002 CodeGolfing - Zahlenspirale

Maximilian Noppel, max@noppelmax.online, 6. August 2018

***Zusammenfassung*—Die Aufgabe war es, ein Program mit möglichst wenig Zeichen zu schreiben, was die Zahlen der Eingabe n spiralförmig und in minimal großen rechtsbündigen Spalten ausgibt. Ich hoffe auch Programmierneulinge können mit meinen Beschreibungen etwas anfangen.**

I. Intro

Generell versuche ich beim CodeGolfen immer möglichst viele Zusammenhänge als Formeln auszudrücken und zu komprimieren. Das ergibt meistens kleineren Quellcode als der naive Ansatz, der einem als erstes einfällt.

II. Die Aufgabe

Die Aufgabe[1] war als CodeGolfing beim vspace.one[2] ausgeschrieben.

A. Eingabe

```
<program> <n>
```

wobei $0 < n < 1000$.

B. Aufgabe

Schreibe ein Programm das eine Zahl als Argument entgegen nimmt und daraus eine Spiralmatrix aufbaut. Alle Zahlen von 1 bis n laufen von oben links spiralförmig im Uhrzeigersinn um die Matrix in deren Zentrum. Alle Spalten müssen über '-' minimal- bündig gemacht werden. Sprich nur soviele '-' damit die jeweilige Spalte bündig wird. Zahlen $> n$ werden nicht ausgegeben und über notwendige '-' gepadded. Die Matrix ist immer quadratisch.

C. Beispiele

```
$python script.py 13
-1--2-3-4
12-13---5
11------6
10--9-8-7
$python script.py 2
1-2
---
$python script.py 5
1-2-3
----4
----5
```

[1] https://wiki.vspace.one/doku.php?id=treffen:002_codegolfing
[2] https://vspace.one

III. Berechne die Grösse des Quadrats d

d ist der Durchmesser des Quadrats.

$$d := \lceil \sqrt{n} \rceil$$

IV. Anzahl an Layer m

$$m := \lfloor d/2 \rfloor$$

V. Berechne die Anzahl an Elementen eines Layers

Im Folgenden ein Beispiel für $d = 5$. Die Zahlen bezeichen hier den jeweiligen Layer l.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sei d_l der Durchmesser in einem bestimmen Layer l.

$$d_l := d - 2l$$

Die Anzahl der Element $|E_l|$ in Layer l lässt sich über

$$|E_l| := \max(1, 4(d_l) - 4$$

berechnen. Das gilt für gerade wie auch für ungerade d. Das $\max$ wird für den letzen Layer eines ungeraden d benötigt. Es wird die vierfach Kantenlänge gewählt und die Ecken (die ja doppelt dabei sind) abgezogen.

VI. Berechne in welchem Layer Punkt (x, y) liegt

Der Layer in dem ein Punkt liegt lässt sich recht einfach durch eine Minimum und Maximum Verschachtelung berechnen.

$$\begin{bmatrix} 0,0 & 1,0 & 2,0 & 3,0 & 4,0 \\ 0,1 & 1,1 & 2,1 & 3,1 & 4,1 \\ 0,2 & 1,2 & 2,2 & 3,2 & 4,2 \\ 0,3 & 1,3 & 2,3 & 3,3 & 4,3 \\ 0,4 & 1,4 & 2,4 & 3,4 & 4,4 \end{bmatrix}$$

```
sub getLayer(x,y):
        return min( min(x,y), d-max(x,y)-1 )
```

Erstes Beispiel:

$$\begin{aligned} n &= 25 \\ d &= \sqrt{(25)} = 5 \\ getLayer(3,2) &= \min(\min(3,2), 5 - \max(3,2) - 1) \\ &= \min(2, 5 - 3 - 1) \\ &= \min(2,1) \\ &= 1 \end{aligned}$$

Zweites Beispiel:

$$\begin{aligned} getLayer(1,0) &= \min(\min(1,0), 5-\max(1,0)-1) \\ &= \min(0, 5-1-1) \\ &= \min(0,3) \\ &= 0 \end{aligned}$$

VII. Berechne den Wert der Zelle $(x, y) \in E$

Der tatsächlich Wert einer Zellen wird berechnet, indem alle umliegenden Layer zusammen addiert werden.

$$\sum_{l=0}^{\text{getLayer}(x,y)-1} |E_l|$$

Es wird ein d_l berechnet das der Länge und Breite des aktuellen Layers entspricht. Pro Layer werden 2 abgezogen. Anschließend werden x und y auf den Layer skaliert. Alle umliegenden Layer werden 'ausgeblendet'. Die Zelle kann jetzt nur noch auf dem äußersten Layer dieses kleineren Quadrates liegen. Wir stellen über eine If-Kette fest an welcher Kante die Zelle liegt und rechnen entsprechend im Kreis herum die vorhergehenden Punkte auf dem aktullen Layer aus.

Hier ein Beispiel mit $d_l = 3$.

$$\begin{bmatrix} x & x & x \\ 4d'-y-3 & & d'+y \\ 3d'-x-2 & 3d'-x-2 & d'+y \end{bmatrix}$$

```
def get(x,y):
        sum := 0
        l := getLayer(x,y)

        for i in 0..l-1:
                sum += #element on layer i

        dl = d - 2*l

        y = y - l
        x = x - l

        if y == 1:
                return sum + x
        if x == dl:
                return sum + dl -1
        if y == dl:
                return sum + 3dl-x-2
        if x == 1:
                return sum + 4dl-y-3
```

VIII. Berechne das Maximum in einer Spalte

Das Maximum einer Spalte lässt sich sehr einfach berechnen in dem man alle Werte einer Spalte vergleicht. Dieses Maximum wird für die Formatierung verwendet.

```
def getMax(x):
        m := 0
        for y..d:
                m = max(m,get(x,y)
        return m
```

IX. Kompletter Algorithmus

Abschließend müssen wir das in eine Form gießen und in Schleifen packen. Wir iterieren über jede Zelle und rechnen jedes mal alles aus.

```
d := ceil(sqrt(n))
for y in 0..d:
        for x in 0..d:
                j = get(x,y)
                print '-'x(log10(getMax())-log10(
                    j))
                if j <= n:
                        print j
                else
                        print '-' x log10(j)
```

X. Lösung in Perl

Das alles führt, zusammengebaut, zu der folgende Lösung in Perl, die minimiert 470 Bytes ergibt.

```
use POSIX;

$n=$ARGV[0];
$d=ceil sqrt$n;

sub v($$){$_[$_[0]<$_[1]]}
sub c($$){$_[$_[0]>$_[1]]}
sub len{1+length@_[0]}

sub g {
        ($x,$y)=@_;
        $s=0;
        $l=c(c($x,$y),$d-v($x,$y)-1);
        $s+=v(1,4*$d-8*$_-4) for 0..$l-1;

        $u=$d-2*$l;
        $a=$y-$l;
        $b=$x-$l;

        $s+=$a==0?($b+1):$b==$u-1?($u+$a):$a==$u
            -1?(3*$u-$b-2):$b==0?(4*$u-$a-3):0;
        $s>$n?0:$s;
}

for(0..$d-1) {
        $z=$_;
        for(0..$d-1) {
                $q=$_;
                $m=0;
                $g=g($q,$_),$m = $g>$m?$g:$m for
                    0..$d-1;
                $j=g($_,$z);
                print"-"x(($_!=0)+len($m)-len($j)
                    );
                print$j!=0?$j:'-';
        }
        print$/;
}
```

Hier kann man sehen, dass von den beschriebenen Funktionen nur noch ein übriggeblieben ist. Alle anderen wurden nur an einer Stelle verwendet und damit integriert.