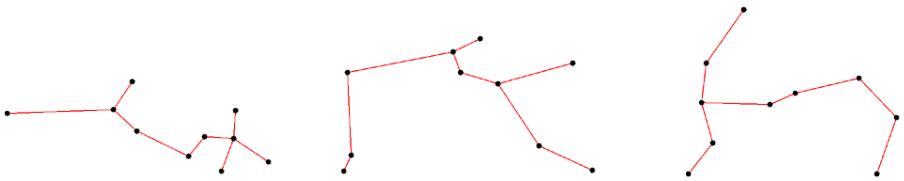
**Computational Thinking in de klas**
**Programmeren in Python**

**Koen Stulens & Bert Wikkerink**

**Aanvulling op het artikel in Euclides 97-4, de ICT-special.**

## Prim-algoritme

Een Prim-algoritme is een algoritme om de minimale (gewogen) opspannende boom van een graaf te bepalen; m.a.w. de boom met minimale gewicht, kost of afstand. We coderen een Prim-algoritme met als gewicht de Euclidische afstand tussen de punten.
Voor een graaf tekenen we de Euclidisch minimale opspannende boom (EMST – Euclidean minimum spanning tree). Een EMST verbindt een verzameling punten met lijnen zodat de totale lengte van de lijnen minimaal is en zodat ieder punt bereikt kan worden vanuit ieder ander punt door deze lijnen te volgen.



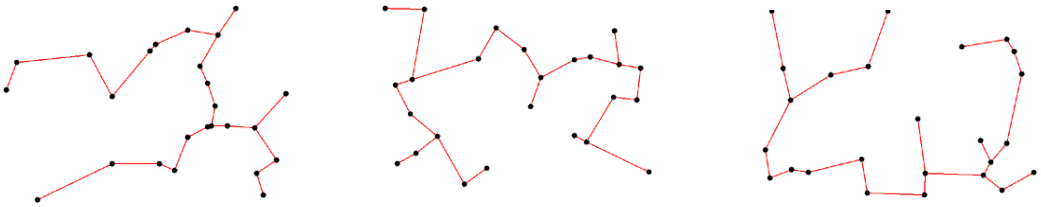
We bouwen het algoritme als volgt op voor een verzameling van punten:

1 Start een boom met een willekeurig punt *p* (verbonden) en een lijst met punten die nog niet toegevoegd zijn aan de boom (niet verbonden). Bij de start zijn dit alle punten uitgezonderd het willekeurig gekozen punt *p*.

2 Zoek voor dit punt *p* het punt *q* uit de lijst van niet verbonden punten waarvoor geldt dat de afstand minimaal is.

   a Verbind de punten *p* en *q*,
   b verwijder *q* van de lijst van niet verbonden punten en
   c voeg *q* toe aan de verbonden punten van de boom.

3 Herhaal stap 2 voor alle verbonden punten tot de er geen niet verbonden punten meer zijn.



De structuur van een programma voor het tekenen van een EMST-boom kan er als volgt uitzien:

DEFINIËREN VAN OBJECTEN EN FUNCTIES

```
class Punt(x,y):
# Definieer een punt a.h.v.
de coördinaten
# van een pixel
    def teken(self):
```

AT RANDOM GENEREN VAN LIJST PUNT-OBJECTEN

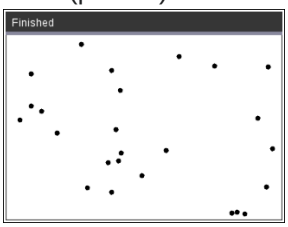
```
w,h = get_screen_dim()
r,n = (3,25)

punten = [ ]
for i in range(n):
    x = randint(r,w-r)
    y = randint(r,h-r)
```

```
# Definieer een methode teken die het
# punt tekent als zwarte cirkelschijf

def afstand(p,q):
# Definieer de afstand tussen twee punten

def lijn(p,q):
# Teken een rode lijn tussen twee punten

def emstboom():
# Bepaal de EMST-boom van een
# verzameling punten.
# Kies eerst een willekeuring punt en
# plaats dit punt in een lijst verbonden.
# Plaats al de andere punten in een
# tweede lijst niet_verbonden.
# Breidt de lijst verbonden uit volgens
# de regels van Stap 2 tot de lijst
# niet_verbonden leeg is.
# Verbind uiteindelijk de punten uit de
# lijst verbonden.
```

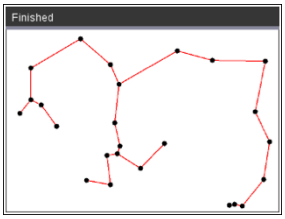
♦ ♦punten.append(Punt(x,y))

TEKENEN VAN PUNTEN & BOOM

teken(punten)




emstboom(punten)




Voor de code van het Prim-algoritme verwijzen we naar www.wil-depython.nl – Deel 5 Verdieping.

### Physical computing

Het combineren van het computationeel denk- en programmeerwerk met het lezen en sturen van een micro-controller verhoogt de motivatie van leerlingen om aan de slag te gaan met programmeren.

We bekijken twee MCU-voorbeelden: een TI-Innovator™ Hub en een BBC micro:bit.

### Binaire getallen

Een eenvoudig algoritme om een natuurlijk getal *n* om te zetten in een binair getal is het getal *n* te delen door twee en vervolgens doorgaan met het quotiënt te delen door 2 tot het quotiënt nul is. Bij iedere stap onthouden we rest en noteer deze in omgekeerde volgorde.

$$12 = 6 \times 2 + 0$$
$$6 = 3 \times 2 + 0$$
$$3 = 1 \times 2 + 1$$
$$1 = 0 \times 2 + 1$$
$$12 = 1100$$

```
n = int(input("Natuurlijk getal: "))

d = n; b1 = [ ]; b2 = " "
while d//2 != 0:
    b1.append(d%2)
    d = d//2
b1.append(d%2)
b1.reverse()

for d in b1:
    b2 += str(d)
print("{} = {}".format(n,b2))
```

$n = 12$

De resten worden bewaard in de lijst b1:

b1 = [1,1,0,0]

en dan omgezet in een string b2:

b2 = 1100

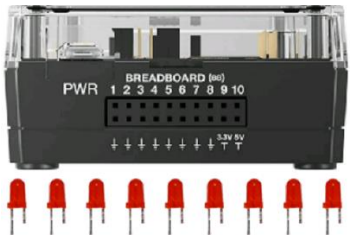
voor de uiteindelijke output

12 = 1100

De uitdaging is de binaire conversie voor te stellen d.m.v. LEDs. We sluiten negen LEDs aan op de breadboard-poorten BB1 t.e.m. BB9 van een TI-Innovator™ hub.

Iedere LED stelt een cijfer voor van de binaire conversie. Hiermee kunnen we de natuurlijke getallen 0 t/m 511 voorstellen:

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8 = 511$$




Hiervoor passen we de bovenstaande code als volgt aan:

```
from ti_hub import *

n = int(input("Natuurlijk getal: "))

bin = [ ]
while n//2 != 0:
    bin.append(n%2)
    n = n//2
bin.append(n%2)

for i in range(0,9-len(bin)):
  bin.append(0)

bin.reverse()

ledarray = [ ]
for i in range(9):
    ledarray.append(led("BB " + str(i+1)))

for i in range(9):
    if binary[i] == 1:
        ledarray[i].on()
```

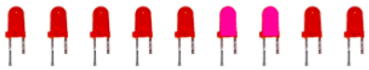
De lijst met binaire cijfers wordt vooraan aangevuld met nullen tot negen elementen.

Bovendien wordt een lijst gedefinieerd met als elementen de LEDs aangesloten op de poorten BB1 t.e.m. BB9.

Afhankelijk van de waarde van een cijfer van de binaire conversie, 1 of 0, worden de LEDs aangezet (1 = ON):

Getal 0 <= ... <= 511: 12

[0, 0, 0, 0, 0, 1, 1, 0, 0]



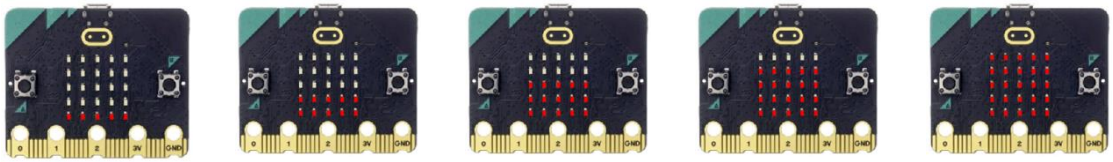
Getal 0 <= ... <= 511: 341

[1, 0, 1, 0, 1, 0, 1, 0, 1]

**Geluidsmeter**

In het volgend voorbeeld combineren we de 5 × 5 LED display en de microfoon van een BBC micro:bit v2 om een geluidsmeter te programmeren.

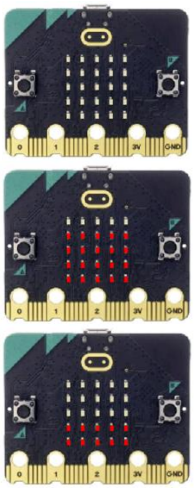
Level 1 Level 2 Level 3 Level 4 Level 5



Eerste stap is het definiëren van de verschillende levels:

```
from microbit import *
```

```
while get_key() != "esc":
    m = microphone.sound_level()
    if m > 60:
        i = 5
    elif m > 30:
        i = 4
    elif m > 20:
        i = 3
    elif m > 10:
        i = 2
    elif m > 4:
        i = 1
    else:
        i = 0
    display.show(level[i])
```



```
lev0 = Image(
    "00000:"
    "00000:"
    "00000:"
    "00000:"
    "00000:")
```

```
lev1 = Image(
    "00000:"
    "00000:"
    "00000:"
    "00000:"
    "99999:")
```

```
lev2 = Image(
    "00000:"
    "00000:"
    "00000:"
    "99999:"
    "99999:")
```

```
lev3 = Image(
    "00000:"
    "00000:"
    "99999:"
    "99999:"
    "99999:")
```

```
lev4 = Image(
    "00000:"
    "99999:"
    "99999:"
    "99999:"
    "99999:")
```

```
lev5 = Image(
    "99999:"
    "99999:"
    "99999:"
    "99999:"
    "99999:")
```

Het vervolg van het programma is, het lezen van de microfoon en afhankelijk van deze waarde het gewenste level af te beelden op de 5 × 5 LED display.