)
135574701267830884920000000000000 288 (10, 259)
157834243992701662170000000000 294 (10, 259)
1578342439927016621700000000000 319 (10, 259)
157834243992701662170000000000000 328 (10, 259)
159431031928118342660000 362 (12, 5908997)
145543667833179372400000000000000000 386 (12, 5908997)
49385510762213487160000000000000000 401 (12, 5908997)
Aantal lussen: 18
Aantal 1-lussen: 6
[1, 370, 54748, 24678051, 32164049651, 35641594208964132]
```

3 Resultaten

3.0.1 Getallen die som van machten van cijfers zijn (anders dan 1):

n	lus lengte 1
3	1, 153, 370, 371, 407
4	1, 1634, 8208, 9474
5	1, 4150, 4151, 54748, 92727, 93084
7	1, 1741725, 9800817, 9926315
8	1, 24678050, 24678051
9	1, 146511208, 472335975, 534494836, 912985153
11	1, 32164049651
17	1, 35641594208964132

Deze getallen zijn verkregen met onderstaande programma. De laatste 2 regels kwamen uit de bijzondere variant.

4 Literatuur

We zoeken op “153, 370, 371, 407” in de OEIS (The On-Line Encyclopedia of Integer Sequences!) en vinden twee verwijzingen. De tweede verwijzing http://oeis.org/A252648 is de meest interessante:

n	lus lengte 1
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
2	0, 1
3	0, 1, 153, 370, 371, 407
4	0, 1, 1634, 8208, 9474
5	0, 1, 4150, 4151, 54748, 92727, 93084, 194979
6	0, 1, 548834
7	0, 1, 1741725, 4210818, 9800817, 9926315, 14459929
8	0, 1, 24678050, 24678051, 88593477
9	0, 1, 146511208, 472335975, 534494836, 912985153

We zaten redelijk dicht bij wat in de literatuur staat!