Even_en_Oneven

July 11, 2021

1 Even en Oneven

We gaan blokken stapelen. De blokken zijn er in twee soorten, laten we zeggen dat we blokken hebben met de tekst *even* en *oneven*. Leg een aantal blokken naast elkaar op de grond. Leg steeds blokken op een laag hoger, zo dat de helft van het nieuwe blok op een blok er onder rust en de andere helft op een belendende blok. Of er gekozen wordt voor het blok met *even* of *oneven* is hangt af van de twee blokken er onder. Zie tabel

Blok L	Blok R	nieuw blok
even	even	even
oneven	even	oneven
even	oneven	oneven
oneven	oneven	even

(In het artikel in Pythagoras staan er ook een paar plaatjes die bovengenoemde illustreren.)

1.1 Wat willen we onderzoeken?

Een aantal blokken (zeg n) liggen in een rij. Daar bovenop komen $n-1$ blokken te liggen, daarop $n-2$, en bovenop 1 blok. Bepaal voor elke n hoe je op de grond de blokken moet neerleggen, zodat er in het gehele bouwwerk zoveel mogelijk oneven blokken komen te liggen.

Stel dat de opdracht zou zijn om zoveel mogelijk even blokken te gebruiken, dan is de oplossing eenvoudig, want dan leg je op de grond ook even blokken en heb je een toren met louter even blokken.

Maar als je op de grond alleen oneven blokken legt dan zal de rest van de toren alleen even blokken bevatten. Je moet dus kiezen voor een mix van even en oneven blokken.

Hieronder staat het voorbeeld met 5 blokken.

|X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |XX even XX| |X oneven X| |XX even XX| |X oneven X|

Maar je kunt meer oneven blokken kwijt door op de grond de even en oneven blokken op een andere manier neer te leggen:

|X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X oneven X| |X

oneven X| |X oneven X| |XX even XX| |X oneven X| |XX even XX| |XX even XX| |X oneven X|

Nu is de vraag hoe moet je de *even* en *oneven* blokken rangschikken om in de gehele toren zoveel mogelijk *oneven* blokken te krijgen?

We gaan dat doen door alle mogelijkheden uit te proberen. Dus als we op de grond N blokken neerleggen dan kan in principe voor elk blok gekozen worden uit *even* of *oneven*, en dat zijn twee mogelijkheden. Voor N blokken zijn dat dus 2^N mogelijkheden. Qua tijd betekent dat dat we als N met 1 wordt vergroot de tijd verdubbelt. Dus we zullen snel tot een einde komen aan het uitproberen van alle mogelijkheden. Uiteraard heeft dat ook te maken met de snelheid van je computer.

Om alle mogelijkheden te kunnen uitproberen zijn er meerdere methodes. Bijvoorbeeld kunnen we gebruik maken van de procedure AlleDeelVerzamelingen(S), die alle deelverzamelingen van S genereert. Elk van deze deelverzamelingen staat voor een verzameling van *oneven* blokken. We bouwen de gehele toren af en tellen het totaal aantal oneven blokken.

We kunnen de opeenvolgende *even* en *oneven* blokken weergeven met nullen en enen. In feite staan er nu binaire getallen. Met de functie bin(n) kun je n omzetten in een binair getal. Vervolgens zet je dit binaire getal om in een array met nullen en enen. Deze methode maakt het programma wel iets trager.

1.2 Resultaten

n	maxoneven	aantal opl	oplossing
1	1	1	1
2	2	3	01
3	4	3	011
4	7	2	1011
5	10	3	01101
6	14	3	011011
7	19	2	1011011
8	24	3	01101101
9	30	3	011011011
10	37	2	1011011011
11	44	3	01101101101
12	52	3	011011011011
13	61	2	1011011011011
14	70	3	01101101101101
15	80	3	011011011011011
16	91	2	1011011011011011
17	102	3	01101101101101101
18	114	3	011011011011011011
19	127	2	1011011011011011011
20	140	3	01101101101101101101

De kolommen maximaal aantal oneven blokken en aantal oplossingen zijn terug te vinden op de website van Online Encyclopedia of Integer Sequences (OEIS):

- A007980 : max oneven blokken: [1, 2, 4, 7, 10, 14, 19, 24, 30, 37, 44, 52, 61, 70, 80, 91, 102, 114, 127, 140]
- A164359 : aantal oplossingen: [1, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3]

Je ziet ongetwijfeld wel de regelmaat. En dan is het natuurlijk de vraag gaat die regelmaat verder als n groter wordt. Kun je dat wiskundig bewijzen? Dat is best nog een lastige vraag, maar zonder het programma was het nog lastiger geweest!

Nu kun je je ook afvragen: wat gebeurt er als je in plaats van twee soorten blokken drie soorten blokken hebt. Blokken met waarde 0, 1 en 2. Boven op twee blokken komt een blok met de som van die twee blokken, en mocht die som drie of meer zijn, dan trek je er drie van af. Hoe ziet de rij blokken er uit als je de som van alle waarden van blokken wilt maximaliseren? En uiteraard, in plaats van drie soorten blokken kun je ook vier, vijf, zes, … soorten blokken kiezen.

[13]:
```
# invoer rij nullen en enen
# uitvoer aantal oneven blokken
import time
from itertools import chain, combinations

def AlleDeelVerzamelingen(verzameling):
    s = list(verzameling)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

# Het tellen van de oneven blokken in de toren
def nblokken(verzameling, n):
    rij = []
    for r in range(n):
        rij.append(0)
    for v in verzameling:
        rij[v] = 1
    n = len(rij)
    teller = 0
    for j in range(n):
        teller += rij[j]
    for ii in range(n):
        i = n - ii
        for j in range(i-1):
            rij[j] = (rij[j] + rij[j+1]) % 2
            teller += rij[j]
    return teller

AANTALBLOKKEN = 10
t0 = time.process_time()

maxoneven = 0
oplossing = []
aantalopl = 0
verzameling = []
```

```
for r in range(AANTALBLOKKEN):
    verzameling.append(r)
for s in AlleDeelVerzamelingen(verzameling):
    m = nblokken(s, AANTALBLOKKEN)
    #print(s,m)
    if m == maxoneven:
        oplossing.append(s)
        aantalopl += 1
    elif m > maxoneven:
        maxoneven = m
        oplossing = [s]
        aantalopl = 1
print(AANTALBLOKKEN, maxoneven, oplossing, aantalopl)
t1 = time.process_time()
print("Aantal secondes rekentijd", t1-t0)
```

```
10 37 [(0, 1, 3, 4, 6, 7, 9), (0, 2, 3, 5, 6, 8, 9)] 2
Aantal secondes rekentijd 0.027483711000000355
```

[17]:
```
# invoer rij nullen en enen
# uitvoer aantal oneven blokken
import time

def nblokken(rij):
    n = len(rij)
    teller = 0
    for j in range(n):
        teller += rij[j]
    for ii in range(n):
        i = n - ii
        for j in range(i-1):
            rij[j] = (rij[j] + rij[j+1]) % 2
            teller += rij[j]
    return teller

MAXAANTALBLOKKEN = 20
A = []
M = []
t0 = time.process_time()
for n in range(1,MAXAANTALBLOKKEN+1):
    aantalrijen = 2**n
    maxoneven = 0
    oplossing = []
    aantalopl = 0
    for k in range(aantalrijen):
        R = []
        c = k
```

```
        for i in range(n):
            R.append(c % 2)
            c = c // 2
        m = nblokken(R)
        if m > maxoneven:
            maxoneven = m
            oplossing = []
            aantalopl = 0
        if m == maxoneven:
            if aantalopl == 0:
                oplossing = R
            aantalopl += 1
    print(n, maxoneven, oplossing, aantalopl)
    M.append(maxoneven)
    A.append(aantalopl)
print("max oneven blokken:", M)
print("aantal oplossingen:", A)
t1 = time.process_time()
print("Aantal secondes rekentijd", t1-t0)
```

```
1 1 [1] 1
2 2 [1, 0] 3
3 4 [1, 1, 0] 3
4 7 [1, 0, 1, 1] 2
5 10 [1, 0, 1, 1, 0] 3
6 14 [1, 1, 0, 1, 1, 0] 3
7 19 [1, 0, 1, 1, 0, 1, 1] 2
8 24 [1, 0, 1, 1, 0, 1, 1, 0] 3
9 30 [1, 1, 0, 1, 1, 0, 1, 1, 0] 3
10 37 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1] 2
11 44 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
12 52 [1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
13 61 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1] 2
14 70 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
15 80 [1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
16 91 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1] 2
17 102 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
18 114 [1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
19 127 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1] 2
20 140 [1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0] 3
max oneven blokken: [1, 2, 4, 7, 10, 14, 19, 24, 30, 37, 44, 52, 61, 70, 80, 91,
102, 114, 127, 140]
aantal oplossingen: [1, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3]
Aantal secondes rekentijd 74.11239952599999
```

[35]:
```
# invoer rij nullen en enen
# uitvoer aantal oneven blokken
```

```
import time

# Het tellen van de oneven blokken in de toren
def nblokken(rij):
    n = len(rij)
    teller = 0
    for j in range(n):
        teller += rij[j]
    for ii in range(n):
        i = n - ii
        for j in range(i-1):
            rij[j] = (rij[j] + rij[j+1]) % 2
            teller += rij[j]
    return teller

MAXAANTALBLOKKEN = 20
A = []
M = []
t0 = time.process_time()
for nn in range(MAXAANTALBLOKKEN):
    n = nn + 1
    maxoneven = 0
    oplossing = []
    aantalopl = 0
    aantalmogelijkheden = 2**n
    for j in range(aantalmogelijkheden):
        s = str(int(bin(j)[2:]))
        ls = len(s)
        for i in range(n-ls):
            s = "0"+s
        rij = []
        for t in range(n):
            rij.append(int(s[t]))
        m = nblokken(rij)
        #print("B", rij, m)
        if m > maxoneven:
            maxoneven = m
            oplossing = [s]
            aantalopl = 1
        elif m == maxoneven:
            oplossing.append(s)
            aantalopl += 1
    print(n, maxoneven, oplossing, aantalopl)
    M.append(maxoneven)
    A.append(aantalopl)
print("max oneven blokken:", M)
print("aantal oplossingen:", A)
```

```
t1 = time.process_time()
print("Aantal secondes rekentijd", t1-t0)
```

```
1 1 ['1'] 1
2 2 ['01', '10', '11'] 3
3 4 ['011', '101', '110'] 3
4 7 ['1011', '1101'] 2
5 10 ['01101', '10110', '11011'] 3
6 14 ['011011', '101101', '110110'] 3
7 19 ['1011011', '1101101'] 2
8 24 ['01101101', '10110110', '11011011'] 3
9 30 ['011011011', '101101101', '110110110'] 3
10 37 ['1011011011', '1101101101'] 2
11 44 ['01101101101', '10110110110', '11011011011'] 3
12 52 ['011011011011', '101101101101', '110110110110'] 3
13 61 ['1011011011011', '1101101101101'] 2
14 70 ['01101101101101', '10110110110110', '11011011011011'] 3
15 80 ['011011011011011', '101101101101101', '110110110110110'] 3
16 91 ['1011011011011011', '1101101101101101'] 2
17 102 ['01101101101101101', '10110110110110110', '11011011011011011'] 3
18 114 ['011011011011011011', '101101101101101101', '110110110110110110'] 3
19 127 ['1011011011011011011', '1101101101101101101'] 2
20 140 ['01101101101101101101', '10110110110110110110', '11011011011011011011']
3
max oneven blokken: [1, 2, 4, 7, 10, 14, 19, 24, 30, 37, 44, 52, 61, 70, 80, 91,
102, 114, 127, 140]
aantal oplossingen: [1, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, 2, 3]
Aantal secondes rekentijd 80.949407861
```

[2]:
```
# Variant waarbij 2 wordt vervangen door kleuren. In eerste instantie 3.
# invoer rij nullen en enen en tweeen in
# uitvoer aantal oneven blokken
import time

def nblokken(rij, kleuren):
    n = len(rij)
    kleurenteller = 0
    for j in range(n):
        kleurenteller += rij[j]
    for ii in range(n):
        i = n - ii
        for j in range(i-1):
            rij[j] = (rij[j] + rij[j+1]) % kleuren
            kleurenteller += rij[j]
    return kleurenteller

AANTALBLOKKEN = 5
```

```
t0 = time.process_time()

kleuren = 3
aantalrijen = kleuren**AANTALBLOKKEN
maxkleurensom = 0
oplossing = []
aantalopl = 0
for k in range(aantalrijen):
    R = []
    c = k
    for i in range(AANTALBLOKKEN):
        R.append(c % kleuren)
        c = c // kleuren
    kleurensom = nblokken(R, kleuren)
    #print(R,m)
    if kleurensom > maxkleurensom:
        maxkleurensom = kleurensom
        oplossing = []
        aantalopl = 0
    if kleurensom == maxkleurensom:
        if aantalopl == 0:
            oplossing = R
        aantalopl += 1
print(AANTALBLOKKEN, maxkleurensom, oplossing, aantalopl)
t1 = time.process_time()
print("Aantal secondes rekentijd", t1-t0)
```

```
5 24 [2, 1, 2, 1, 2] 1
Aantal secondes rekentijd 0.002135540000000047
```

[20]:
```
# invoer rij kleuren
# uitvoer aantal maximale sommen van kleuren
# kleuren = 2 : 1,  2,  4,  7, 10,  14,  19,  24, 30, 37, 44, 52
# kleuren = 3 : 2,  5, 10, 16, 24,  33,  44,  56 (A001859)
# kleuren = 4 : 3,  8, 14, 23, 34,  44,  60,  76
# kleuren = 5 : 4, 11, 19, 31, 45,  61,  80, 103
# kleuren = 6 : 5, 14, 25, 40, 57,  78, 103, 134
# kleuren = 7 : 6, 17, 31, 51, 76, 104, 138, 177, 219, 267
# kleuren = 8 : 7, 20, 37, 58, 85, 116
# kleuren = 9 : 8, 23, 43, 65, 97, 130
import time

def nblokken(rij, kleuren):
    n = len(rij)
    crij = []
    for i in rij:
        crij.append(i)
```

```
    kleurenteller = 0
    for j in range(n):
        kleurenteller += crij[j]
    for ii in range(n):
        i = n - ii
        for j in range(i-1):
            crij[j] = (crij[j] + crij[j+1]) % kleuren
            kleurenteller += crij[j]
        #print(ii, rij, kleurenteller)
    return kleurenteller

MAXAANTALBLOKKEN = 6
kleuren = 9
A = []
M = []
t0 = time.process_time()
for n in range(1,MAXAANTALBLOKKEN+1):
    aantalrijen = kleuren**n
    maxkleurensom = 0
    oplossing = []
    aantalopl = 0
    for k in range(aantalrijen):
        R = []
        c = k
        for i in range(n):
            R.append(c % kleuren)
            c = c // kleuren
        kleurensom = nblokken(R, kleuren)
        if kleurensom > maxkleurensom:
            maxkleurensom = kleurensom
            oplossing = []
            aantalopl = 0
        if kleurensom == maxkleurensom:
            if aantalopl == 0:
                oplossing = R
                #print(n, maxkleurensom, R)
            aantalopl += 1
    print(n, maxkleurensom, oplossing, aantalopl)
    M.append(maxkleurensom)
    A.append(aantalopl)
print("max kleurensom blokken:", M)
print("aantal oplossingen:", A)
t1 = time.process_time()
print("Aantal secondes rekentijd", t1-t0)
```

```
1 8 [8] 1
2 23 [8, 8] 1
```

```
3 43 [8, 8, 8] 1
4 65 [8, 8, 8, 6] 3
5 97 [8, 8, 0, 8, 8] 1
6 130 [8, 8, 8, 0, 8, 8] 2
max kleurensom blokken: [8, 23, 43, 65, 97, 130]
aantal oplossingen: [1, 1, 1, 3, 1, 2]
Aantal secondes rekentijd 4.4617039449999965
```

[3]:
```python
# Voorbeeld van AlleDeelVerzamelingen gebruikmakende van itertools
# Bron: documentatie van itertools. Daar heet de procedure powerset.

from itertools import chain, combinations

def AlleDeelVerzamelingen(verzameling):
    "AlleDeelVerzamelingen([1,2,3]) --> () (1,) (2,) (3,) (1,2) (1,3) (2,3)␣
 ↪(1,2,3)"
    s = list(verzameling)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

for result in AlleDeelVerzamelingen([1, 2, 3]):
    print(result)
results = list(AlleDeelVerzamelingen([1, 2, 3]))
print(results)
```

```
()
(1,)
(2,)
(3,)
(1, 2)
(1, 3)
(2, 3)
(1, 2, 3)
[(), (1,), (2,), (3,), (1, 2), (1, 3), (2, 3), (1, 2, 3)]
```

[31]:
```python
for x in range(16):
    s0 = str(int(bin(x)[2:]))
    ls = len(s0)
    S = []
    for i in range(4-ls):
        s0 = "0"+s0
    for t in range(4):
        S.append(int(s0[t]))
    print(s0, S)
```

```
0000 [0, 0, 0, 0]
0001 [0, 0, 0, 1]
0010 [0, 0, 1, 0]
0011 [0, 0, 1, 1]
```

```
0100 [0, 1, 0, 0]
0101 [0, 1, 0, 1]
0110 [0, 1, 1, 0]
0111 [0, 1, 1, 1]
1000 [1, 0, 0, 0]
1001 [1, 0, 0, 1]
1010 [1, 0, 1, 0]
1011 [1, 0, 1, 1]
1100 [1, 1, 0, 0]
1101 [1, 1, 0, 1]
1110 [1, 1, 1, 0]
1111 [1, 1, 1, 1]
```

[ ]: