Self-avoiding walks

in het hexagonale vlak

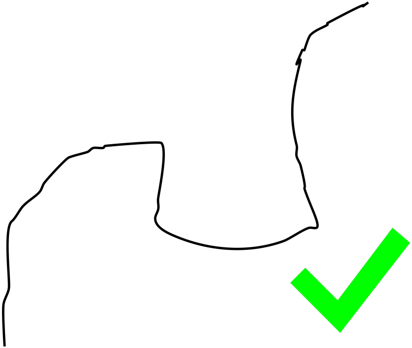
Thomas van Maaren

24 mei 2022

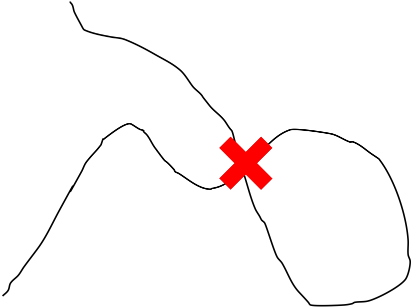
Inhoud

- Wat zijn self-avoiding walks?
- Toepassing van self-avoiding walks
- Lengte van self-avoiding walks
- Mogelijke self-avoiding walks
- Hongingraat
- Algoritme voor het bepalen van mogelijke walks
- Demonstratie algoritme (Turtle graphics)
- Ondergrenzen/bovengenzen
- Antwoord op de vraag
- Connective constant

Wat zijn self-avoiding walks?

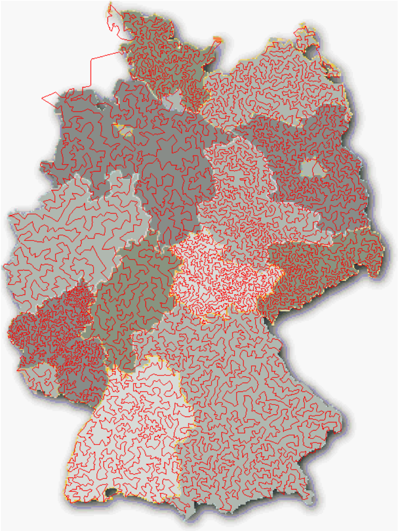


Wel een self-avoiding walk



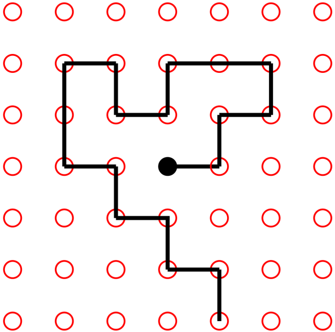
Geen self-avoiding walk

Toepassingen van self-avoiding walks

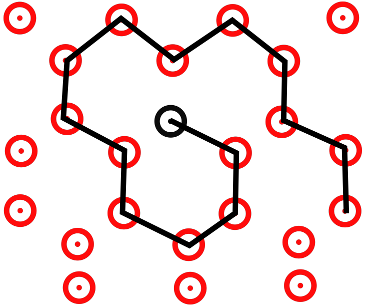


Figuur: Optimale weg door 15112 duitse steden. Bron:University of Waterloo

Lengte van een Avoiding walk



Lengte 18 op een vierkant rooster



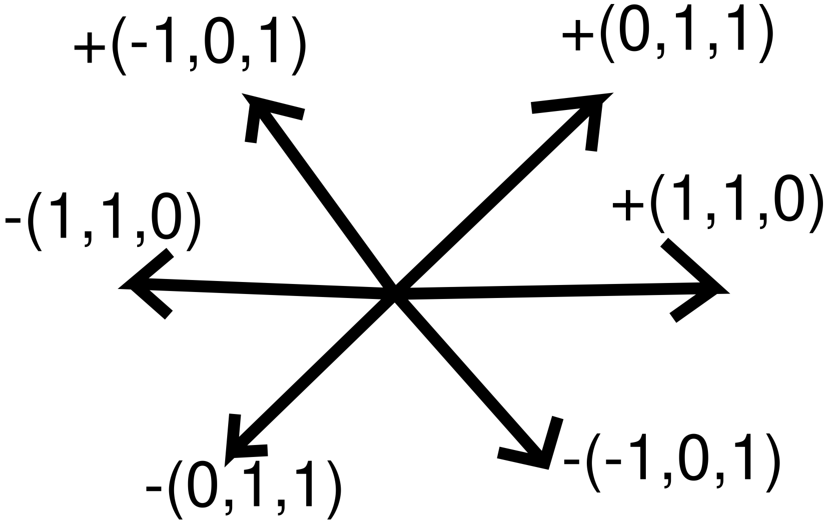
Lengte 13 op een honingraat

Mogelijke self-avoiding walks

Zij $n \in \mathbb{N}$. Dan definieren c_n als het aantal mogelijke self-avoiding walks met lengte n die vanaf de oorsprong op een honingraat gemaakt kunnen worden.

- $c_1 = 3$
- $c_2 = 3 \cdot 2 = 6$
- $c_3 = 3 \cdot 2^2 = 12$
- $c_4 = 3 \cdot 2^3 = 24$
- $c_5 = 3 \cdot 2^4 = 48$
- $c_6 = 3 \cdot 2^5 - 6 = 96 - 6 = 90$

Hongingraat



Figuur: Coördinaten op een honingraat

Algoritme voor het bepalen van mogelijke walks (1)

```
import numpy as np

n=20
visited = np.zeros((2*n+1,2*n+1,2*n+1),dtype =bool)

def go(x,y,z,stap,visited,n,even):
    if(visited[x,y,z]):
        return 0 #ga terug
    elif stap==n:
        return 1
    visited[x,y,z] = True
```

Algoritme voor het bepalen van mogelijke walks (2)

```
count = 0
if(even):
    count+= go(x+1,y+1,z,stap+1,visited,n, not
                                    even)
    count+= go(x-1,y,z+1,stap+1,visited,n, not
                                    even)
    count+= go(x,y-1,z-1,stap+1,visited,n, not
                                    even)
else:
    count+= go(x-1,y-1,z,stap+1,visited,n, not
                                    even)
    count+= go(x+1,y,z-1,stap+1,visited,n, not
                                    even)
    count+= go(x,y+1,z+1,stap+1,visited,n, not
                                    even)
visited[x,y,z] = False
return count
```

Demonstratie

Bovengrenzen en ondergrenzen

We kunnen zeggen dat c_n de volgende ondergrenzen en bovengrenzen heeft.

$$\sqrt{2}^n \leq c_n \leq 3 \cdot 2^n$$

Aangezien we een pad op kunnen splitsen zien we ook dat

$$c_{m+n} \leq c_m c_n$$

Vraag aan jullie: Convegeert de rij $c_n^{\frac{1}{n}}$?

Antwoord op de vraag

Het convergeert wel. We zien eerst dat $\sqrt{2} \leq c^{\frac{1}{n}}$, dus heeft het een infimum die we noteren met L. Zij $\epsilon > 0$. We weten dan dat er een $k \in \mathbb{N}$ zodat $c_k^{\frac{1}{k}} \leq L + \epsilon$. dat $n = p_n k + q_n$ voor een $p_n, q_n \in \mathbb{N}$, waarbij $0 \leq q_n \leq p_n$. We zien nu dat

$$c_n^{1/n} = c_{p_n k + q_n}^{1/n} \leq (c_k^{p_n} c_{q_n})^{1/n} = (c_k^{\frac{1}{k}})^{\frac{p_n k}{n}} c_{q_n}^{\frac{1}{n}}$$

We zien dat $\lim_{n\to\infty} \frac{p_n}{n} = 1$ en $\lim_{n\to\infty} \frac{1}{n} = 0$, dus

$$\lim_{n\to\infty} (c_k^{\frac{1}{k}})^{\frac{p_n k}{n}} c_{q_n}^{\frac{1}{n}} = (c_k^{\frac{1}{k}})^1 c_{q_n}^0 = c_k^{\frac{1}{k}}$$

Dus is er een $N \in \mathbb{N}$ waar voor alle $n > N$ geldt $(c_k^{\frac{1}{k}})^{\frac{p_n k}{n}} c_{q_n}^{\frac{1}{n}} \leq L + \epsilon$.
Dus voor alle $n > N$ geldt dan ook dat $c_n^{\frac{1}{n}} \leq L + \epsilon$

Connective constant

De connective constant is $lim_{n\to\infty} c_n^{\frac{1}{n}}$. Voor een bijenraad is dit

$$\sqrt{2+\sqrt{2}} \approx 1.8478$$

Voor andere roosters is dit ongeveer

- Driehoeksrooster: 4.15079
- Vierkantrooster: 2.6381585