Zelfmijdende wandelingen

Rob Bisseling

Mathematisch Instituut, Universiteit Utrecht
Samen met Gerard Barkema (UU Informatica), Raoul Schram (ENS Lyon)


Rob

Gerard


Raoul

College Wat is wiskunde 12 oktober 2017

Universiteit Utrecht

Inhoud

Introductie zelfmijdende wandelingen

Recursieve algoritmen

Nieuwe methode: lengteverdubbeling

Resultaten

Conclusie

Universiteit Utrecht

Nieuwsgierig wandelen

Bron: my-new-york.com, nyc-architecture.com

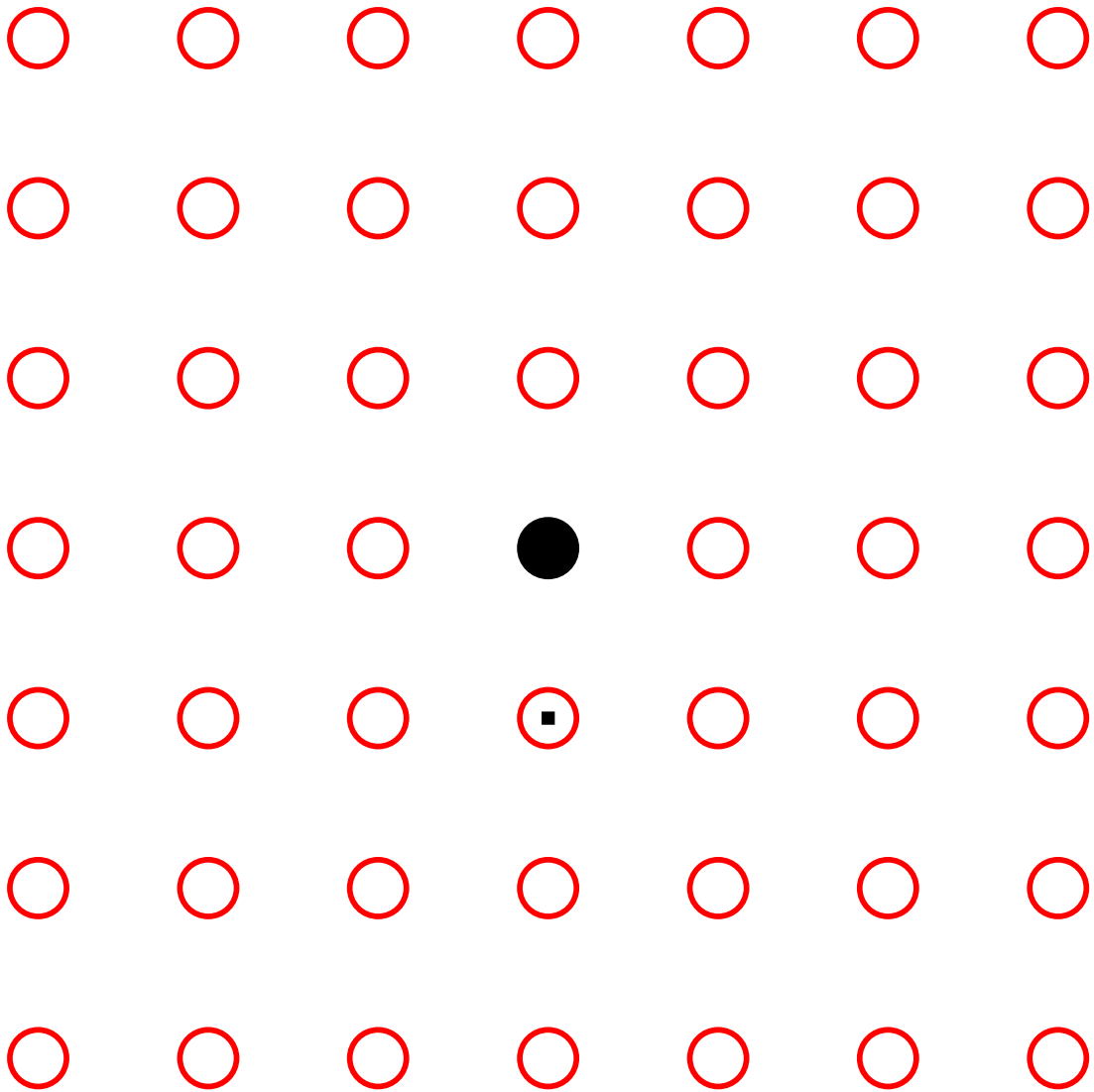
Universiteit Utrecht

Definitie zelfmijdende wandeling

- Een zelfmijdende wandeling (self-avoiding walk, SAW) is een wandeling op een rooster waarbij wij nooit op de dezelfde plek terugkeren.
- We beginnen in de oorsprong.
- De lengte van een wandeling, d.w.z. het aantal stappen, noemen we N.

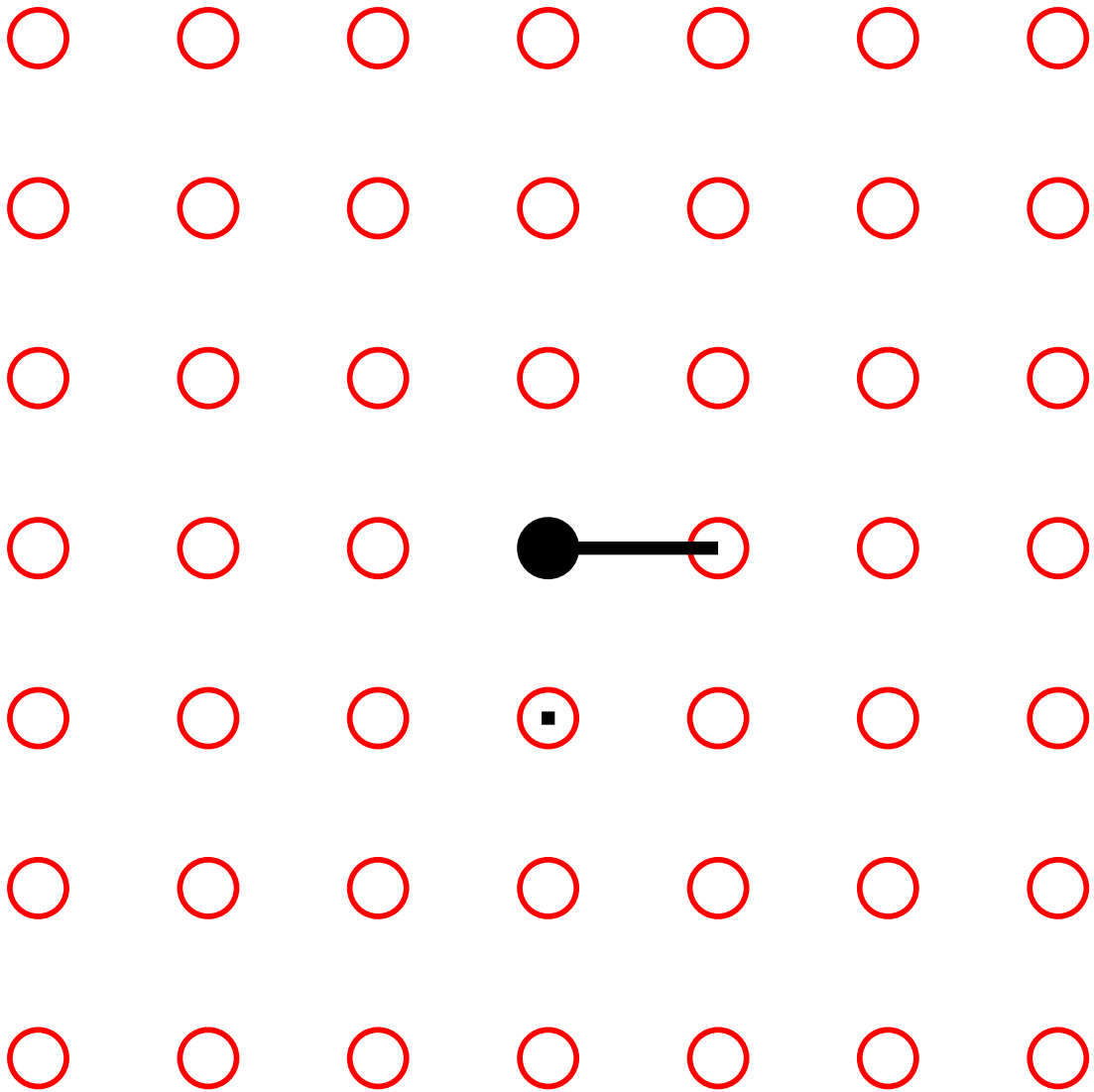
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 0 in 2D



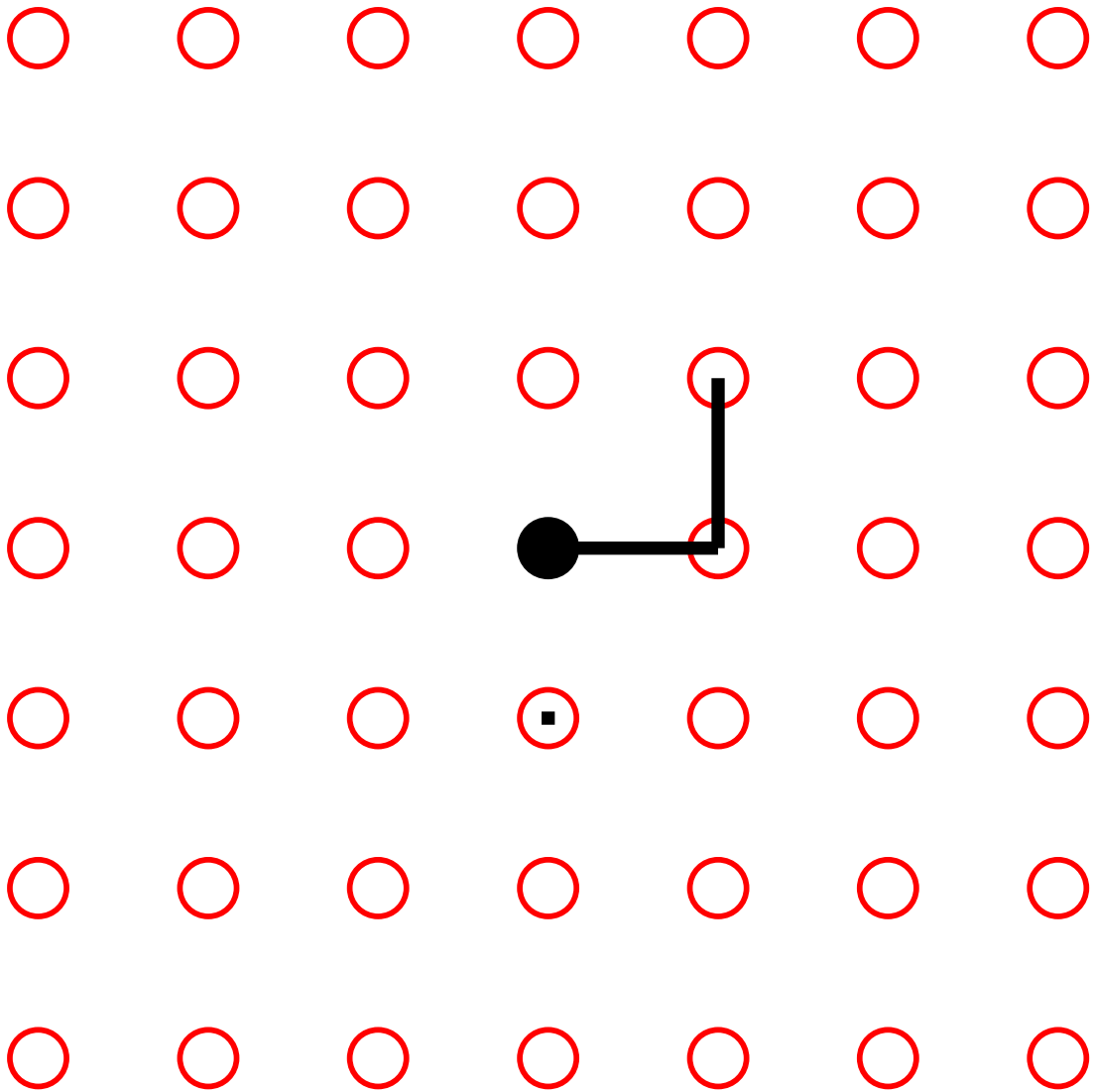
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 1 in 2D



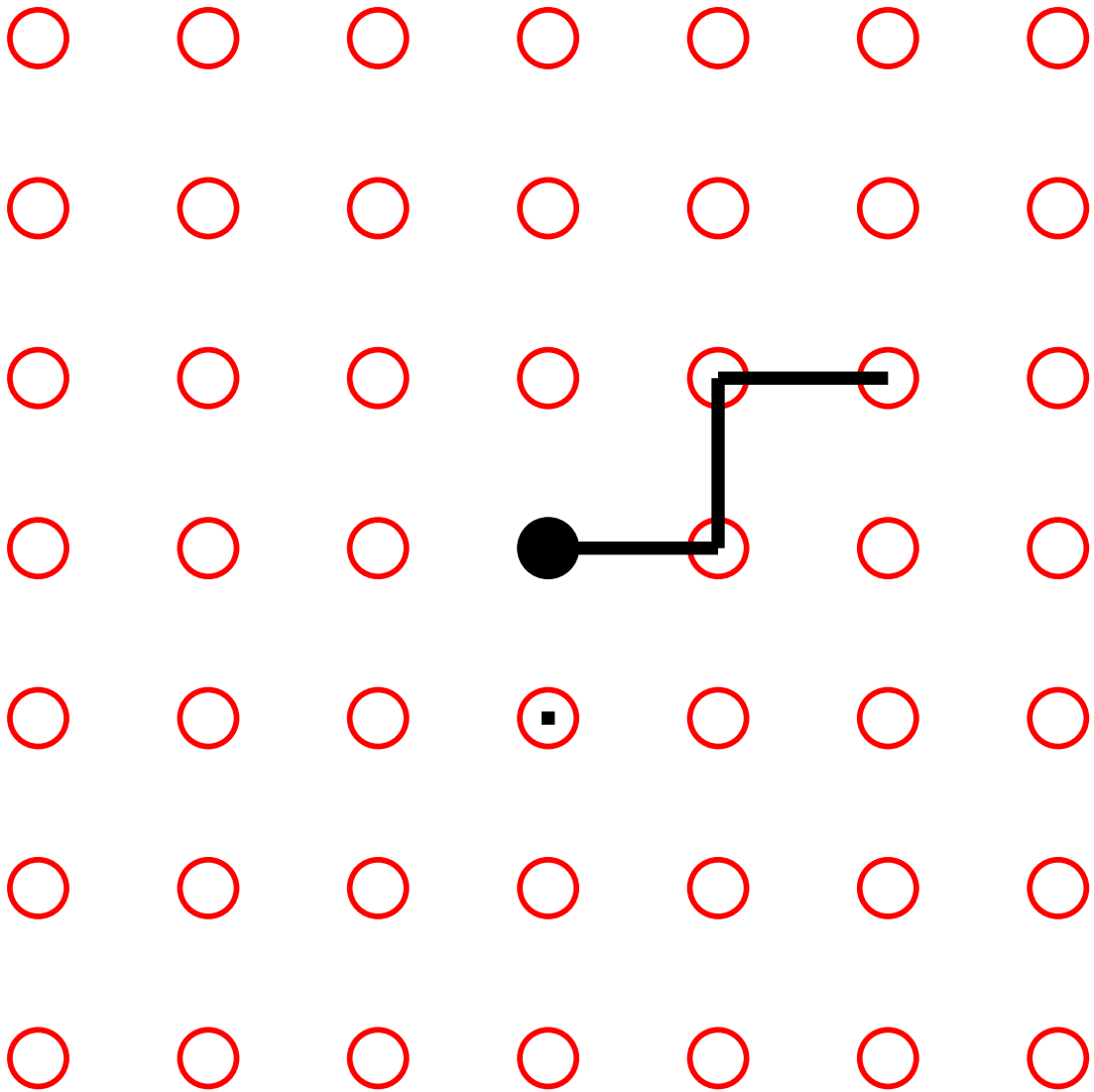
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 2 in 2D



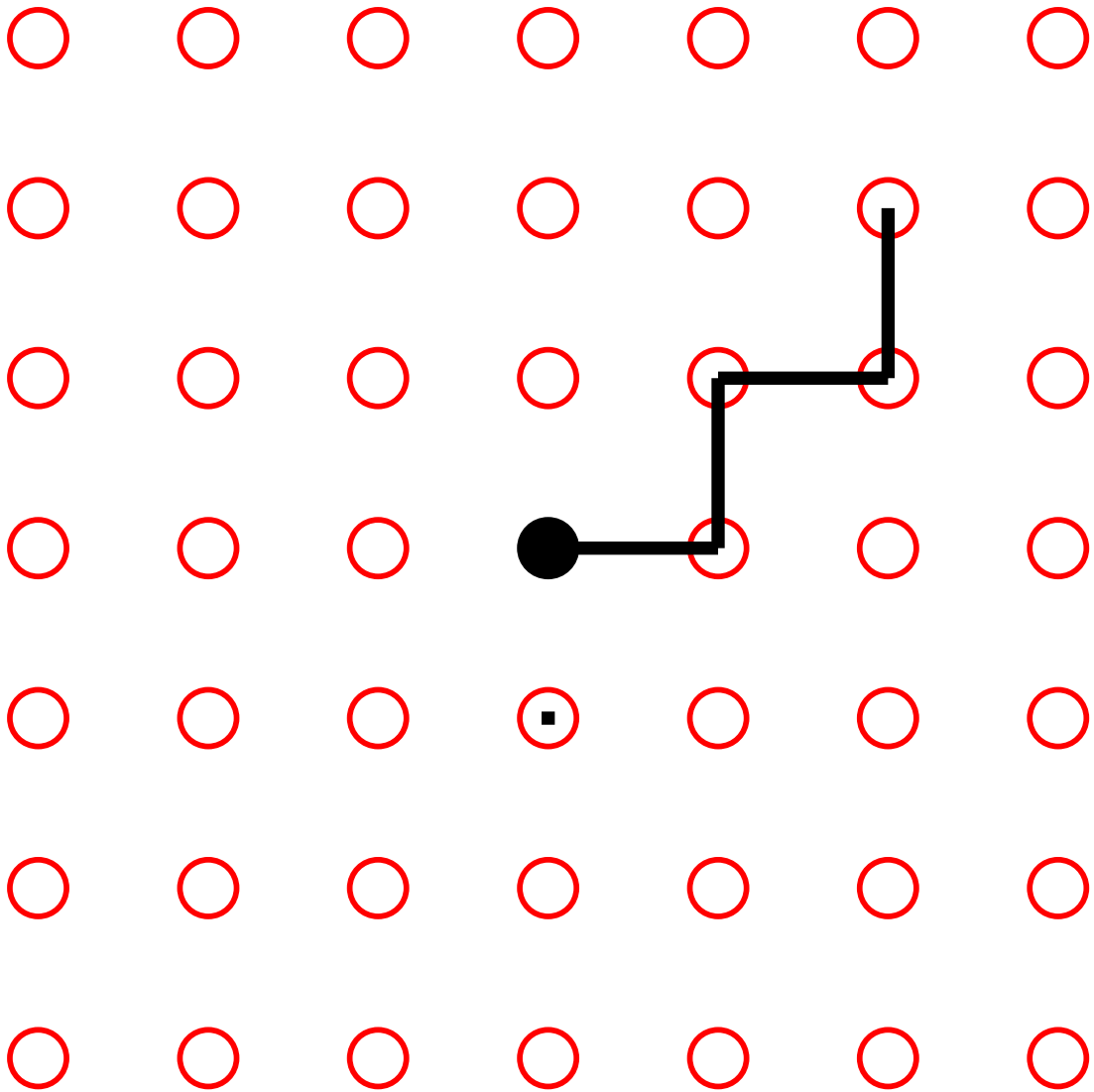
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 3 in 2D



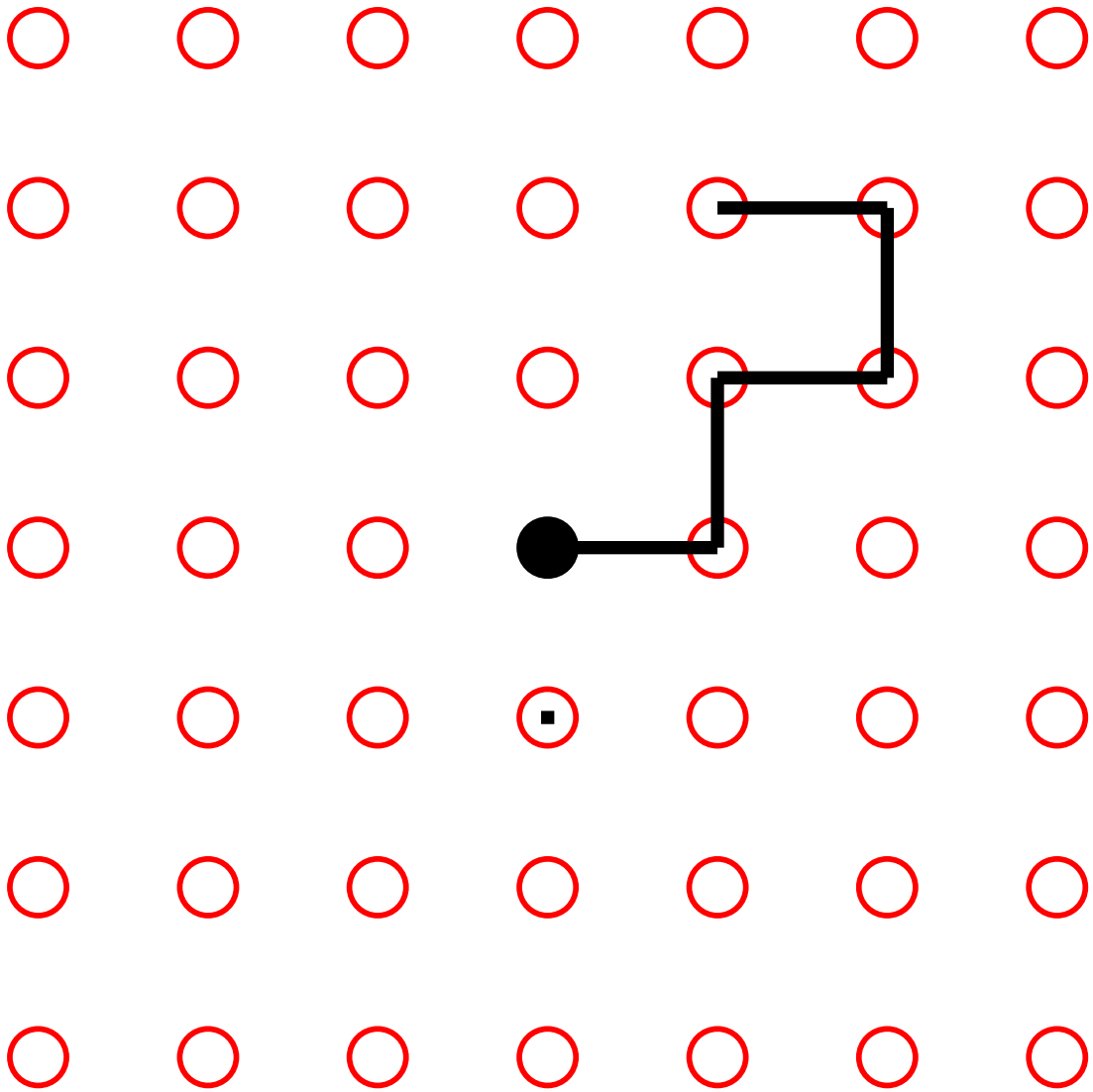
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 4 in 2D



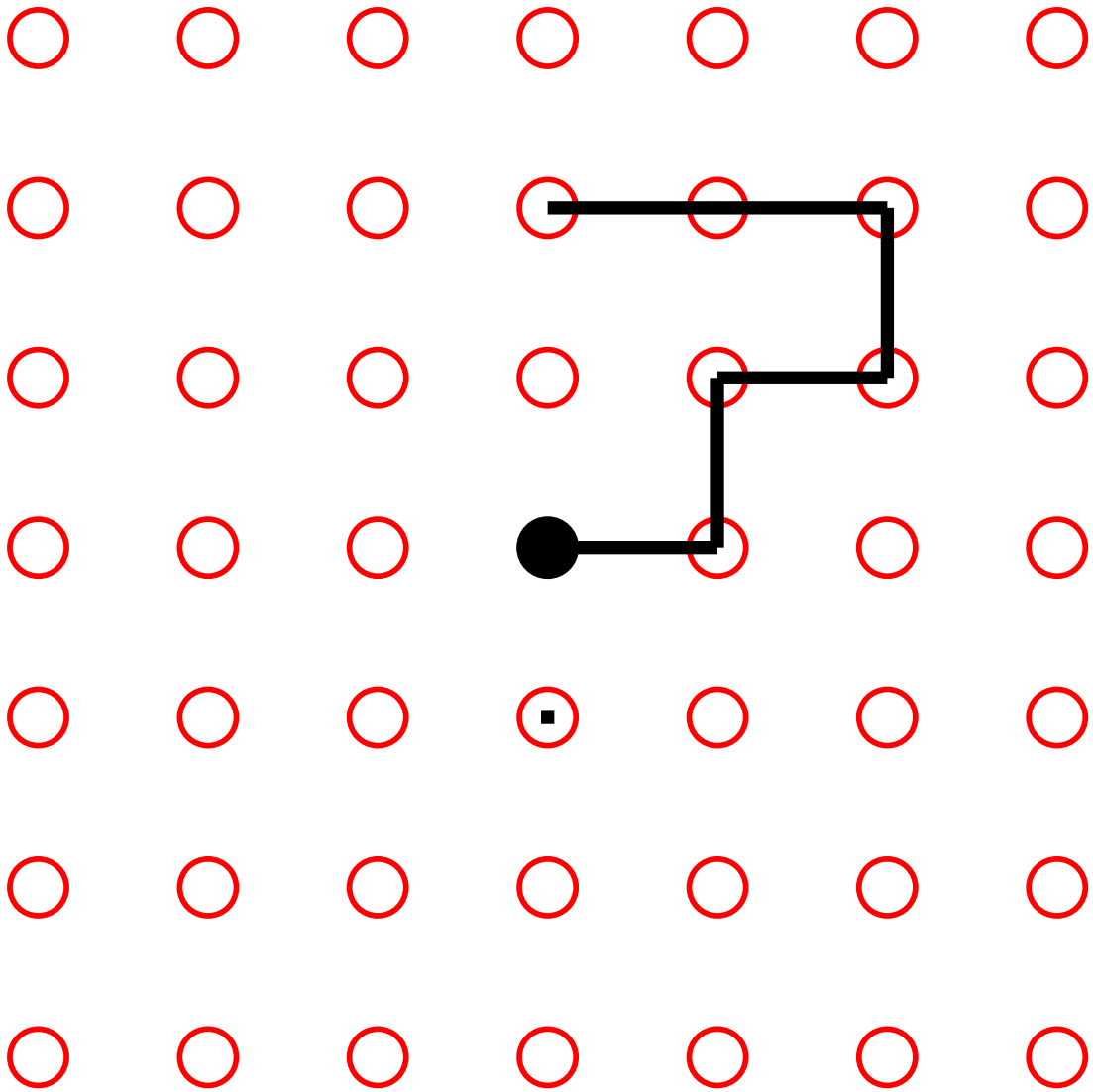
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 5 in 2D



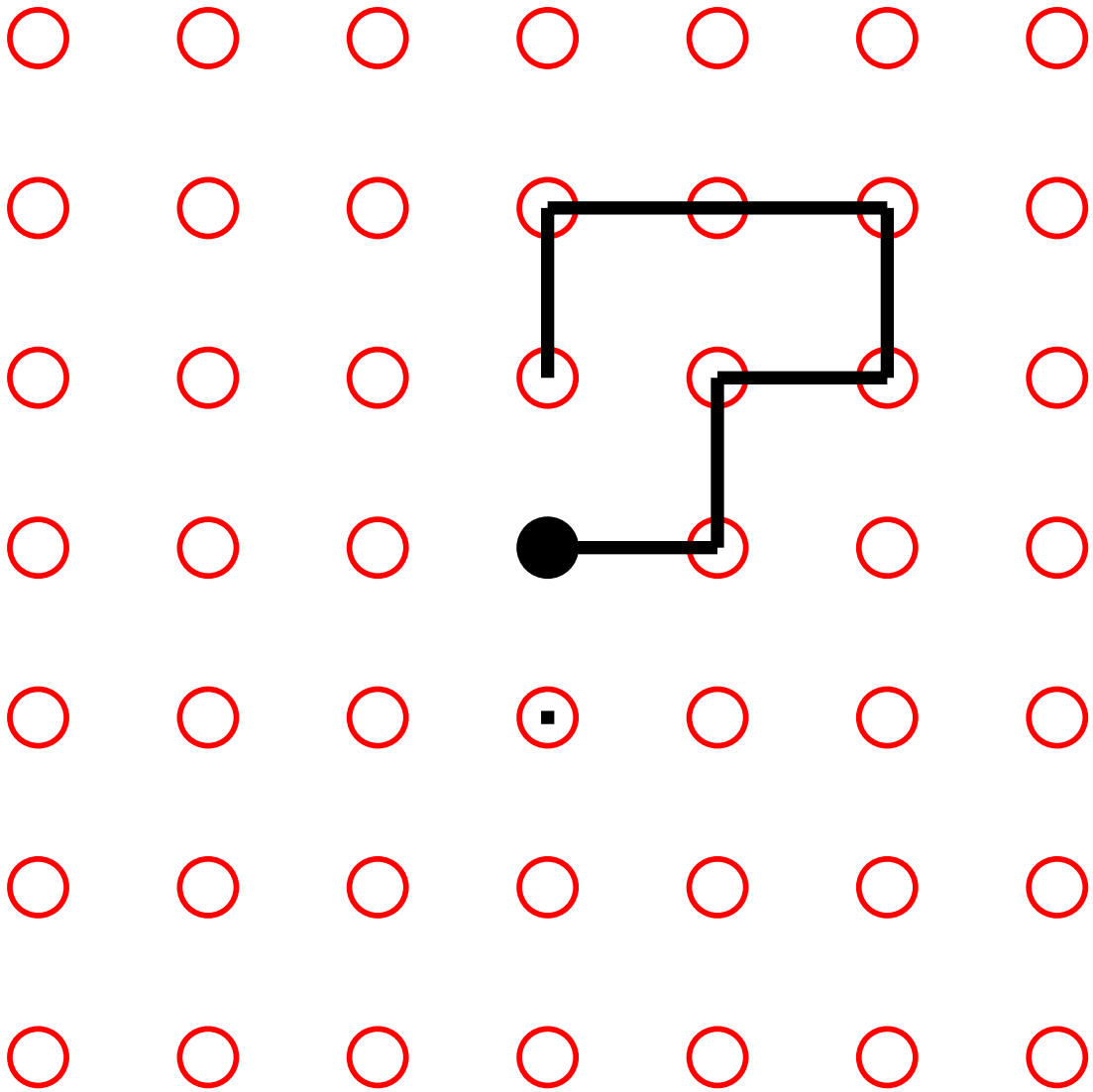
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 6 in 2D



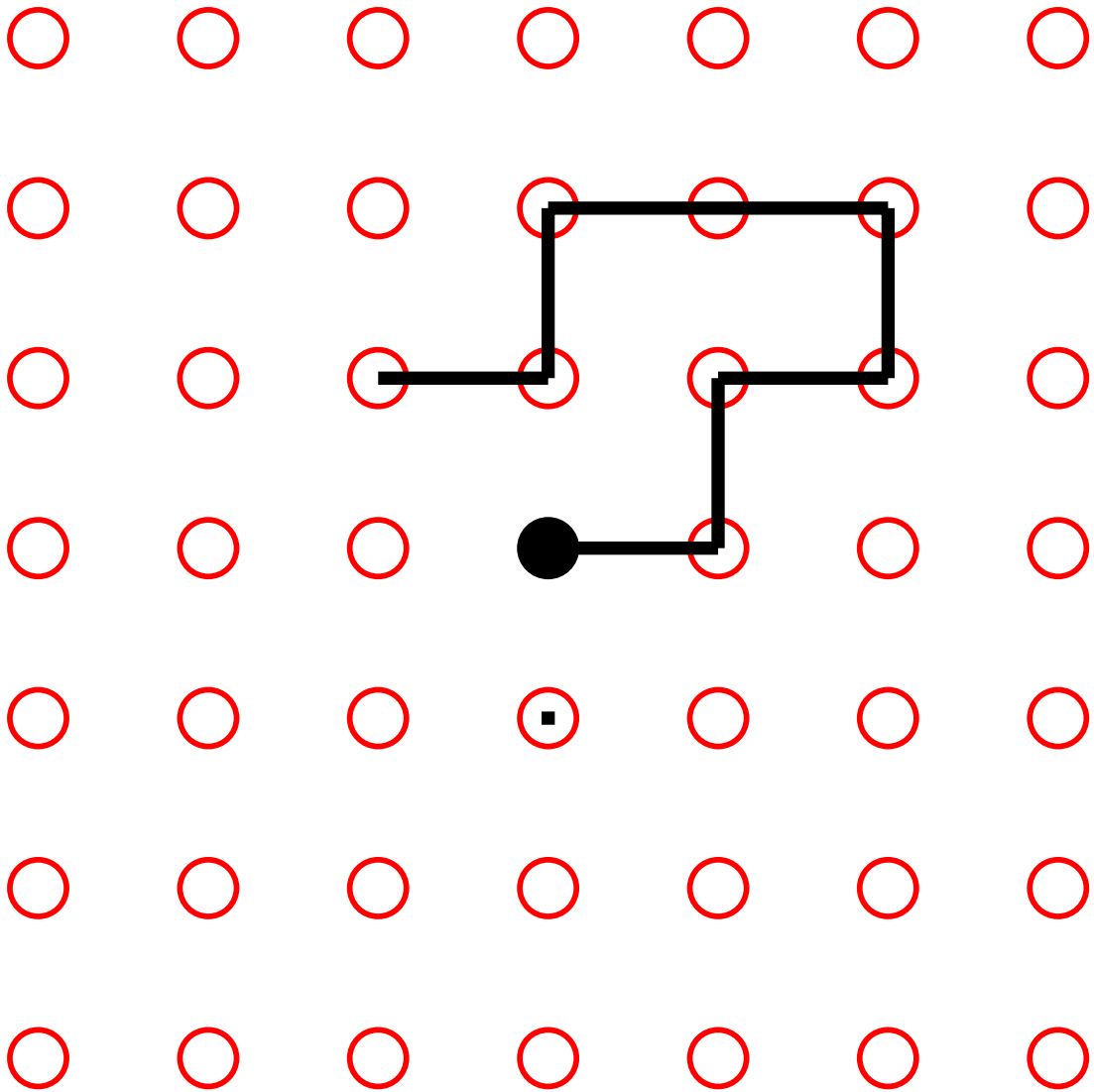
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 7 in 2D



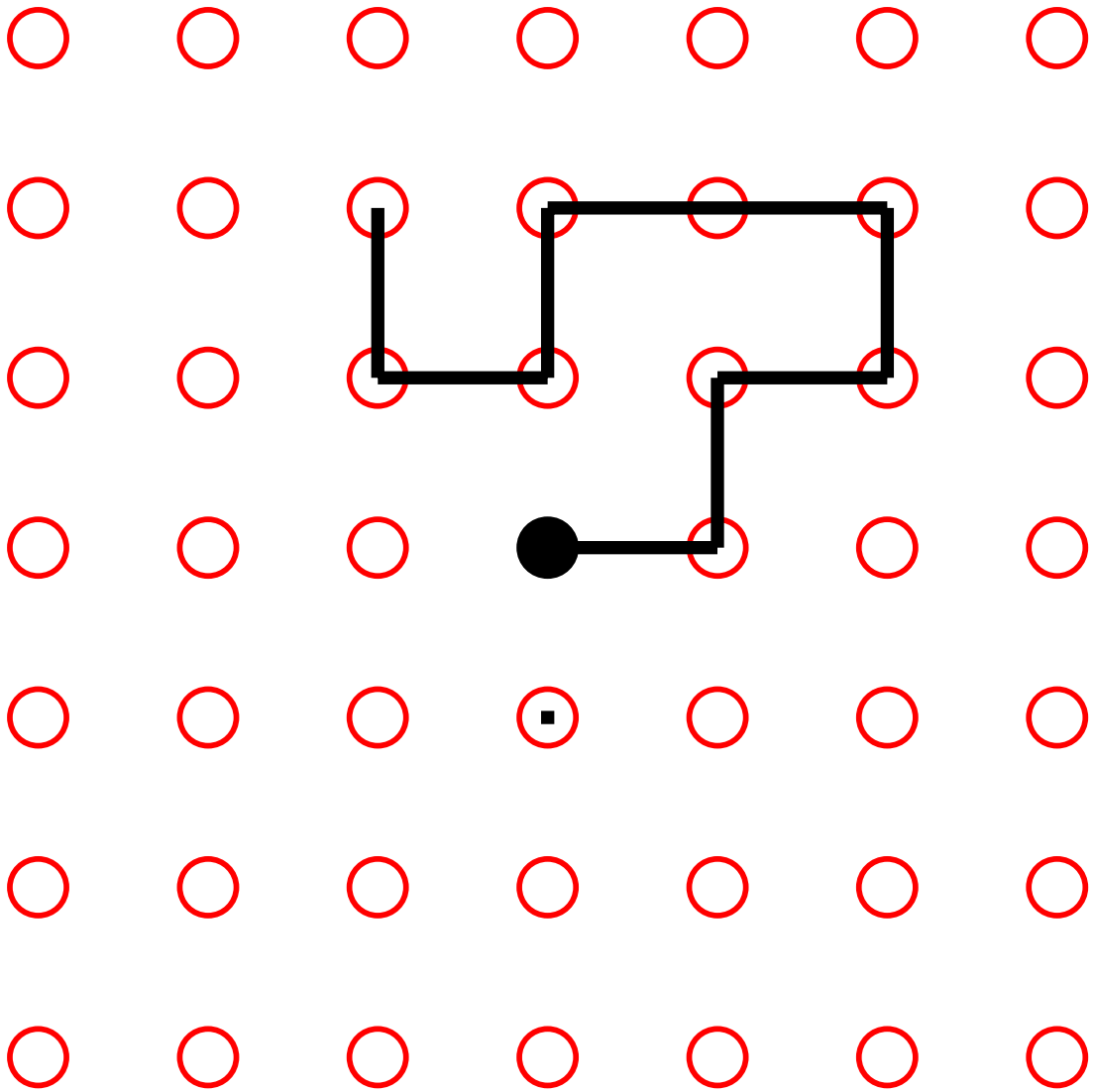
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 8 in 2D



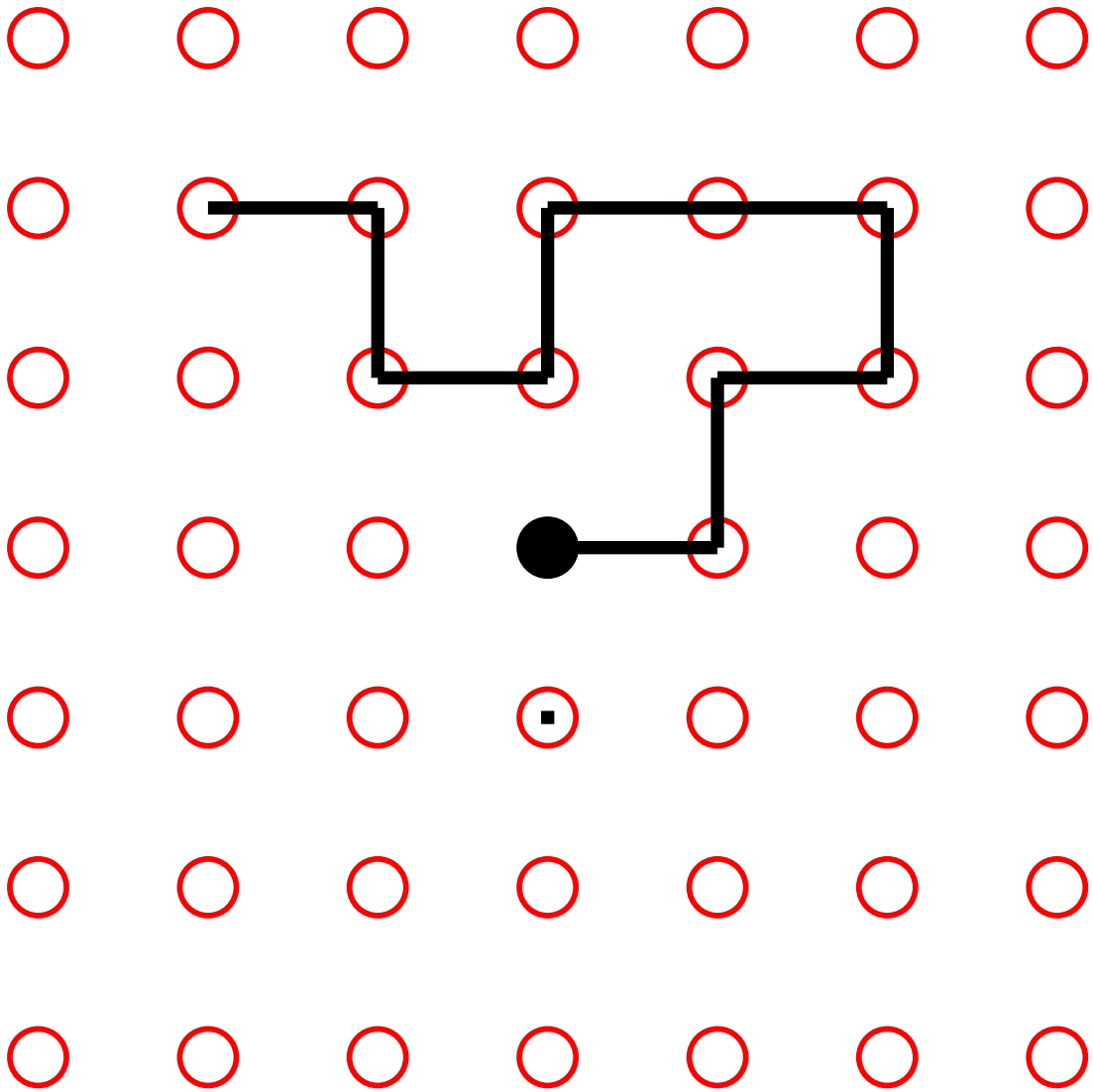
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 9 in 2D



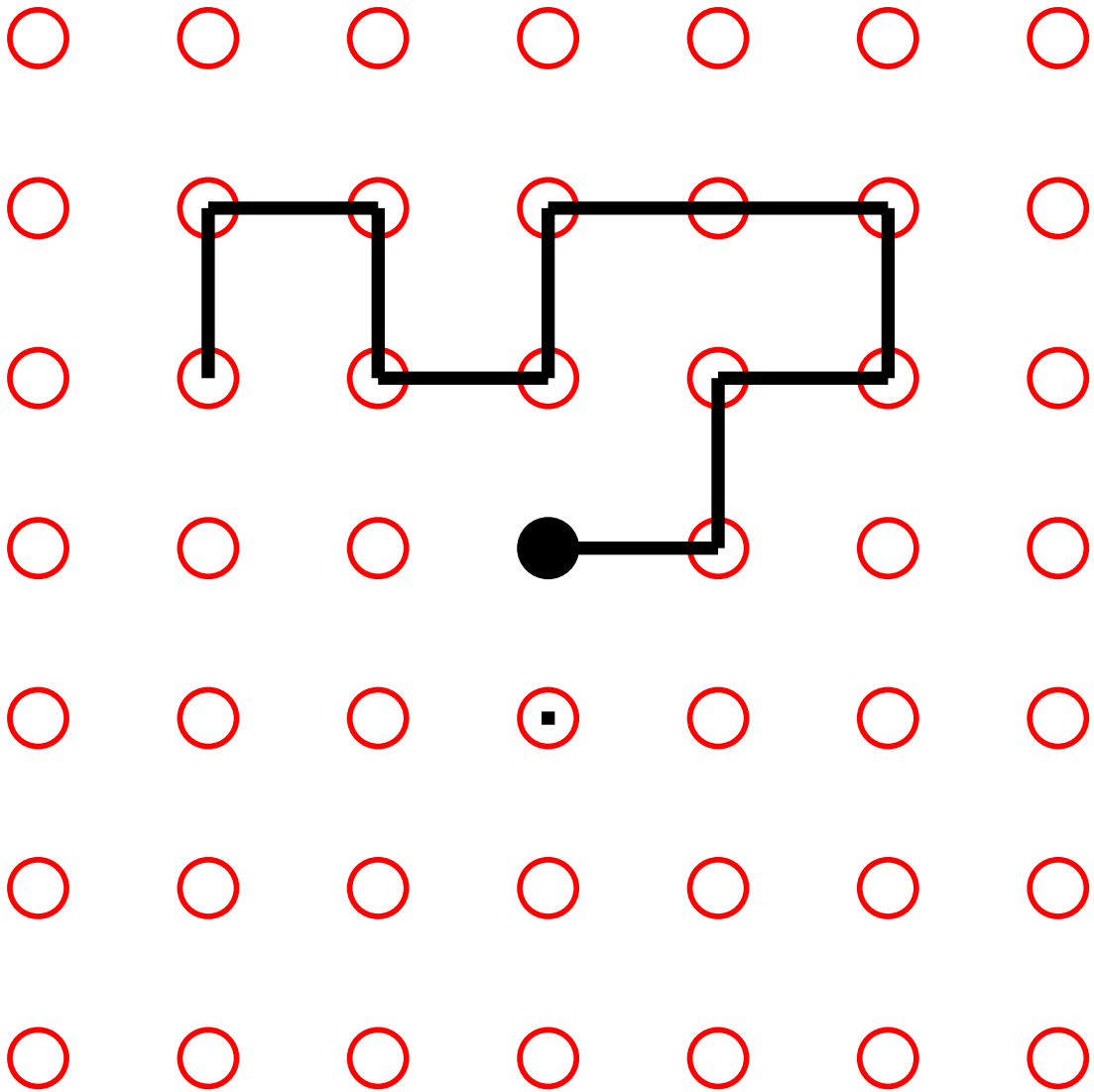
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 10 in 2D



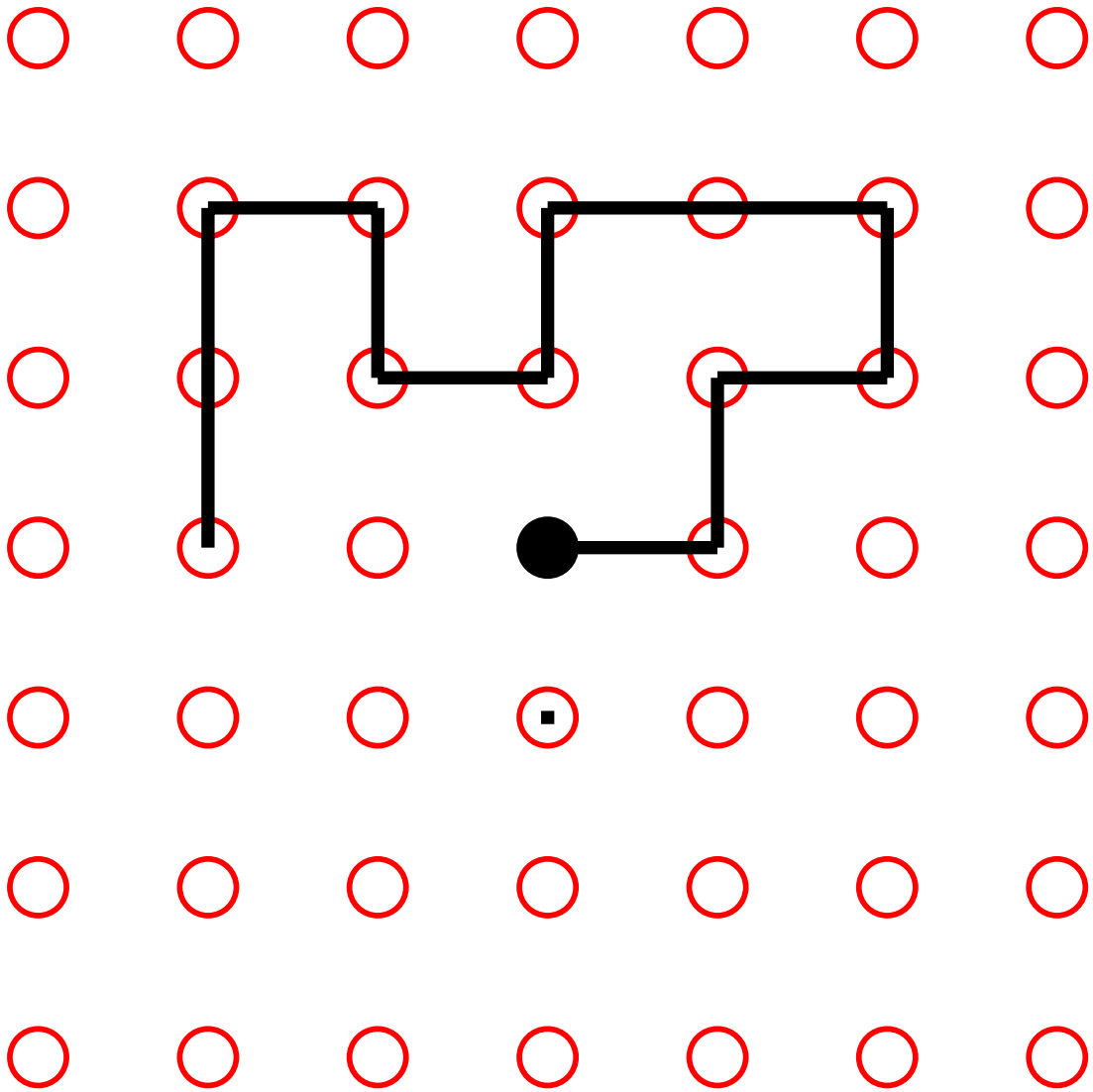
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 11 in 2D



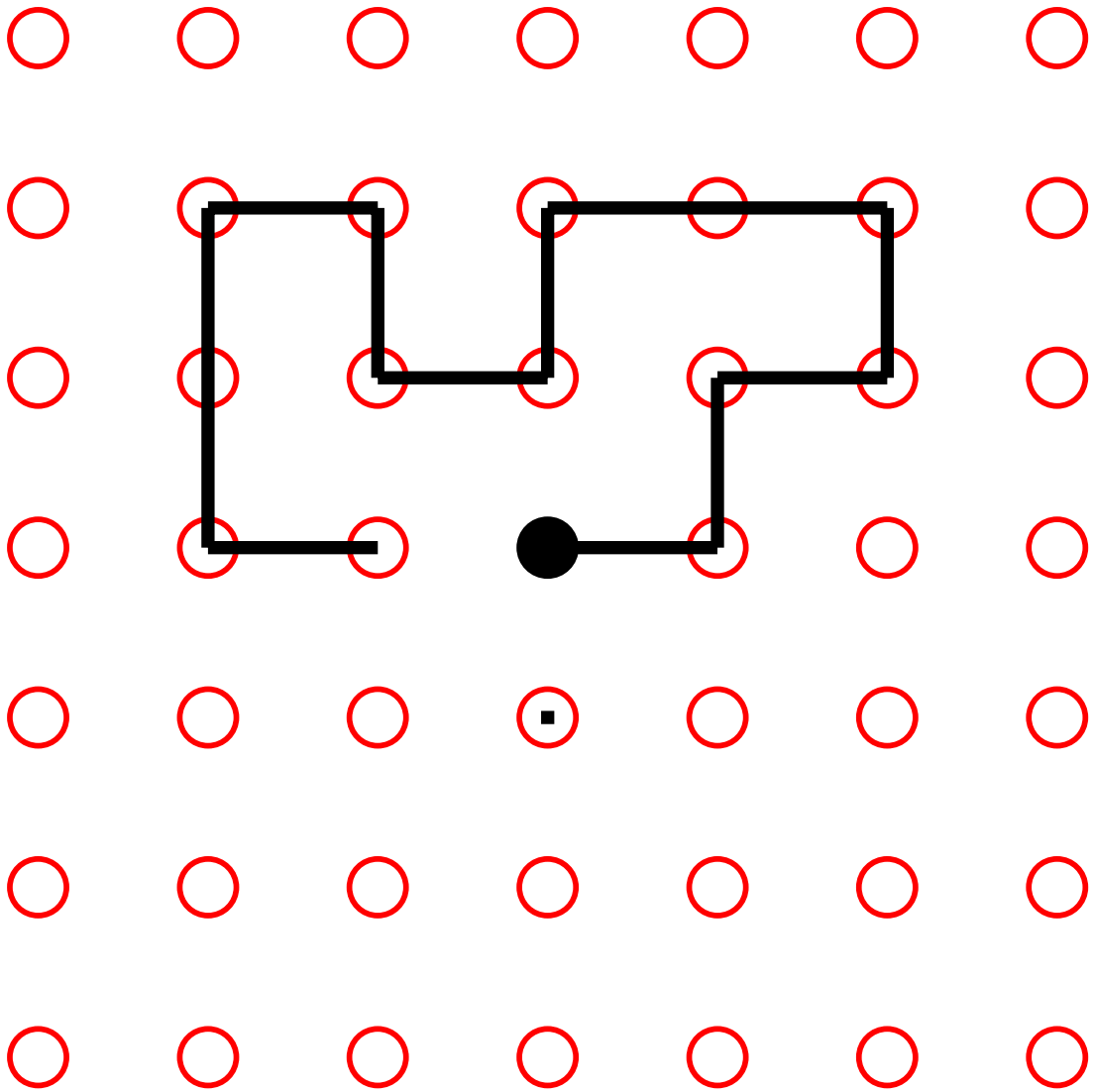
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 12 in 2D



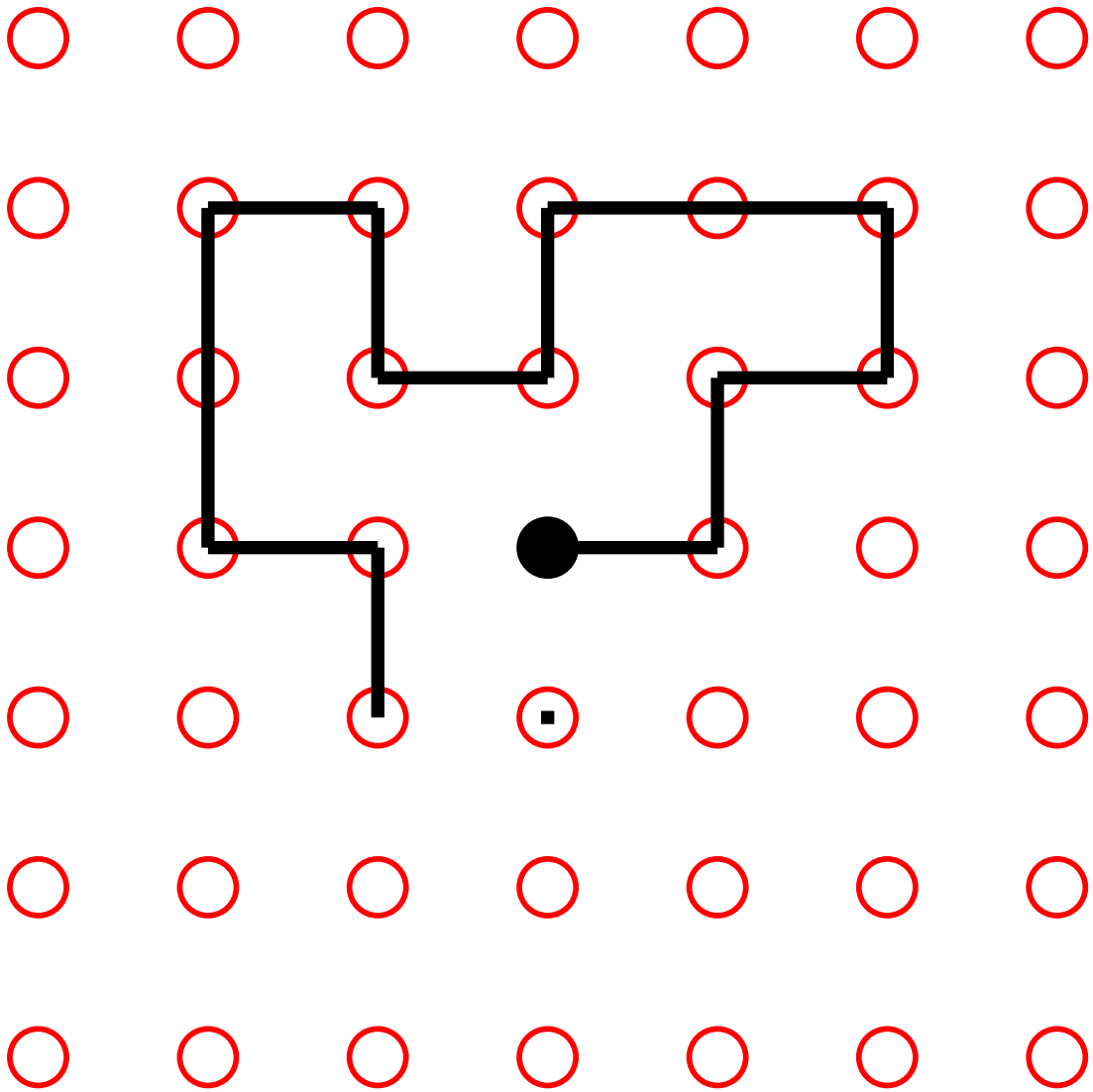
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 13 in 2D



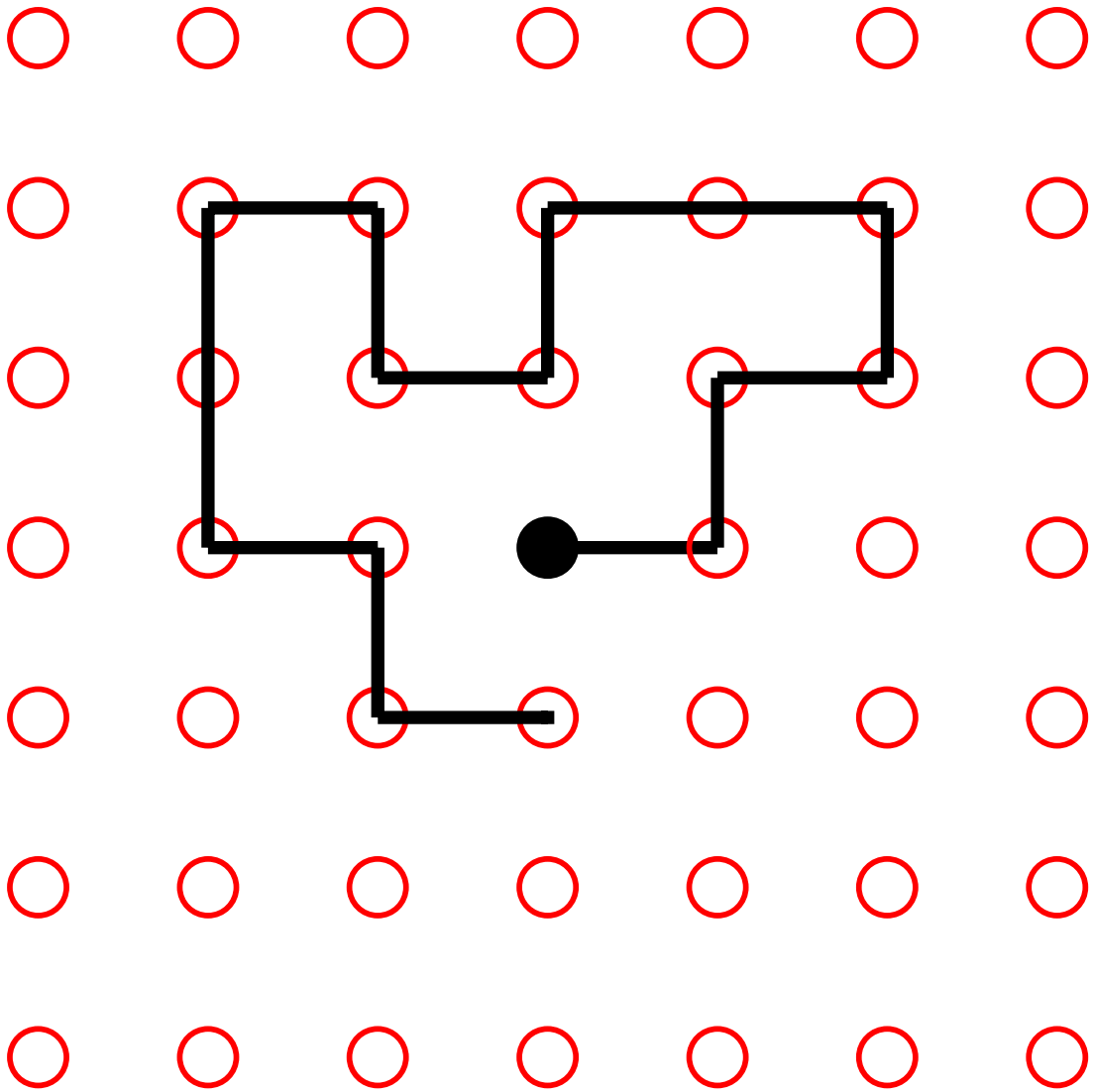
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 14 in 2D



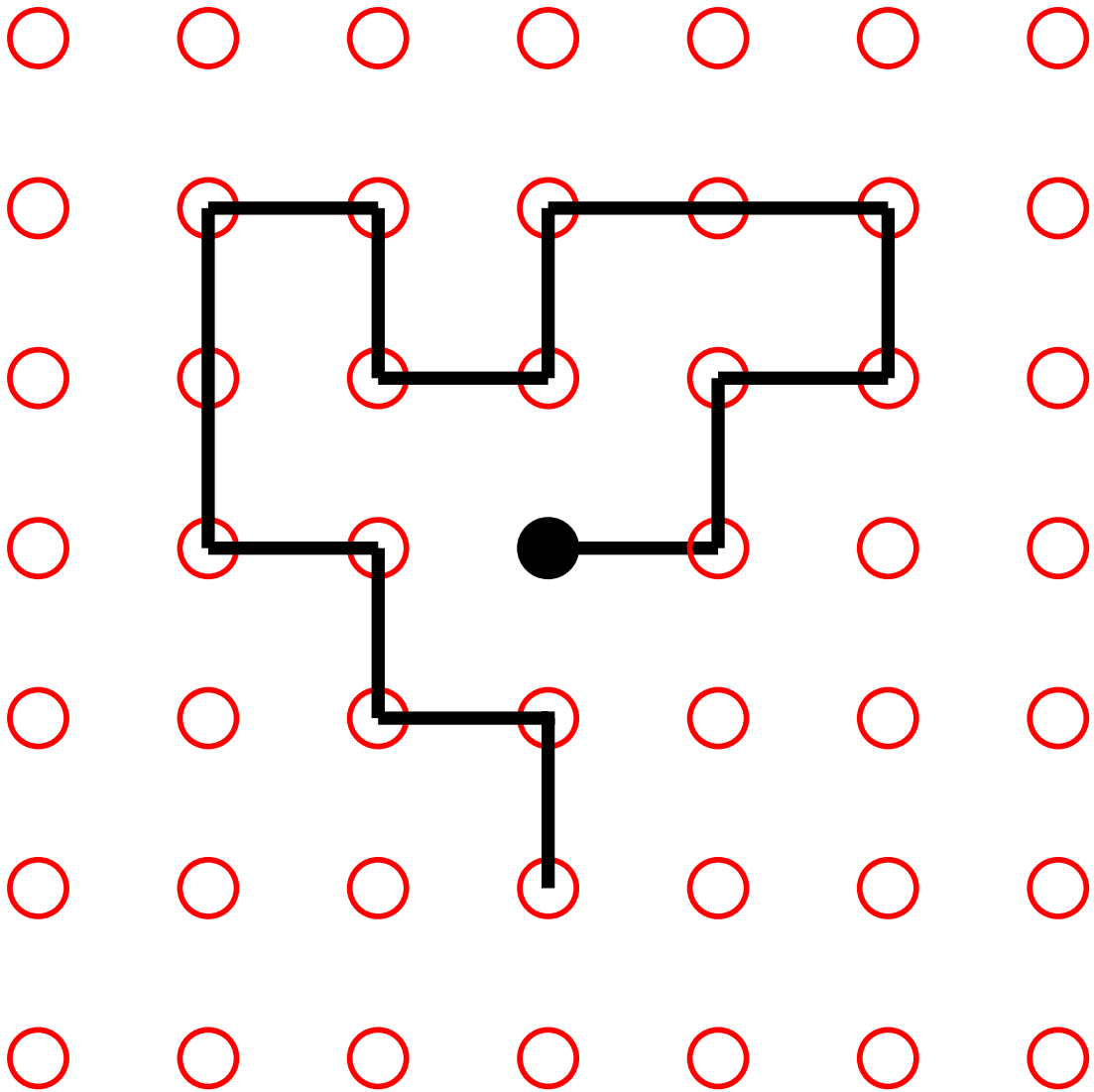
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 15 in 2D



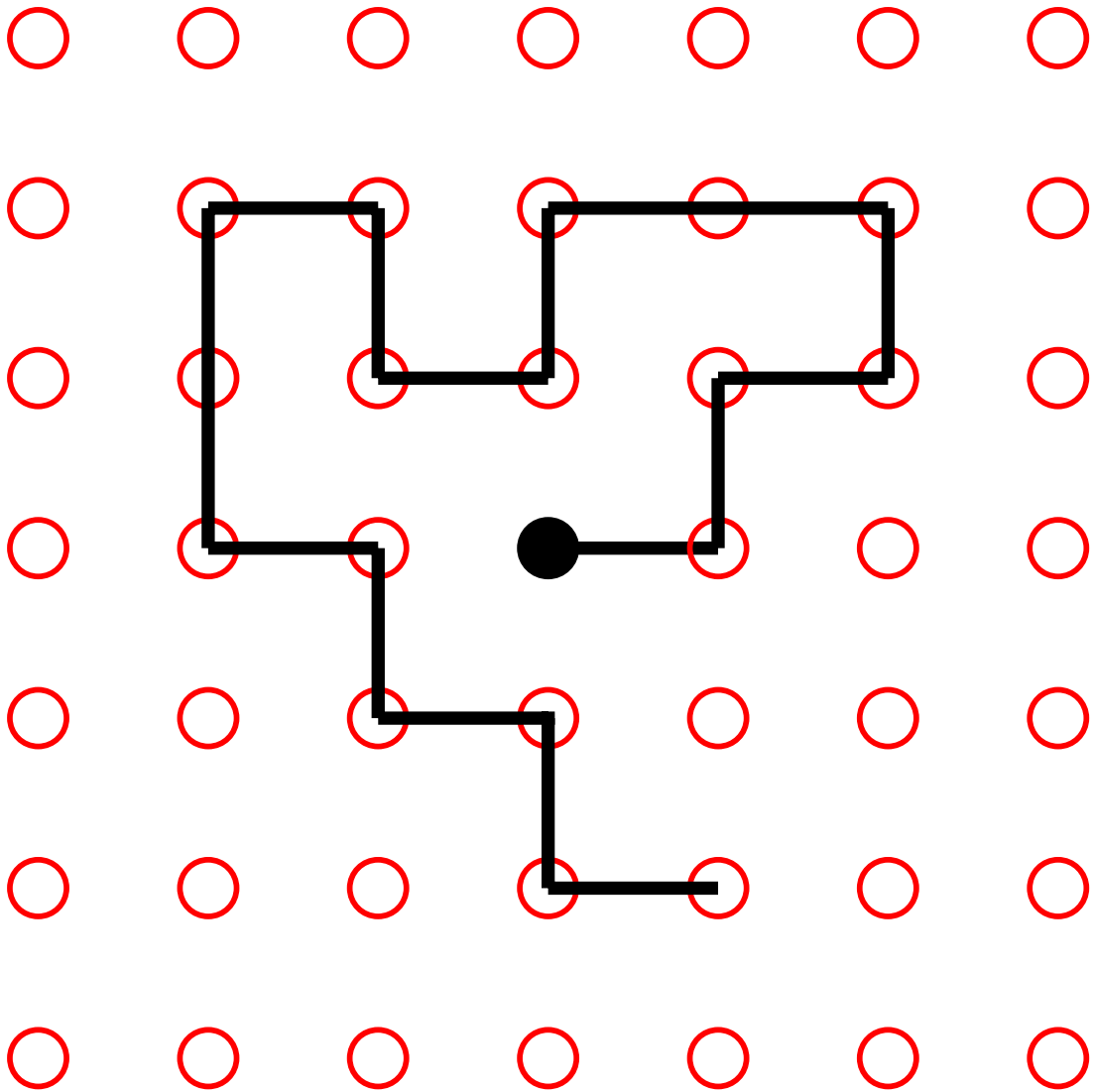
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 16 in 2D



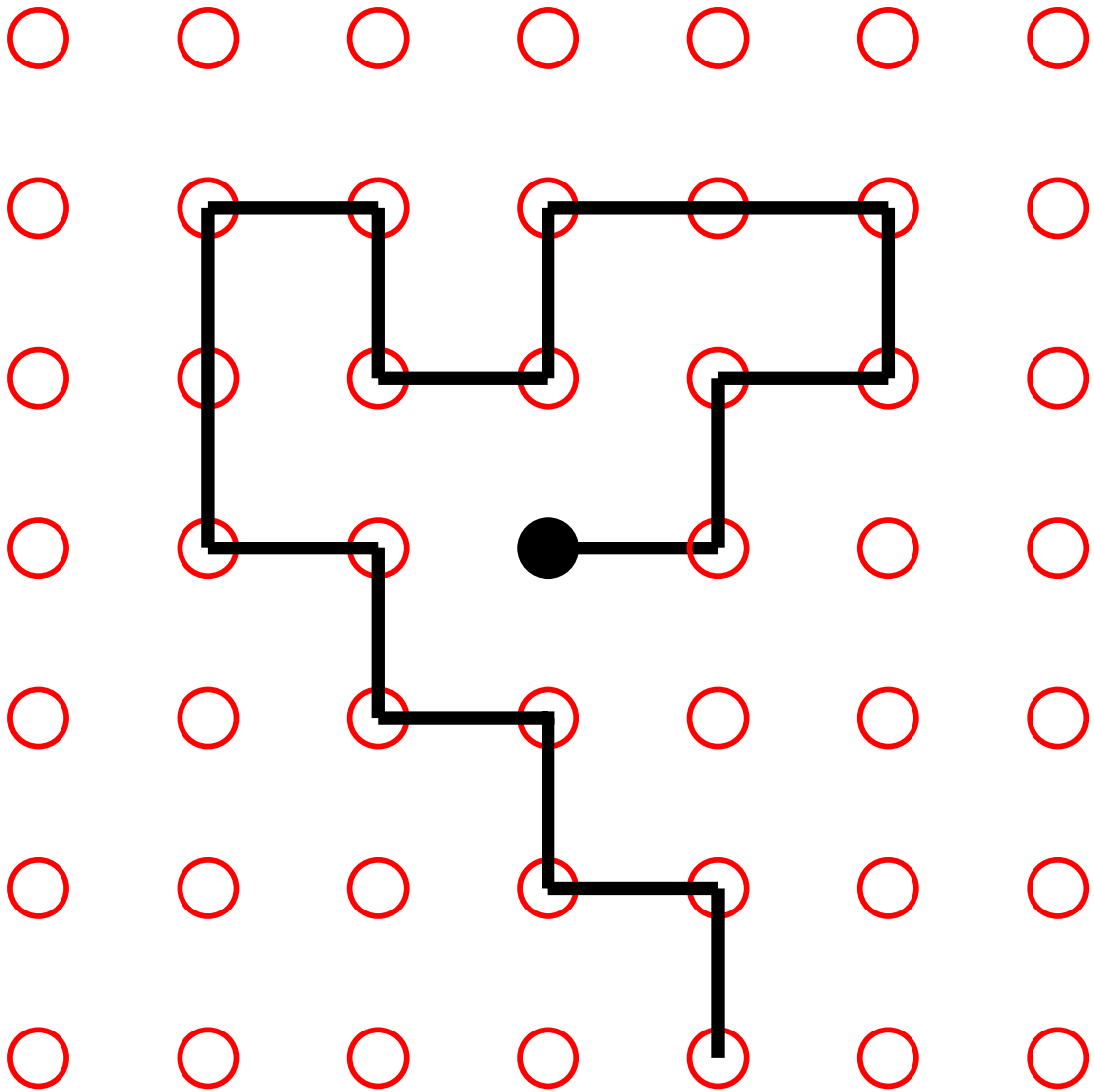
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 17 in 2D



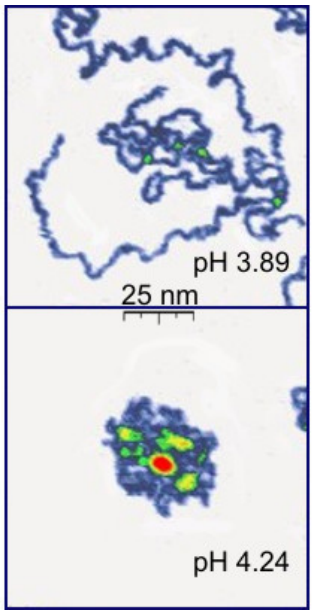
Universiteit Utrecht

Een zelfmijdende wandeling van lengte 18 in 2D



Universiteit Utrecht

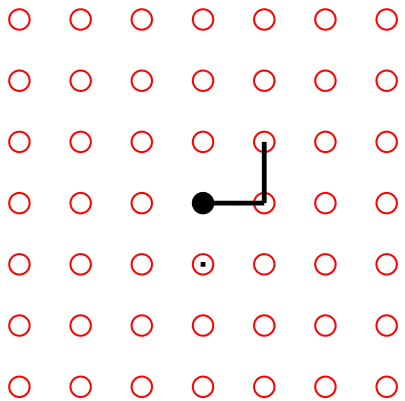
Waarom zijn zelfmijdende wandelingen nuttig?



Poly(2-vinylpyridine) geobserveerd m.b.v. Atomic Force Microscope. Bron: Roiter en Minko (2007).

- Wandeling modelleert een polymeer, een lang molecuul, gebaseerd op een koolstofketen C–C–C–C$\cdots$ C.
- DNA is een polymeer. Wij bestaan uit DNA.
- Zelfmijdend omdat je niet twee koolstofatomen op dezelfde plaats kunt zetten.

Hoeveel zelfmijdende wandelingen zijn er?



- In 2D lengte $N = 2$: $Z_2 = 12$ wandelingen.
- Vraag: in 2D, wat is Z_3, Z_4?
- Vraag: in 3D, wat is Z_1, Z_2, Z_3, Z_4?
- Grenzen in 2D:

$$2^N \leq Z_N \leq 4 \cdot 3^{N-1}$$

Universiteit Utrecht

Recursie: de torens van Hanoi

- De torens van Hanoi bevatten een aantal schijven die van links naar rechts verplaatst moeten worden.
- Je mag een grotere schijf niet op een kleinere plaatsen.
- Oorsprong: legende over monniken in een klooster in Hanoi (Vietnam) die 64 schijven verplaatsen.
- Als ze klaar zijn is het einde der wereld gekomen.

Universiteit Utrecht

Snapshots oplossing 8 schijven: beginsituatie

Universiteit Utrecht

Tussensituatie 1

7 schijven zijn verplaatst van links naar midden

Universiteit Utrecht

Tussensituatie 2

Grootste schijf is verplaatst naar eindpositie

Universiteit Utrecht

Eindsituatie

7 schijven verplaatst naar eindpositie

Universiteit Utrecht

Recursieve algoritmen

- Een recursief algoritme roept zichzelf aan.
- Dit is handig om een probleem op te splitsen in kleinere deelproblemen.
- Korte programmateksten, maar wel goed nadenken.

Recursief algoritme: torens van Hanoi

move(n, i, j) /* verplaatst n schijven van paal i naar paal j */

$t \neq i, j$ is de andere paal, de tussenpaal
als $n = 1$ dan
    print "verplaats schijf van paal i naar paal j"
anders
    move$(n-1, i, t)$;
    move$(1, i, j)$;
    move$(n-1, t, j)$;

Is het einde der wereld nabij?

- Rekentijd $T_n = 2T_{n-1} + 1$ voor $n > 1$. $T_1 = 1$.
- Afleiden expliciete formule:

$$\begin{aligned} T_n &= 2T_{n-1} + 1 \\ &= 2(2T_{n-2} + 1) + 1 \\ &= 4T_{n-2} + 3 \\ &= 4(2T_{n-3} + 1) + 3 \\ &= 8T_{n-3} + 7 \\ &= \cdots \\ &= 2^{n-1}T_1 + 2^{n-1} - 1 \\ &= 2^{n-1} + 2^{n-1} - 1 \\ &= 2^n - 1 \end{aligned}$$

- Netjes opschrijven: bewijs met volledige inductie.

Is het einde der wereld nabij?

- Voor $n = 64$, moeten de monniken $2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$ (18 triljoen) verplaatsingen uitvoeren.
- 1 verplaatsing per seconde: 585 miljard jaar
- We hebben nog de tijd!

Universiteit Utrecht

Recursief 2D SAW in programmeertaal C

Start de recursie:

```
int main (...) {
    ...

    /* Initialiseer rooster en wandeling */
    for (i=-N; i<=N; i++)
        for (j=-N; j<=N; j++)
            visited[i,j] = FALSE;
    x[0] = y[0] = 0;

    /* Creeer alle wandelingen */
    go(0,N);
}
```


Recursieve functie go

Invariant: de wandeling $(x[0], y[0]), \ldots, (x[i], y[i])$ is zelfmijdend t/m $(x[i-1], y[i-1])$.

```
void go(int i, int N) {

    /* De invariant geldt hier */

    if (visited[x[i],y[i]])
        return; /* zet een stapje terug */

    /* Als i=N, dan is de wandeling volledig */
    if (i==N) {
        printf(" Hoera! \n "); return;
    }
    ...
```


Universiteit Utrecht

Recursieve functie go, vervolg

```
    /* Zet de volgende stap */
    visited[x[i],y[i]] = TRUE;

    x[i+1]=x[i]+1;   y[i+1]=y[i]   ;   go(i+1,N);
    x[i+1]=x[i]-1;   y[i+1]=y[i]   ;   go(i+1,N);
    x[i+1]=x[i]    ;   y[i+1]=y[i]+1;   go(i+1,N);
    x[i+1]=x[i]    ;   y[i+1]=y[i]-1;   go(i+1,N);

    visited[x[i],y[i]] = FALSE ;

    return ;
}
```


Universiteit Utrecht

Opgave: Z_{M+N} ('buzz' opgave)

- Zoek een afschatting van Z_{M+N}, het aantal zelfmijdende wandelingen van lengte $M + N$, uitgedrukt in Z_M en Z_N.
- Wat betekent dit voor de rij Z_1, $(Z_2)^{1/2}$, $(Z_4)^{1/4}$, ...?

Antwoord opgave: Z_{M+N}

- Er geldt
$$Z_{M+N} \leq Z_M \cdot Z_N.$$
- Immers, een zelfmijdende wandeling van lengte $M+N$ kun je opknippen in een wandeling van lengte M en een van lengte N.
- Er zijn in totaal hoogstens $Z_M \cdot Z_N$ verschillende mogelijkheden.
- Neem $M = N$, dan is $Z_{2N} \leq (Z_N)^2$.
- Dus $Z_N \geq (Z_{2N})^{1/2}$ voor alle N dus

$$Z_1 \geq (Z_2)^{1/2} \geq (Z_4)^{1/4} \geq (Z_8)^{1/8} \geq \cdots$$

Aantal wandelingen in 2D

- $Z_N \geq (Z_{2N})^{1/2}$ voor alle N dus

$$Z_1 \geq (Z_2)^{1/2} \geq (Z_4)^{1/4} \geq (Z_8)^{1/8} \geq \cdots$$

N	Z_N	$(Z_N)^{1/N}$
1	4	4.00
2	12	3.46
4	100	3.16
8	5916	2.96
16	17245332	2.83
32	119034997913020	2.75
64	4549252727304405545665901684	2.70
79	1019471029355746619378790007192367 6	2.695

- $N = 79$ is huidig wereldrecord (Iwan Jensen, 2013).
- Dalend rijtje getallen, van beneden begrensd, dus limiet:

$$\lim_{N\to\infty} (Z_N)^{1/N} = \mu \approx 2.64, \qquad \text{met } Z_N \sim \mu^N$$

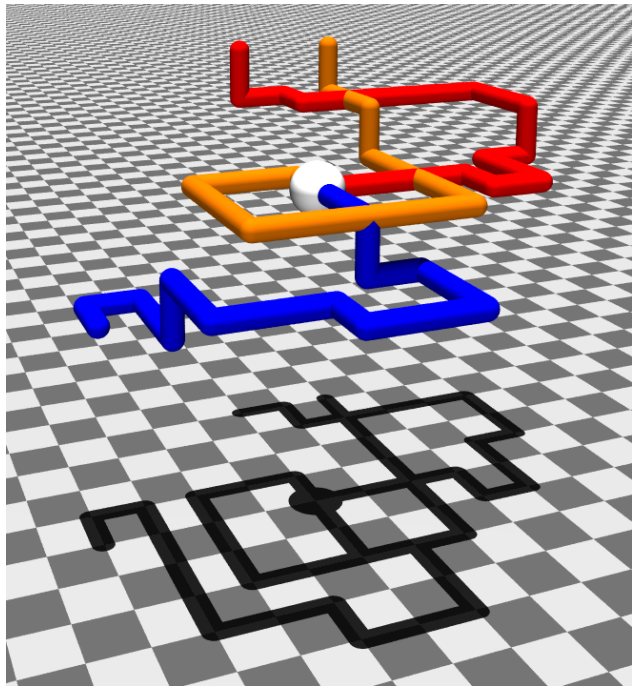
Honingraatrooster in 2D

Bron: slate.com

- $\mu = \sqrt{2+\sqrt{2}} \approx 1.848$ is theoretisch bewezen (Duminil-Copin & Smirnov 2010).

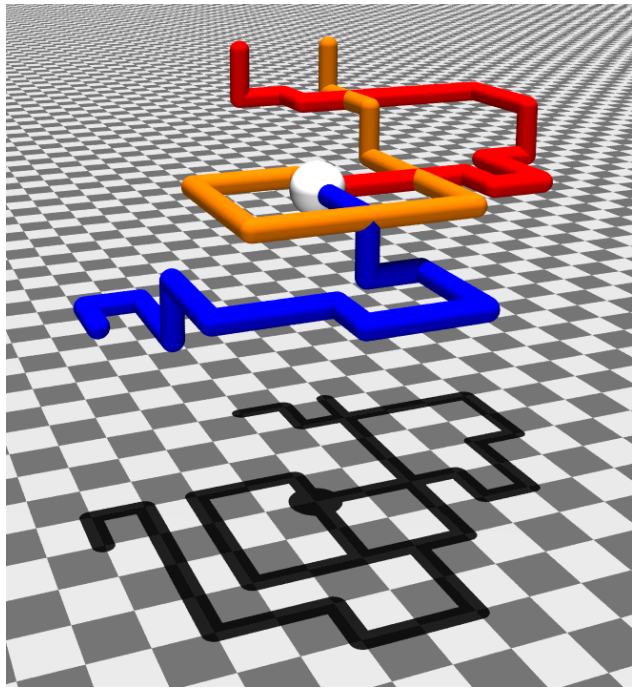
Universiteit Utrecht

Drie zelfmijdende wandelingen van lengte 18 in 3D



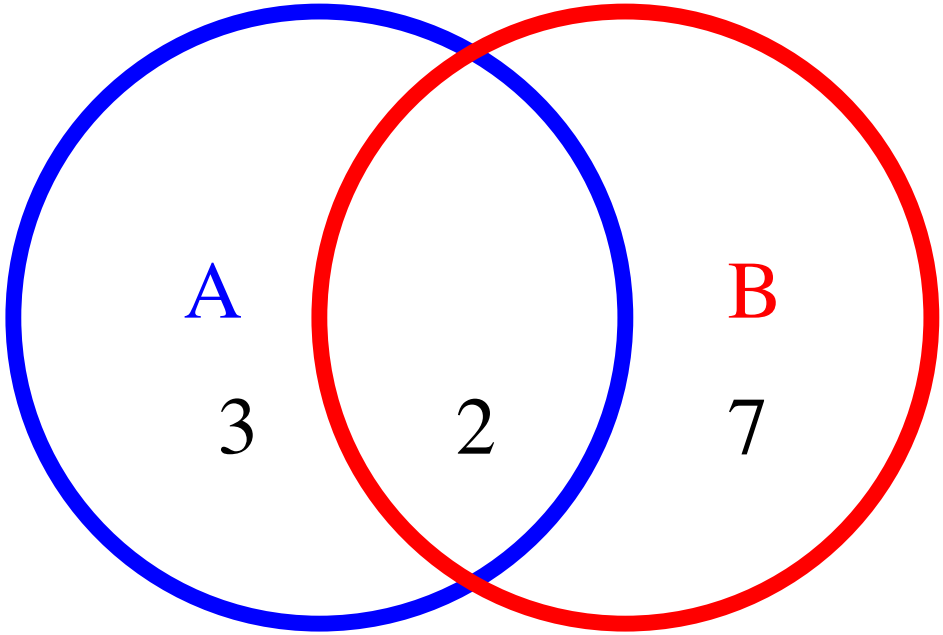
- Zelfmijdende wandelingen van lengte 18:
  rood, oranje, blauw.
- Vraag: hoeveel combinaties van twee verschillende zelfmijdende wandelingen zijn er die je tot een wandeling van lengte 36 kunt aaneenknopen?
- Vraag: hetzelfde maar nu met zelfmijdend resultaat.

Andere telmethode: kijk naar snijpunten



- Snijpunt $a = (2, 0, 0)$: rood/oranje
- Snijpunt $b = (2, 3, 1)$: rood/oranje
- Snijpunt $c = (0, -2, 0)$: blauw/oranje
- Er zijn 3 paren $\{v, w\}$ met $v \neq w$.
- Voor elk snijpunt, verwijder het bijbehorende paar.
- Corrigeer: {rood, oranje} was dubbel verwijderd, dus totaal 3-3+1 = 1 combinatie blijft over.

Principe van inclusie-exclusie: 2 verzamelingen



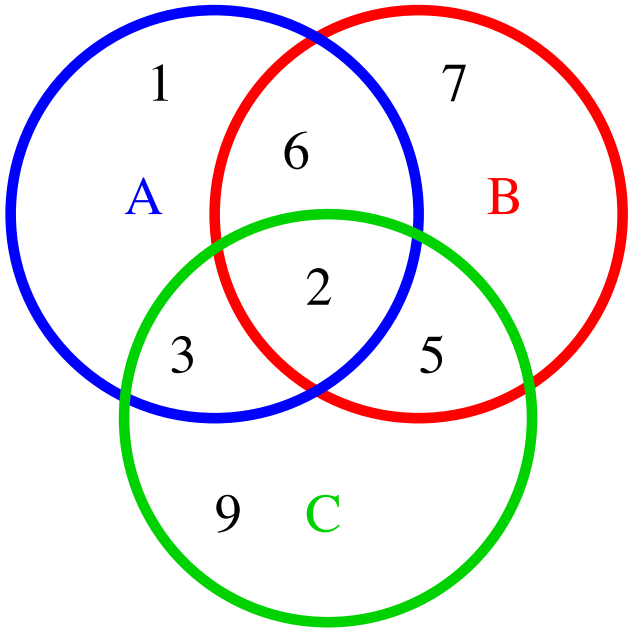
- Aantal elementen van A is $|A| = 5$.
- $|B| = 9$. Aantal elementen van de vereniging is

$$|A \cup B| = |A| + |B| - |A \cap B|$$

- Hier: $|A \cup B| = 5 + 9 - 2 = 12$.
- Andere telwijze: $3+2+7 = 12$.

Universiteit Utrecht

Principe van inclusie-exclusie: 3 verzamelingen

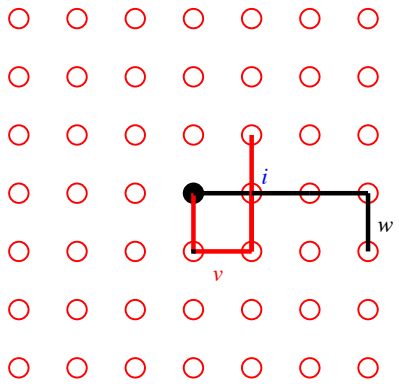


- Aantal elementen van de vereniging is

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$$

- Hier: $|A \cup B \cup C| = 12 + 20 + 19 - 8 - 7 - 5 + 2 = 33$.
- Andere telwijze: $1+6+2+3+7+5+9 = 33$.

Tellen van paren wandelingen



- Pas het inclusie-exclusie principe uit de combinatoriek toe met A_i de verzameling van paren zelfmijdende wandelingen $\{v, w\}$ van lengte N die beide door roosterpunt i gaan.
- De roosterpunten zijn genummerd van 1 tot n; de oorsprong $i = 0$ doet niet mee.
- De verzameling

$$\bigcup_{i=1}^{n} A_i = A_1 \cup A_2 \cup \cdots \cup A_n$$

bevat alle paren die elkaar snijden.

Universiteit Utrecht

Berekening van Z_{2N}

- Zelfde aantal
  - zelfmijdende wandelingen van lengte $2N$
  - niet-snijdende paren van lengte N
- Er geldt dus:

$$Z_{2N} = Z_N^2 - \left| \bigcup_i A_i \right| .$$

- Dit kunnen we efficiënt uitrekenen door alleen naar wandelingen van lengte N te kijken.

Lengteverdubbelingsformule

- We krijgen

$$Z_{2N} = Z_N^2 + \sum_{S \neq \emptyset} (-1)^{|S|} Z_N^2(S).$$

- $Z_N(S)$ is het aantal zelfmijdende wandelingen van lengte N dat door alle roosterpunten van de deelverzameling S gaat.

Rekencomplexiteit

- Om $Z_N(S)$ te berekenen, maken we alle wandelingen van lengte N.
- Voor elke wandeling creëren we alle 2^N deelverzamelingen van haar N roosterpunten en verhogen de tellers van de deelverzamelingen met 1 in een globale datastructuur.
- De totale rekencomplexiteit is

$$\mathcal{O}(2^N \cdot Z_N) = \mathcal{O}(2^N \mu^N) = \mathcal{O}((2\mu)^N).$$

  Dit is veel minder dan $\mathcal{O}(\mu^{2N}) = \mathcal{O}((\mu^2)^N)$, mits $\mu > 2$.
- 3D kubisch rooster: $\mu = 4.68$, voor $2N = 36$ besparen we een factor $(\mu/2)^{18} \approx 4.4 \times 10^6$.

Benutten van 48-voudige symmetrie van kubus

- 8 spiegelingen, zoals $(x, y, z) \to (-x, y, z)$.
- 6 draaiingen, zoals $(x, y, z) \to (y, z, x)$.
- We hebben dus een symmetriegroep G van 48 operaties.

Nationale supercomputer

- Nationale supercomputer Huygens is genoemd naar Christiaan Huygens (1629–1695) o.a. ontdekker van de ringen van Saturnus.
- Huygens, de machine, stond in Amsterdam op het Science Park Watergraafsmeer, bij het rekencentrum SURFsara.
- Heeft 3456 rekenkernen (cores), 2 cores per processor.
- Elke core heeft een kloksnelheid van 4.7 GHz.

Rekentijd


Cartesius (achter)

- Totale rekensnelheid Huygens 60 Teraflop/s $= 60 \times 10^{12}$ floating-point operations (drijvende-komma berekeningen) per seconde. Totaal elektriciteitsverbruik 552 kW.
- Wij hebben 200 rekenkernen gebruikt, gedurende 10 dagen in 2011, in totaal zo'n 50.000 uur rekentijd.
- De elektriciteitsrekening was 5000 euro.
- Inmiddels is er een nieuwere machine, Cartesius, met 40,928 rekenkernen en 1052 Teraflop/s rekensnelheid

Universiteit Utrecht

Aantal zelfmijdende wandelingen in 3D

N	Z_N	Jaar Auteur
1	6	
2	30	
3	150	
4	726	
5	3 534	
6	16 926	1947 Orr
7	81 390	
8	387 966	
9	1 853 886	1959 Fisher, Sykes
10	8 809 878	
11	41 934 150	
12	198 842 742	
13	943 974 510	
14	4 468 911 678	
15	21 175 146 054	
16	100 121 875 974	
17	473 730 252 102	
18	2 237 723 684 094	

Aantal zelfmijdende wandelingen in 3D

N	Z_N	Jaar	Auteur
19	10 576 033 219 614		
20	49 917 327 838 734	1987	Guttmann
21	235 710 090 502 158	1989	Guttmann
22	1 111 781 983 442 406		
23	5 245 988 215 191 414	1992	MacDonald e.a.
24	24 730 180 885 580 790		
25	116 618 841 700 433 358		
26	549 493 796 867 100 942	2000	MacDonald e.a.
27	2 589 874 864 863 200 574		
28	12 198 184 788 179 866 902		
29	57 466 913 094 951 837 030		
30	270 569 905 525 454 674 614	2007	Clisby, Liang, Slade
31	1 274 191 064 726 416 905 966		
32	5 997 359 460 809 616 886 494		
33	28 233 744 272 563 685 150 118		
34	132 853 629 626 823 234 210 582		
35	625 248 129 452 557 974 777 990		
36	2 941 370 856 334 701 726 560 670	2011	Schram, Barkema, Bisseling

Publicatie in *Journal of Statistical Mechanics*

Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

Exact enumeration of self-avoiding walks

R D Schram[1,2], G T Barkema[1] and R H Bisseling[2]

[1] Institute for Theoretical Physics, Utrecht University, PO Box 80195, 3508 TD Utrecht, The Netherlands
[2] Mathematical Institute, Utrecht University, PO Box 80010, 3508 TA Utrecht, The Netherlands
E-mail: raouldschram@gmail.com, g.t.barkema@uu.nl and R.H.Bisseling@uu.nl

Received 12 April 2011
Accepted 9 June 2011
Published 27 June 2011

Online at stacks.iop.org/JSTAT/2011/P06019
doi:10.1088/1742-5468/2011/06/P06019

Abstract. A prototypical problem on which techniques for exact enumeration are tested and compared is the enumeration of self-avoiding walks. Here, we show an advance in the methodology of enumeration, making the process thousands or millions of times faster. This allowed us to enumerate self-avoiding walks on the simple cubic lattice up to a length of 36 steps.

Keywords: loop models and polymers, critical exponents and amplitudes (theory), exact results

J. Stat. Mech. (2011) P06019

Universiteit Utrecht

Recent werk

- Record voor Body-Centred-Cubic (BCC) rooster:
  $Z_{28} = 129,713,248,317,927,812,262,200$
- Record voor Face-Centred-Cubic (FCC) rooster:
  $Z_{24} = 2,089,765,228,215,904,826,153,292$
  (2 quadriljoen)
- Samen met Nathan Clisby (Univ. Melbourne), 2017, zelfde tijdschrift.
- 6 jaar later vanwege verdachte uitkomsten voor BCC met $N = 27, 28$.
- Bachelorscriptie Sarita de Berg: lengteverdrievoudiging is mogelijk. Gunstig voor $\mu > 8$ zoals $\mu_{\mathrm{FCC}} = 10.03$.
- Wordt vervolgd ...

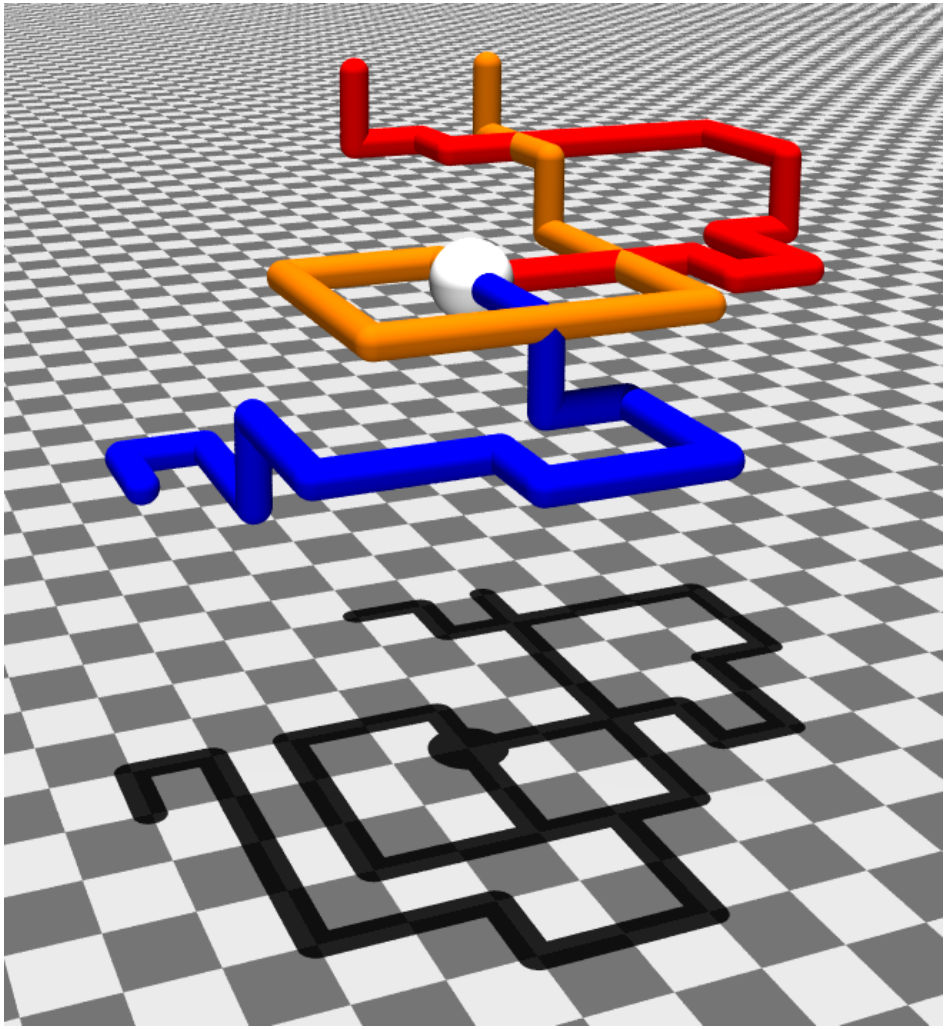
Conclusie

- Onze nieuwe telmethode, lengteverdubbeling, reduceert de rekentijd van het tellen van zelfmijdende wandelingen op het 3D kubisch rooster van 4.68^N naar 3.06^N.
- We hebben het vorige wereldrecord verbeterd van 30 naar 36 stappen, gebruikmakend van een beter schaalbaar algoritme, symmetrie, supercomputing, en een speciale roosternummering.
- Lengteverdubbeling kan ook gebruikt worden voor:
  - 4D hyperkubisch rooster
  - zelfmijdende polygonen
- Softwarepakket Sawdoubler is beschikbaar als open-source code sinds 2012. Sawdoubler 2.0 bevat ook BCC, FCC, en eind-tot-eind afstand P_N.

Zin in meer?

- Programmeren in de Wiskunde (WISB152): recursieve algoritmen, taal Python 3.0
- Groepentheorie (WISB221): 48-voudige symmetrie benutten in 3D
- Numerieke Wiskunde (WISB251): algoritmen voor numeriek oplossen van vergelijkingen
- Discrete Wiskunde (INFOB3DW): inclusie/exclusie, telproblemen en optimaliseringsproblemen
- Wiskundig Modelleren (WISB357): modelleren met differentiaalvergelijkingen, o.a. van materialen
- Inleiding Scientific Computing (WISB356): toepassingen numerieke wiskunde in Google PageRank, MRI-scans
- Parallel Algorithms (WISM459): parallel programmeren

Slot



Dank voor jullie aandacht!

Universiteit Utrecht