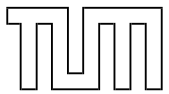
Grundlagen: Algorithmen und Datenstrukturen

Prof. Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2010



Übersicht

Grundlage

- Inhalt der Vorlesung basiert auf dem Buch

  K. Mehlhorn, P. Sanders:
  Algorithms and Data Structures – The Basic Toolbox
  (Springer, 2008)

- Vorlage für die Slides:

  Slides aus SS'08 von Prof. Dr. Christian Scheideler bzw.
  Slides aus SS'09 von Prof. Dr. Helmut Seidl

Vorlesungsdaten

- Modul: IN0007
- Titel: "Grundlagen: Algorithmen und Datenstrukturen"
- 3 SWS Vorlesung + 2 SWS Übung
- ECTS: 6 Credit Points
- Dozent: Prof. Dr. Hanjo Täubig
- Vorlesungszeiten:

Dienstag	14:15 – 15:45 Uhr	(MI Hörsaal 1)
Donnerstag	12:15 – 13:00 Uhr	(MI Hörsaal 1)

- Voraussetzung:

  Modul IN0001: Einführung in die Informatik 1

Übung

- 2 SWS Tutorübungen
- 24 Gruppen (davon 3 englische) an 16 verschiedenen Terminen
- jeweils maximal 15 Teilnehmer
- Anmeldung über Grundstudium-Webseite:
  https://grundstudium.in.tum.de/

  Installation eines Benutzerzertifikats erforderlich(!)
  Falls nicht vorhanden: $\Rightarrow$ InfoPoint
- Übungsleitung:
  Tobias Lieber (lieber@in.tum.de)
- Webseite:
  http://www14.in.tum.de/lehre/2010SS/gad/uebung/

Übungstermine

Montag	16 - 18 Uhr
Dienstag	8 - 10 Uhr
	10 - 12 Uhr
	12 - 14 Uhr
	16 - 18 Uhr
Mittwoch	8 - 10 Uhr
	10 - 12 Uhr
	12 - 14 Uhr
	14 - 16 Uhr
	16 - 18 Uhr
Donnerstag	10 - 12 Uhr
	14 - 16 Uhr
Freitag	8 - 10 Uhr
	10 - 12 Uhr
	12 - 14 Uhr
	14 - 16 Uhr

Zielgruppe

- Bachelor Informatik
- Bachelor Wirtschaftsinformatik
- Bachelor Bioinformatik
- Andere Studiengänge mit Neben-/Zweitfach Informatik
- Masterstudiengang Angewandte Informatik
- Aufbaustudium Informatik

- planmäßig im 2. Fachsemester
  (für Wirtschaftsinformatik im 4. Fachsemester)

Dozent / Kontaktdaten

- Prof. Dr. Hanjo Täubig

  Vertretungsprofessor für Theoretische Informatik
  am Lehrstuhl für Effiziente Algorithmen
  (Lehrstuhlinhaber: Prof. Dr. Ernst W. Mayr)

- eMail: taeubig@in.tum.de

- Web: http://www14.in.tum.de/personen/taeubig/

- Telefon: 089 / 289-17740

- Raum: 03.09.039

- Sprechstunde: Mittwoch 13-14 Uhr
  (oder nach Vereinbarung)

Inhalt

- Grundlagen der Komplexitätsanalyse
- Komplexität der Operationen von Listen, Stacks und Schlangen
- binäre Bäume und Algorithmen (preorder, inorder, postorder)
- binäre Suchbäume und balancierte Suchbäume (AVL, B-Bäume)
- Prioritätswarteschlangen
- Hashing
- Sortieren und sortierbasierte Algorithmen
- Graph-Repräsentation und einfache Graphalgorithmen
- Pattern Matching, Datenkompression

Weitere Literatur

- Cormen, Leiserson, Rivest, Stein:
  Introduction to Algorithms
- Goodrich, Tamassia:
  Algorithm Design: Foundations, Analysis, and Internet Examples
- Heun:
  Grundlegende Algorithmen
  Einführung in den Entwurf und die Analyse effizienter Algorithmen
- Kleinberg, Tardos:
  Algorithm Design
- Schöning:
  Algorithmik
- Sedgewick:
  Algorithmen in Java. Teil 1-4

Übersicht

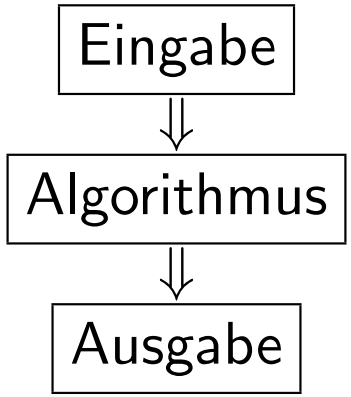
Algorithmus - Herkunft des Begriffs

- Bezeichnung ist abgeleitet vom Namen Muhammed al-Chwarizmi (ca. 783-850)
- sein arabisches Lehrbuch "Über das Rechnen mit indischen Ziffern" begann in lateinischer Übersetzung mit "Dixit Algorismi" ("Algorismi hat gesagt")
- Mittelalter: "algorismus" als Kunst des Rechnens mit den arabischen Ziffern
  (angeblich zusammengesetzt aus dem Namen des Erfinders, dem Philosophen Algus, und dem griechischen Wort rismus/arithmós für Zahl)

Algorithmus - Definition

Definition

Ein Algorithmus ist eine formale Handlungsvorschrift zur Lösung von Instanzen eines Problems in endlich vielen Schritten.



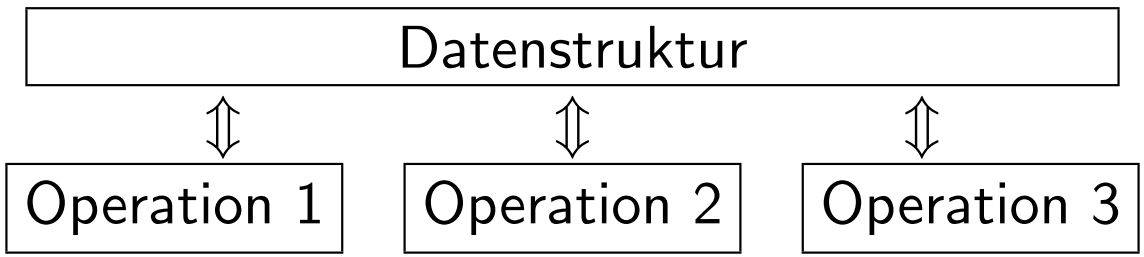
Algorithmus - Beispiele

- Kochrezept
  - Eingabe: Zutaten
  - Algorithmus: Rezept
  - Ausgabe: Essen
- Bauanleitung
  - Eingabe: Einzelteile
  - Algorithmus: Bauanleitung
  - Ausgabe: Schrank
- Weitere Beispiele
  - Weg aus dem Labyrinth
  - Zeichnen eines Kreises

Datenstruktur - Definition

Definition

Eine Datenstruktur ist ein formalisiertes Objekt, das der Speicherung und der Verwaltung von bzw. dem Zugriff auf Daten in geeigneter Weise dient, wobei die Daten dabei zweckdienlich angeordnet, kodiert und miteinander verknüpft werden.

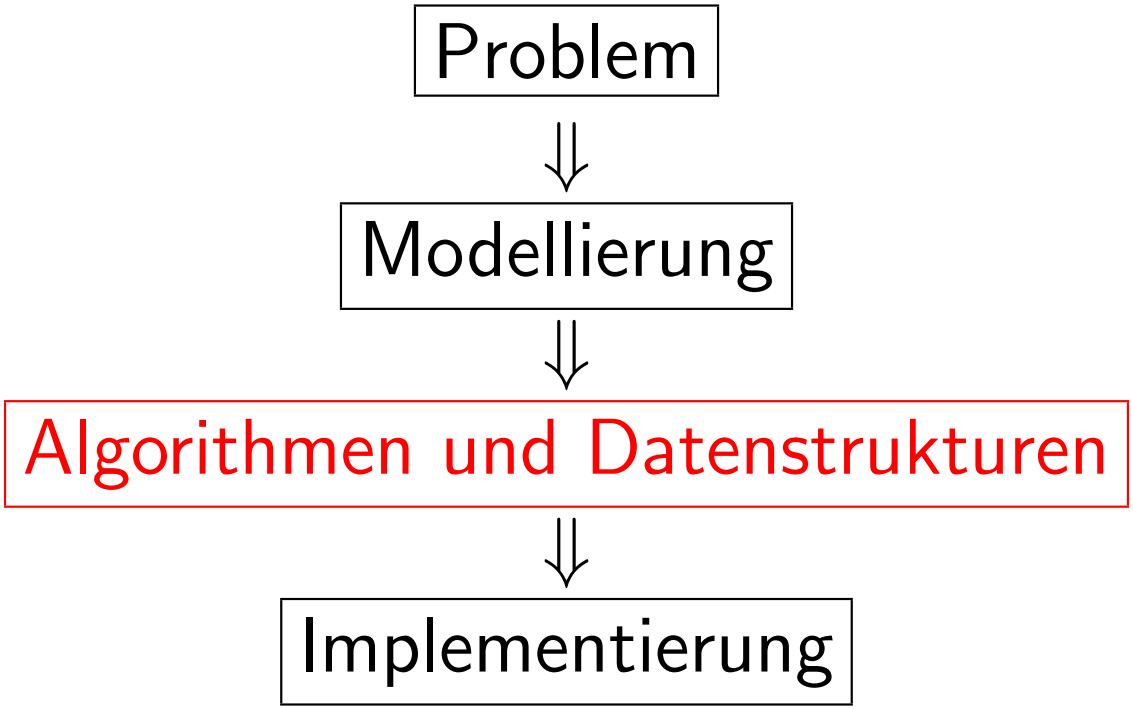


Datenstruktur - Beispiele

- Lexikon
  Operation: Suche(Name)
  (Algorithmus mit Eingabe ⟨Name⟩,
  Ausgabe Information zu ⟨Name⟩)

- Kalender
  Operation: Wochentag(Datum)
  (Algorithmus mit Eingabe ⟨Datum⟩,
  Ausgabe Wochentag zum ⟨Datum⟩)

Softwareentwicklung



Grundsätzliche Probleme

- Korrektheit
- Effizienz
- Komplexität
- Robustheit / Sicherheit

Effizienz

im Sinn von

- Laufzeit
- Speicheraufwand
- Energieverbrauch

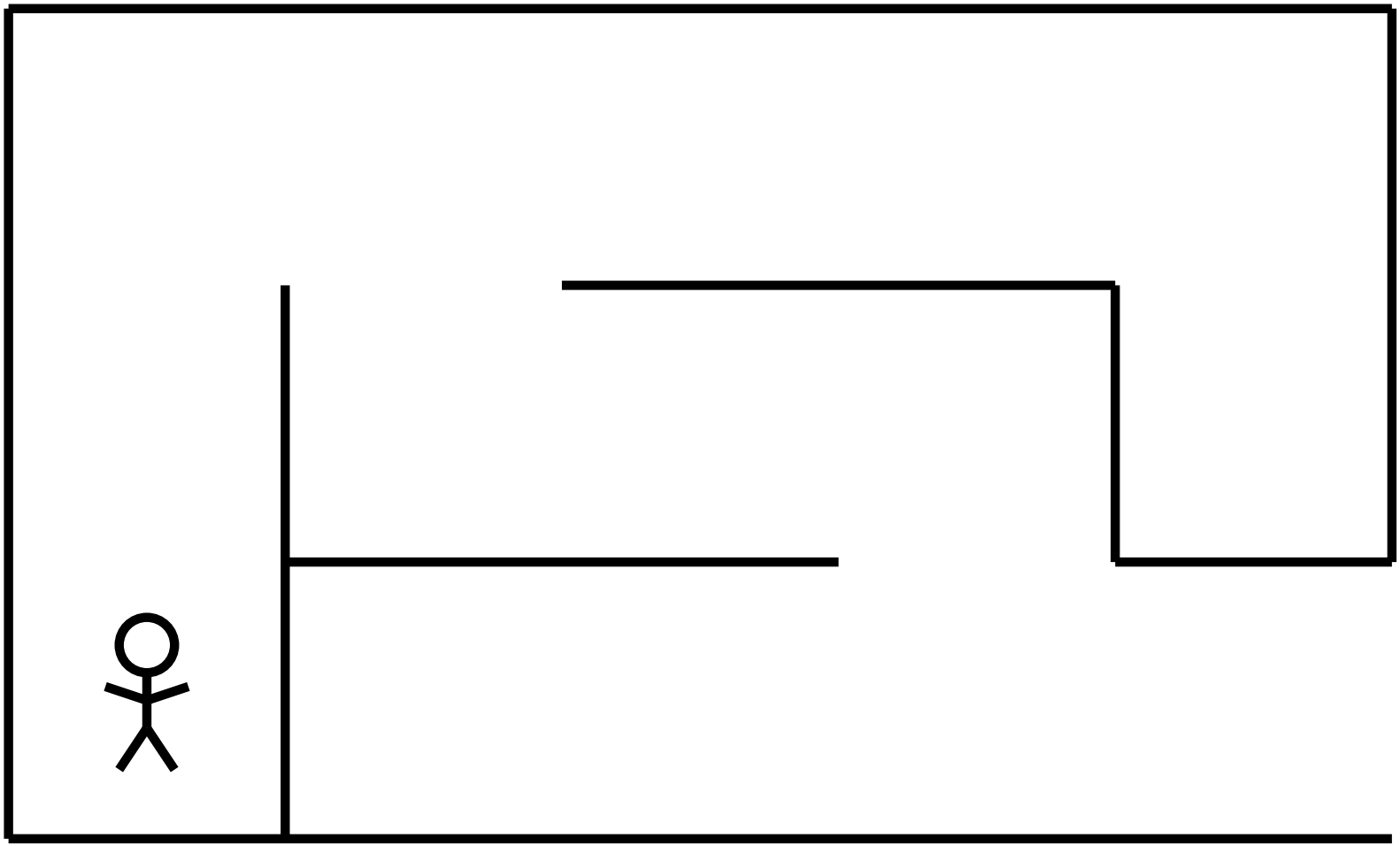
Kritische Beispiele:

- Riesige Datenmengen (Bioinformatik)
- Echtzeitanwendungen (Spiele, Flugzeugsteuerung)

Ziel der Vorlesung:

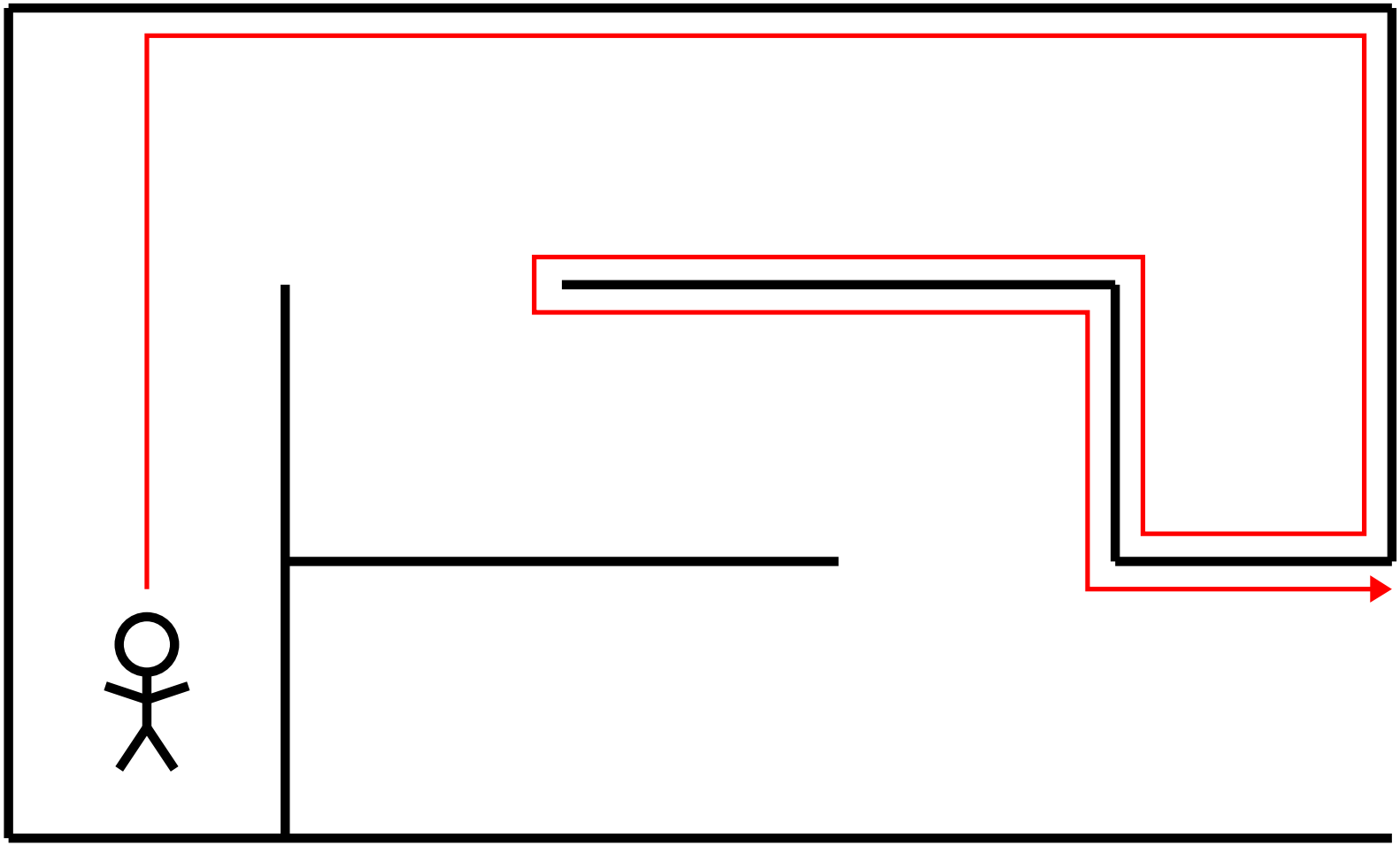
Grundstock an effizienten Algorithmen und Datenstrukturen für Standardprobleme

Weg aus dem Labyrinth



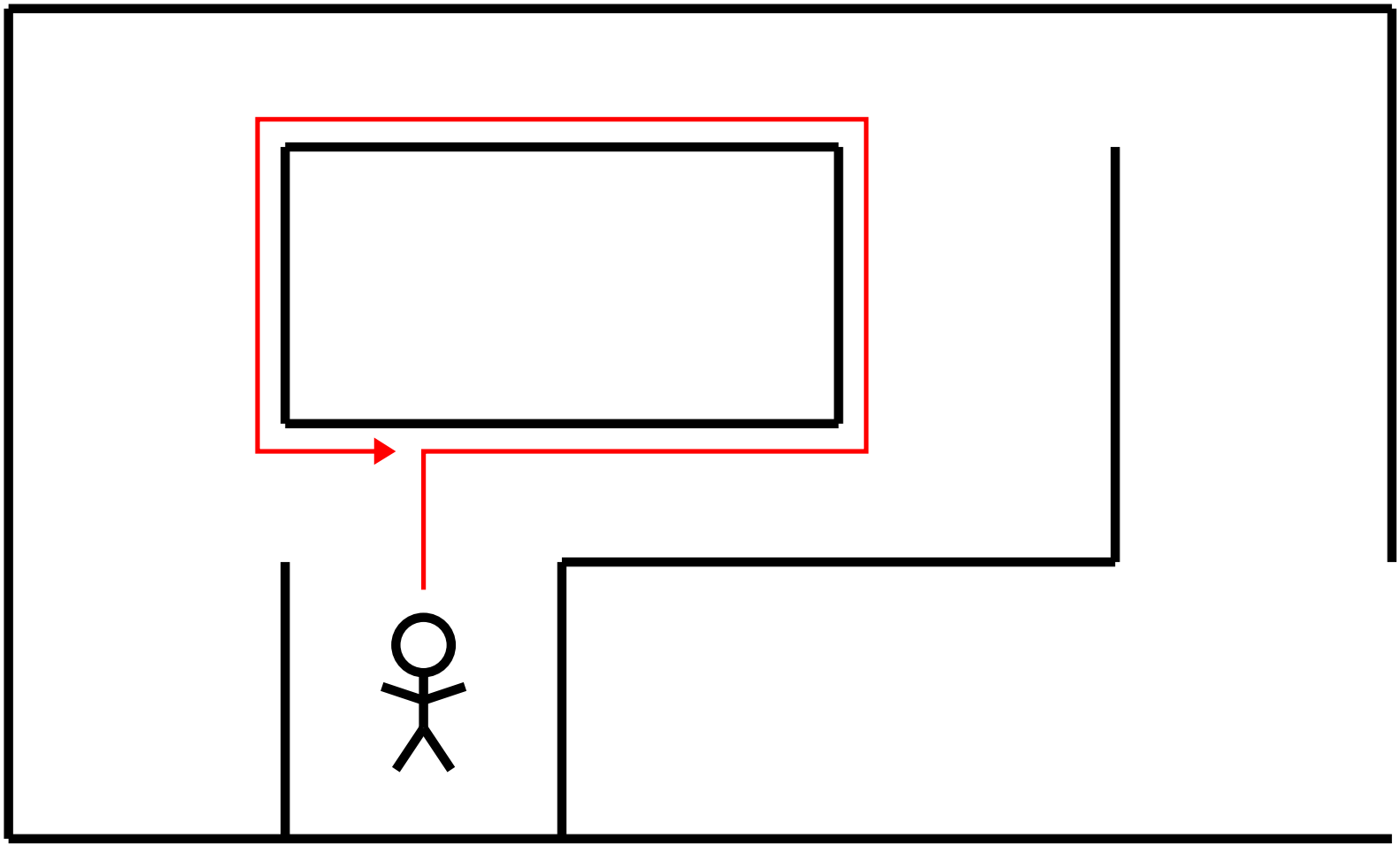
Problem: Es ist dunkel!

Weg aus dem Labyrinth



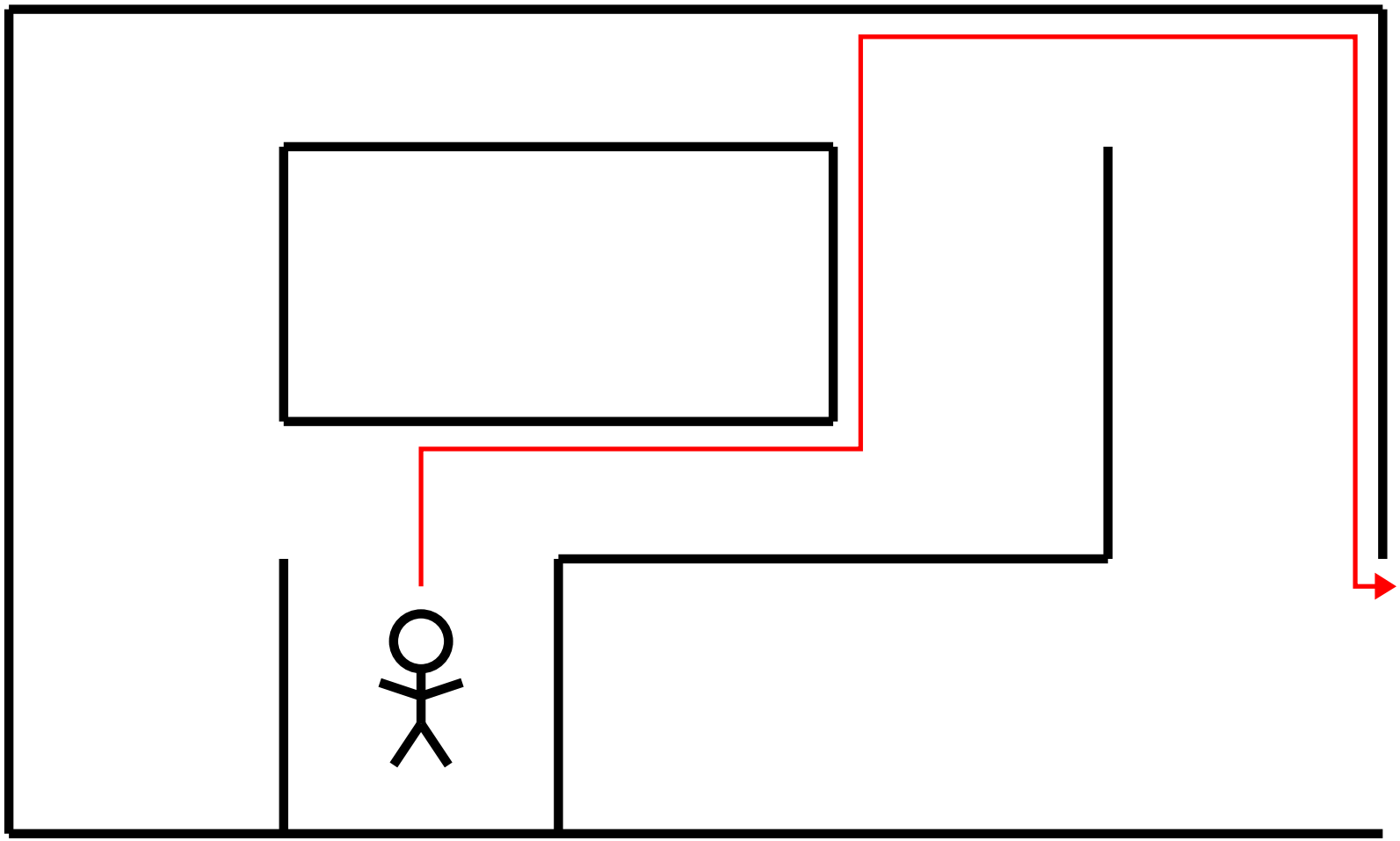
1. Versuch: mit einer Hand immer an der Wand lang

Weg aus dem Labyrinth



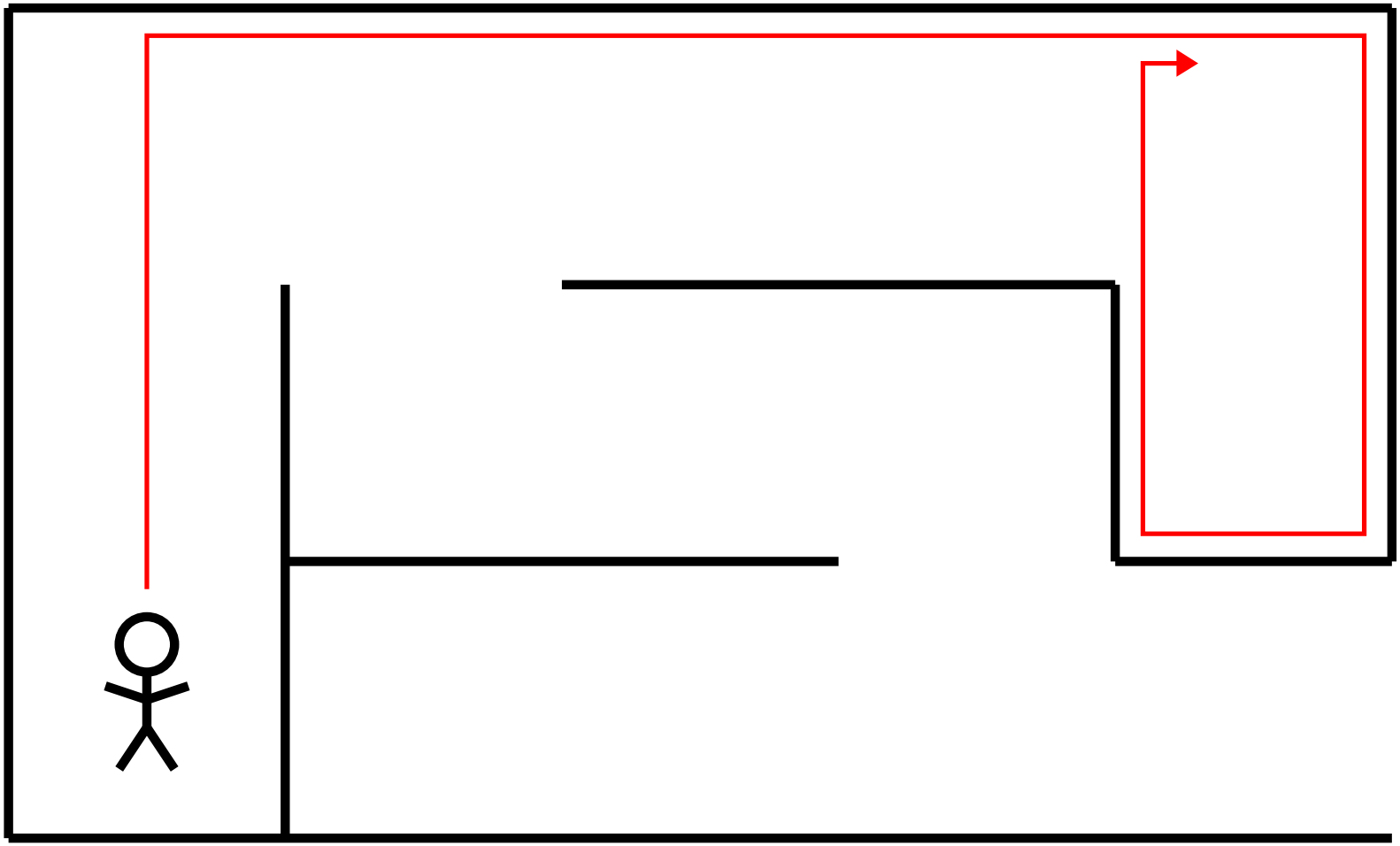
Problem: Inseln werden endlos umkreist

Weg aus dem Labyrinth



2. Versuch: gerade bis zur Wand, der Wand folgen bis man wieder in dieselbe Richtung läuft, dann wieder gerade bis zur Wand usw.

Weg aus dem Labyrinth



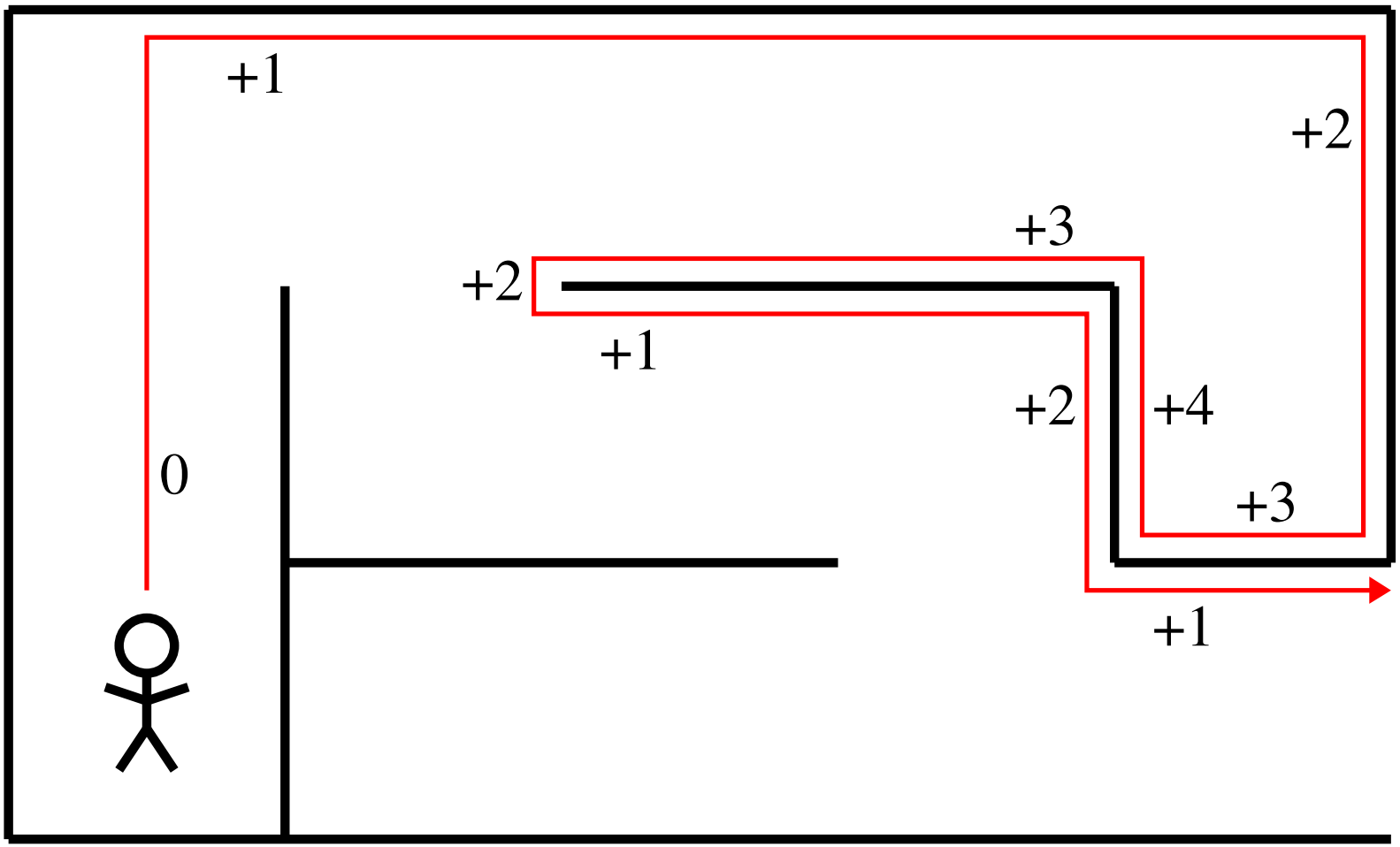
Problem: Jetzt laufen wir im ersten Beispiel im Kreis

Pledge-Algorithmus

Algorithmus 1 : Labyrinth-Suche

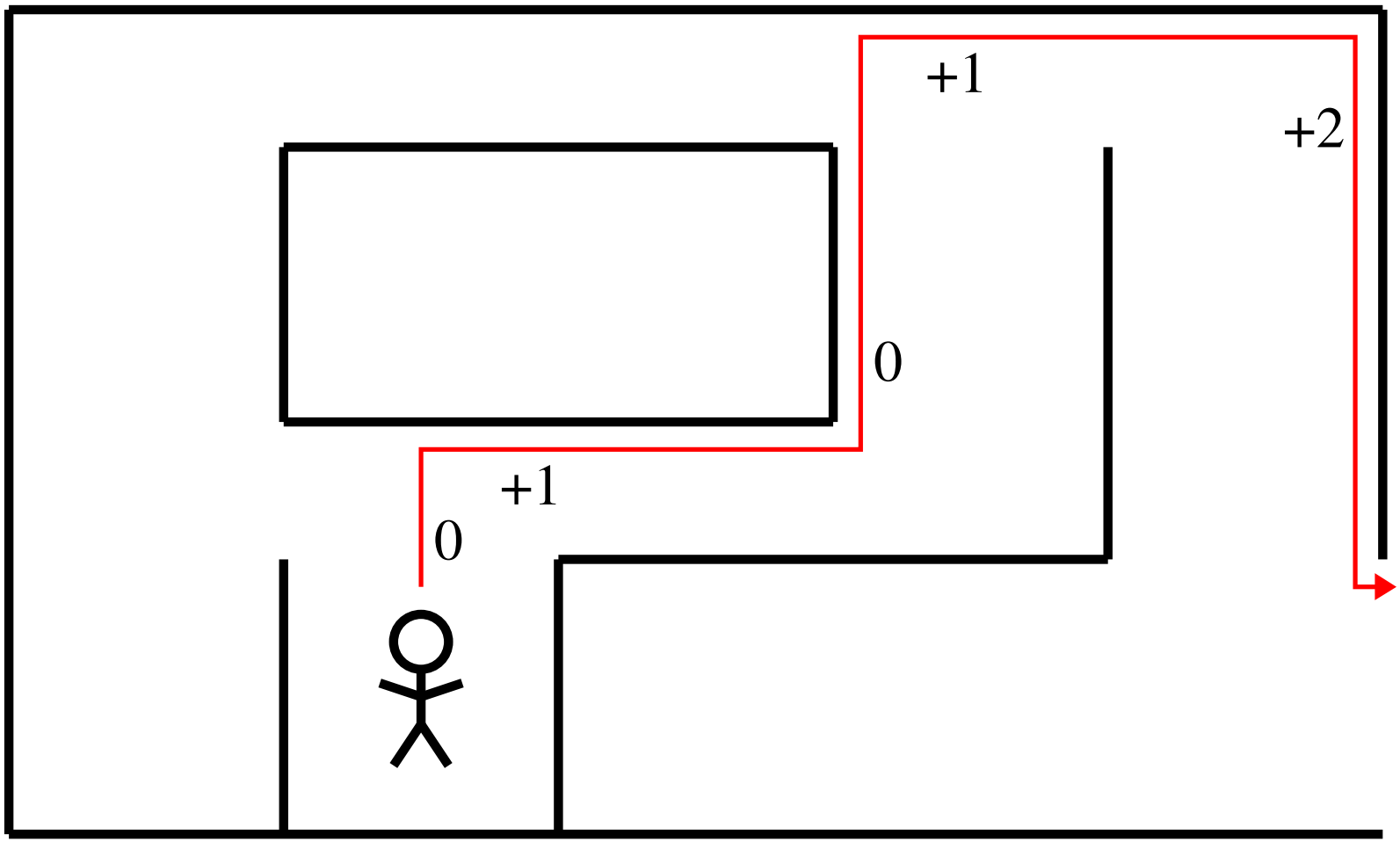
```
Setze Umdrehungszähler auf 0;
repeat
    repeat
        Gehe geradeaus;
    until Wand erreicht ;
    Drehe nach rechts;
    Aktualisiere Umdrehungszähler;
    repeat
        Folge dem Hindernis mit einer Hand;
        dabei: je nach Drehrichtung Umdrehungszähler
        inkrementieren/dekrementieren;
    until Umdrehungszähler=0 ;
until Ausgang erreicht ;
```

Weg aus dem Labyrinth



1. Beispiel funktioniert

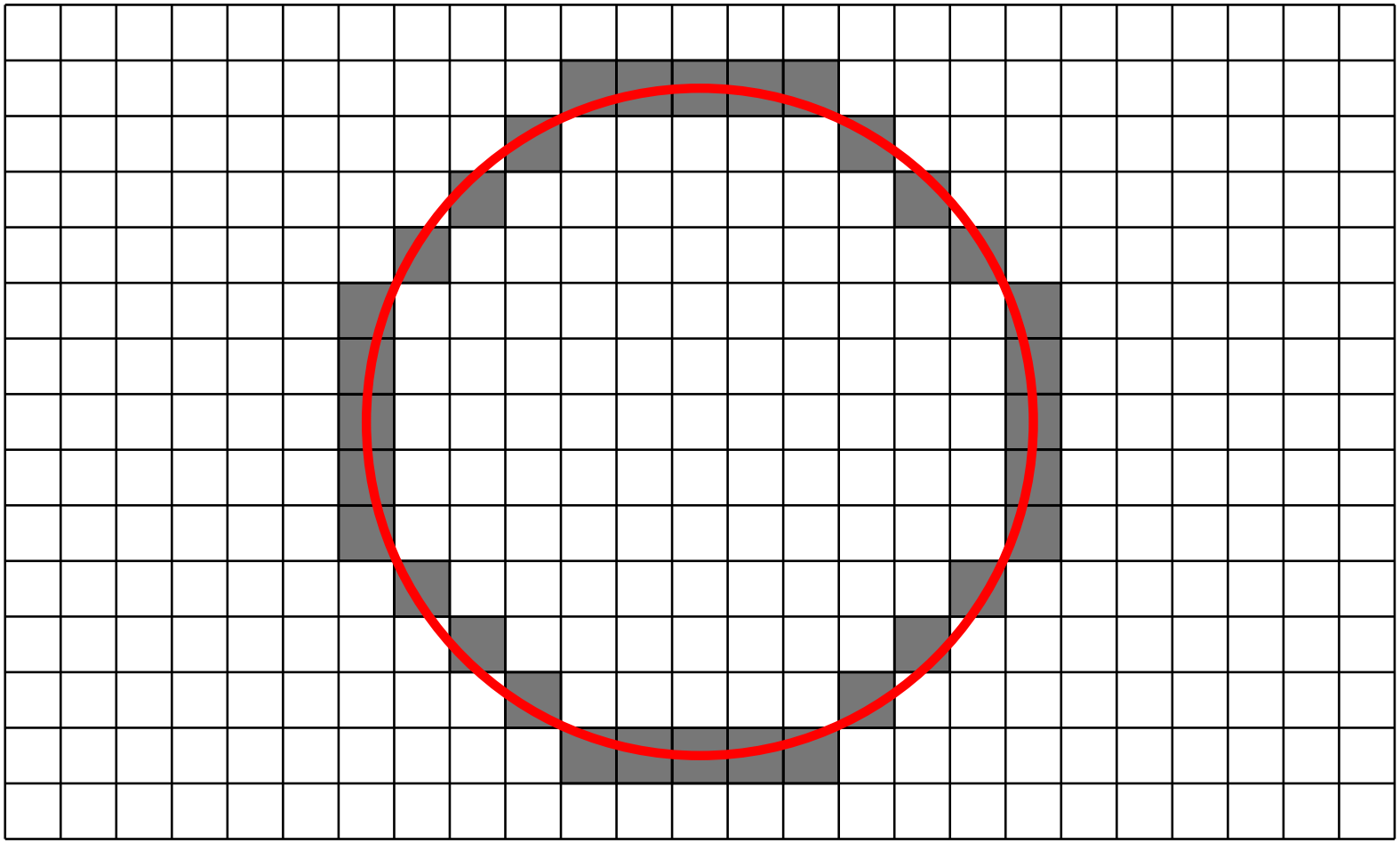
Weg aus dem Labyrinth



2. Beispiel funktioniert auch

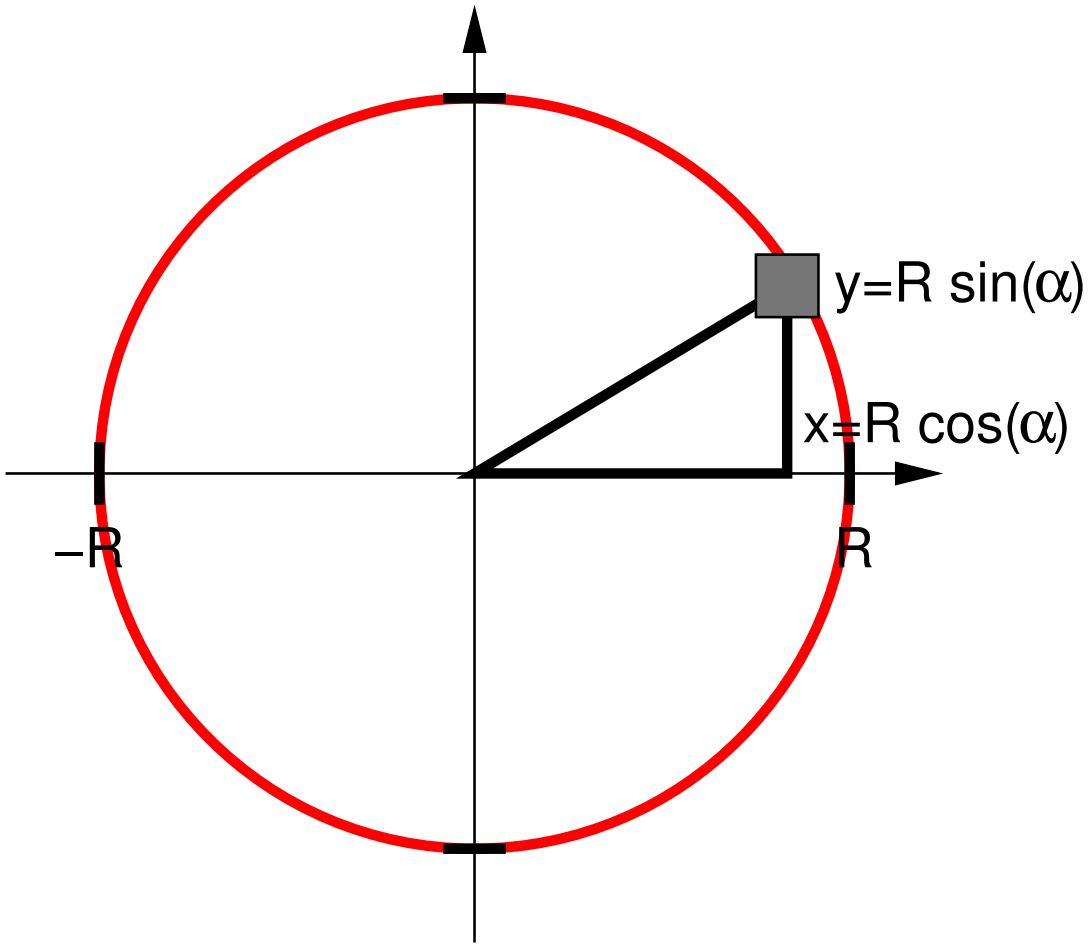
Kreis zeichnen

Wie kann ein Computer einen Kreis zeichnen?



Kreis zeichnen

Naiver Ansatz: Verwendung von sin und cos für $\alpha = 0 \ldots 2\pi$



Kreis zeichnen

Algorithmus 2 : Kreis1

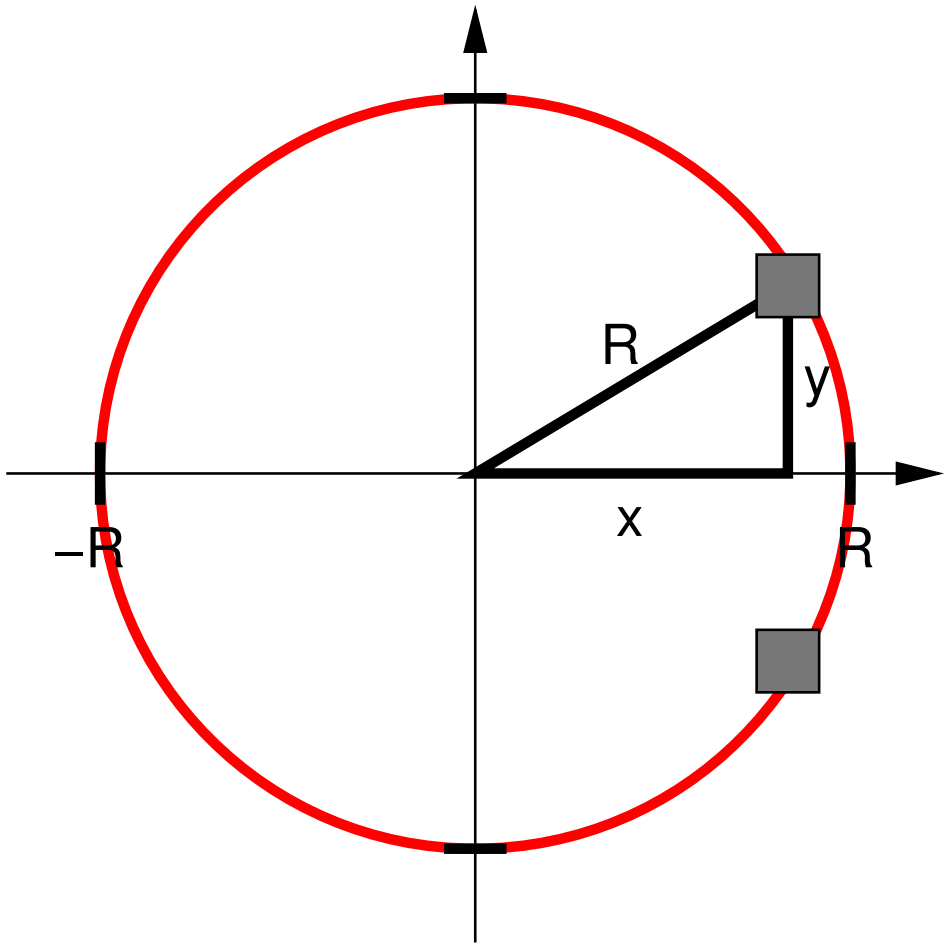
for $i = 0; i < n; i{+}{+}$ **do**
  plot$(R \cdot \cos(2\pi \cdot i/n),\ R \cdot \sin(2\pi \cdot i/n))$;

Kreisumfang: $2\pi \cdot R$
$\Rightarrow$ Bei Pixelbreite von 1 Einheit reicht $n = 7R$.

Problem: sin und cos sind teuer!

Kreis zeichnen

Besserer Ansatz: $x^2 + y^2 = R^2$ bzw. $y = \pm\sqrt{R^2 - x^2}$



Kreis zeichnen

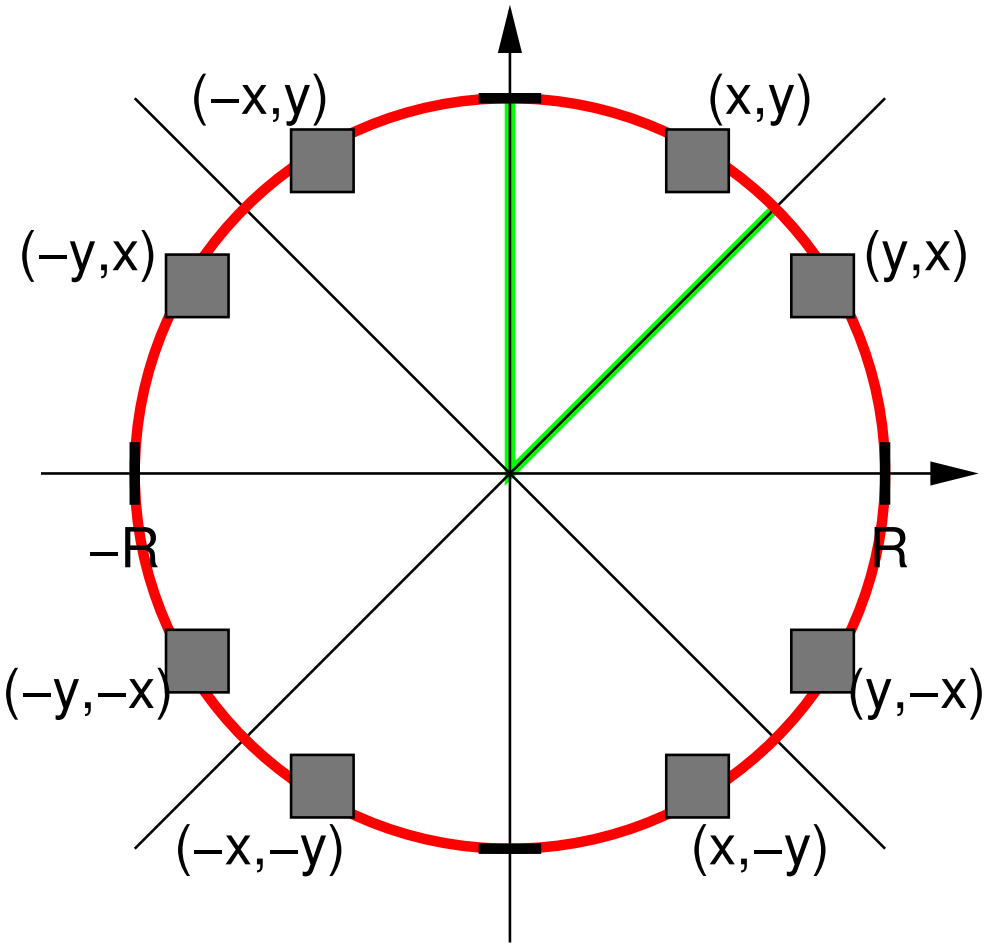
Algorithmus 3 : Kreis2

for $x = -R;\ y \leq R;\ x{+}{+}$ **do**
    $y = sqrt(R \cdot R - x \cdot x)$;
    plot(x, y);
    plot(x, -y);

Problem: *sqrt* ist auch noch relativ teuer!

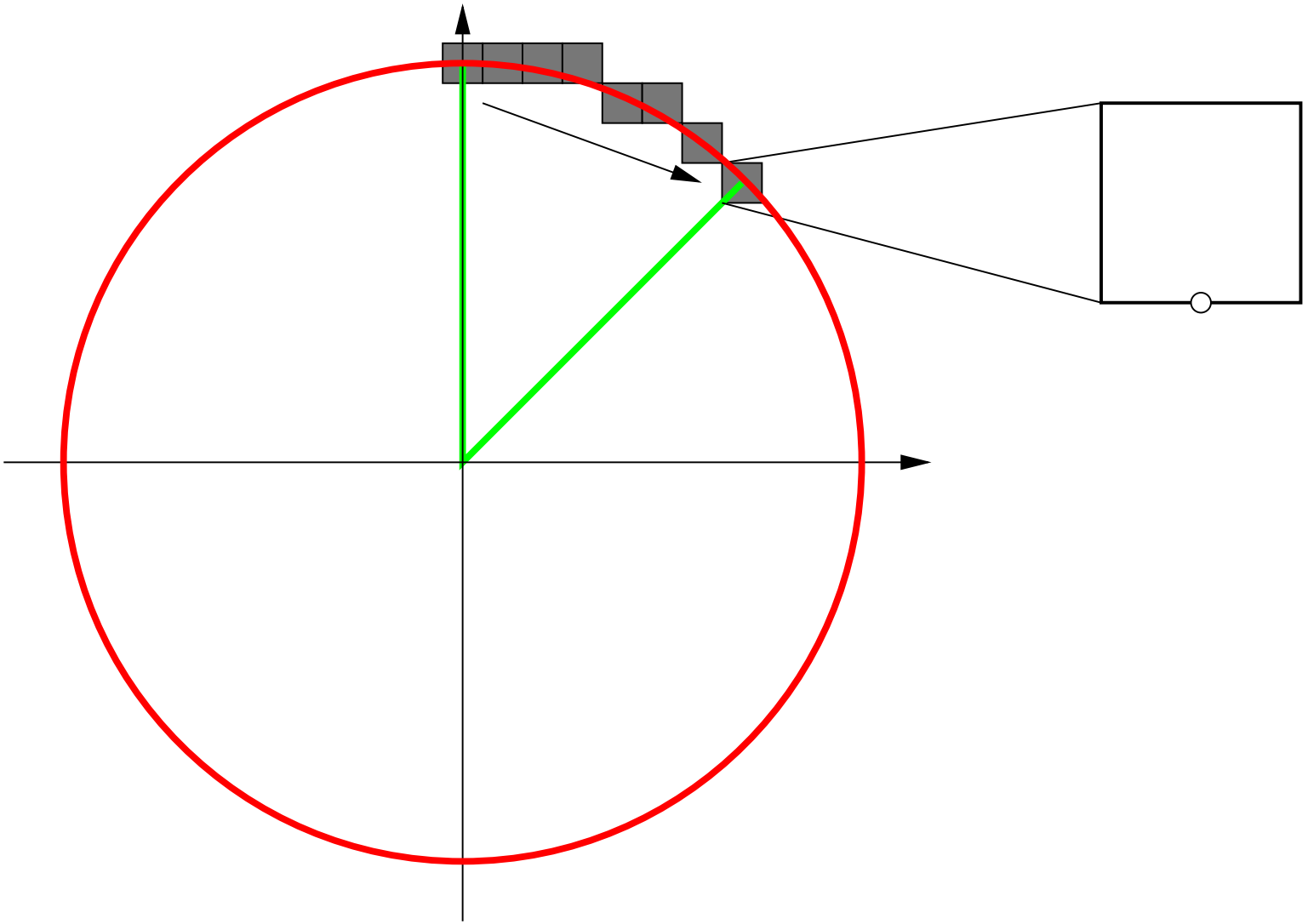
Kreis zeichnen

Besserer Ansatz: Ausnutzung von Spiegelachsen



Kreis zeichnen

Noch besserer Ansatz: Ist der Mittelpunkt der Grundlinie des Pixelquadrats innerhalb des Kreises? ja: $x \leftarrow x + 1$, nein: $x \leftarrow x + 1$, $y \leftarrow y - 1$



Kreis zeichnen

- Mittelpunkt des ersten Quadrats: $(x, y) = (0, R)$
- Position seines Grundlinienmittelpunkts: $(0, R - \frac{1}{2})$
- Test, ob (x, y) innerhalb des Kreises:

$$F(x, y) := x^2 + y^2 - R^2 < 0$$

- 1. Quadrat: $F(0, R - \frac{1}{2}) = 0^2 + (R - \frac{1}{2})^2 - R^2 = \frac{1}{4} - R < 0$?
- Quadrat rechts daneben:
  $F(1, R - \frac{1}{2}) = 1^2 + (R - \frac{1}{2})^2 - R^2 = \frac{5}{4} - R < 0$?
- Update:

$$\begin{aligned}
F(x+1, y) &= (x+1)^2 + y^2 - R^2 = (x^2 + 2x + 1) + y^2 - R^2 \\
F(x+1, y) &= F(x, y) + 2x + 1 \\
F(x+1, y-1) &= (x+1)^2 + (y-1)^2 - R^2 \\
&= (x^2 + 2x + 1) + (y^2 - 2y + 1) - R^2 \\
F(x+1, y-1) &= F(x, y) + 2x - 2y + 2
\end{aligned}$$

Kreis zeichnen

Algorithmus 4 : Bresenham1

$(x, y) = (0, R)$;
plot$(0, R)$; plot$(R, 0)$; plot$(0, -R)$; plot$(-R, 0)$;
$F = \frac{5}{4} - R$;
while $x < y$ **do**
  if $F < 0$ **then**
    $F = F + 2x + 1$;
  else
    $F = F + 2x - 2y + 2$;
    $y = y - 1$;
  $x = x + 1$;
  plot(x, y); plot(y, x); plot$(-x, y)$; plot$(y, -x)$;
  plot$(x, -y)$; plot$(-y, x)$; plot$(-x, -y)$; plot$(-y, -x)$;

Es geht sogar noch etwas schneller!

Kreis zeichnen

- Ersetzung der Korrekturterme für F:

$$F = F + 2x + 1 \quad \rightarrow \quad F = F + d_E$$
$$F = F + 2x - 2y + 2 \quad \rightarrow \quad F = F + d_{SE}$$

  mit $d_E = 2x + 1$ und $d_{SE} = 2x - 2y + 2$

- Anfangswerte:

$$\begin{aligned} d_E(0, R) &= 2 \cdot 0 + 1 = 1 \\ d_{SE}(0, R) &= 2 \cdot 0 - 2 \cdot R + 2 = 2 - 2 \cdot R \end{aligned}$$

- Update nach rechts:

$$\begin{aligned} d_E(x + 1, y) &= 2 \cdot (x + 1) + 1 = d_E(x, y) + 2 \\ d_{SE}(x + 1, y) &= 2 \cdot (x + 1) - 2 \cdot y - 2 = d_{SE}(x, y) + 2 \end{aligned}$$

- Update nach rechts unten:

$$\begin{aligned} d_E(x + 1, y - 1) &= 2 \cdot (x + 1) + 1 = d_E(x, y) + 2 \\ d_{SE}(x + 1, y - 1) &= 2 \cdot (x + 1) - 2 \cdot (y - 1) + 2 = d_{SE}(x, y) + 4 \end{aligned}$$

Kreis zeichnen

- Außerdem kann der Bruch $\frac{5}{4}$ durch 1 ersetzt werden, weil sich F immer um einen Betrag ganzer Zahlen ändert.
- D.h.

$$F = \frac{5}{4} - R + k < 0$$

ist äquivalent zu

$$F = 1 - R + k < 0$$

- Vorteil:

nur noch ganze Zahlen!

Kreis zeichnen

Algorithmus 5 : Bresenham2

$(x, y) = (0, R)$; plot$(0, R)$; plot$(R, 0)$; plot$(0, -R)$; plot$(-R, 0)$;
$F = 1 - R$;
$d_E = 1; \quad d_{SE} = 2 - 2 \cdot R$;
while $x < y$ **do**
  if $F < 0$ **then**
    $F = F + d_E$;
    $d_{SE} = d_{SE} + 2$;
  else
    $F = F + d_{SE}$;
    $y = y - 1$;
    $d_{SE} = d_{SE} + 4$;
  $x = x + 1$;
  $d_E = d_E + 2$;
  plot(x, y); plot(y, x); plot$(-x, y)$; plot$(y, -x)$;
  plot$(x, -y)$; plot$(-y, x)$; plot$(-x, -y)$; plot$(-y, -x)$;

Bresenham-Algorithmus

- wurde Anfang der 1960er Jahre von Jack Bresenham (damals bei IBM) entwickelt
- verwendet nur einfache Additionen ganzer Zahlen
- ist damit deutlich schneller als die naiven Ansätze

- diese und weitere Beispiele:

  Taschenbuch der Algorithmen (Springer, 2008)