Script generated by TTT

Title: Seidl: GAD (12.04.2016)

Date: Tue Apr 12 14:11:34 CEST 2016

Duration: 56:49 min

Pages: 38

Grundlagen: Algorithmen und Datenstrukturen

Helmut Seidl

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Institut für Informatik
Technische Universität München

Sommersemester 2016

Vorlesungsdaten

Titel: "Grundlagen: Algorithmen und Datenstrukturen" / GAD

Modul: IN0007, SWS (3+2), ECTS: 6 Credit Points

Zeiten:
Dienstag 14:00 – 16:00 Uhr (Hörsaal MW 0001)
Mittwoch 13:15 – 14:15 Uhr (Hörsaal MW 0001)

Webseite: `http://www2.in.tum.de/hp/Main?nid=304`

Voraussetzung:
IN0001 – Einführung in die Informatik 1
IN0015 – Diskrete Strukturen

Klausur:
Gesamtklausur: 06.08.2016
Wiederholungsklausur: 26.09.2016

Zielgruppe

- Bachelor Informatik
- Bachelor Wirtschaftsinformatik
- Bachelor Bioinformatik
- Bachelor Informatik: Games Engineering
- Andere Studiengänge mit Neben-/Zweitfach Informatik
- Masterstudiengang Angewandte Informatik
- Aufbaustudium Informatik
- Schülerstudium

Dozent / Kontaktdaten

- Helmut Seidl

  Lehrstuhl für Sprachen und Beschreibungsstrukturen
- eMail: seidl@in.tum.de
- Web: http://www2.in.tum.de/
- Telefon: 089 / 289-18155
- Raum: 02.07.054 (2. Stock, Finger 7)
- Sprechstunde: Donnerstag 14-15 Uhr
  (oder nach Vereinbarung)

Übung

- 2 SWS Tutorübungen
- 46 Gruppen (geplant)
- jeweils maximal 16-30 Teilnehmer
- Anmeldung zur Vorlesung über TUMonline:
  https://campus.tum.de/
- Anmeldung zu den Übungsgruppen bis **Donnerstag, 14.4.2016 um 23:00 Uhr** über Matchingsystem:
  `https://matching2.in.tum.de/m/wwo2sgp-gad-students`
- Übungsleitung:

  Julian Kranz, Andreas Reuss, Ralf Vogler
- Webseite:

  `https://www.moodle.tum.de/course/view.php?id=26768`

Übung (Forts.)

- Die Bearbeitung der Übungen ist freiwillig, aber empfehlenswert.
- Es ist ein Programmierwettbewerb geplant, auf den es u.U. Bonuspunkte gibt.
- Es gibt sowohl theoretische Aufgaben, wie Programmieraufgaben.
- Für jedes Übungsblatt gibt es Punkte.
- Für 2/3 der Gesamtpunktzahl gibt es einen Notenbonus auf die erfolgreich bestandene Klausur (oder Wiederholungsklausur).
- Die Hausaufgaben sind **selbstständig** anzufertigen! Nach Plagiaten wird automatisiert und manuell gesucht.

Übung

- 2 SWS Tutorübungen
- 46 Gruppen (geplant)
- jeweils maximal 16-30 Teilnehmer
- Anmeldung zur Vorlesung über TUMonline:
  https://campus.tum.de/
- Anmeldung zu den Übungsgruppen bis **Donnerstag, 14.4.2016 um 23:00 Uhr** über Matchingsystem:
  `https://matching2.in.tum.de/m/wwo2sgp-gad-students`
- Übungsleitung:

  Julian Kranz, Andreas Reuss, Ralf Vogler
- Webseite:

  `https://www.moodle.tum.de/course/view.php?id=26768`

Inhalt

- Grundlagen der Analyse von Effizienz / Komplexität
- Sequenzrepräsentation (dynamische Felder, Listen)
- Hashing
- Sortierverfahren
- Prioritätswarteschlangen (Binary Heaps, Binomial Heaps)
- Suchbäume (AVL-Bäume, (a, b)-Bäume)
- Graph-Repräsentation und Graphalgorithmen
- Pattern Matching
- Datenkompression

Vorlesungsdaten

Titel: "Grundlagen: Algorithmen und Datenstrukturen" / GAD

Modul: IN0007, SWS (3+2), ECTS: 6 Credit Points

Zeiten:

Dienstag	14:00 – 16:00 Uhr	(Hörsaal MW 0001)
Mittwoch	13:15 – 14:15 Uhr	(Hörsaal MW 0001)

Webseite: `http://www2.in.tum.de/hp/Main?nid=304`

Voraussetzung:

IN0001	–	Einführung in die Informatik 1
IN0015	–	Diskrete Strukturen

Klausur:

Gesamtklausur: 06.08.2016
Wiederholungsklausur: 26.09.2016

Inhalt

- Grundlagen der Analyse von Effizienz / Komplexität
- Sequenzrepräsentation (dynamische Felder, Listen)
- Hashing
- Sortierverfahren
- Prioritätswarteschlangen (Binary Heaps, Binomial Heaps)
- Suchbäume (AVL-Bäume, (a, b)-Bäume)
- Graph-Repräsentation und Graphalgorithmen
- Pattern Matching
- Datenkompression

Grundlage

- Inhalt der Vorlesung basiert auf dem Buch

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

  http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html

- Vorlage für die Slides:

  GAD SS'08: Prof. Dr. Christian Scheideler
  GAD SS'09: Prof. Dr. Helmut Seidl
  GAD SS'14: Dr. Hanjo Täubig
  Skript Alg. Bioinf.: Prof. Dr. Volker Heun

Algorithmus - Definition

Definition

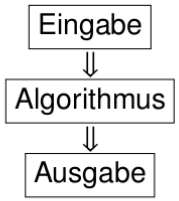
Ein Algorithmus ist eine formale Handlungsvorschrift zur Lösung von Instanzen einer bestimmten Problemklasse.

Die Bezeichnung ist abgeleitet aus dem Namen des persischen Gelehrten Muhammad ibn Musa al-Chwarizmi.

Informelle Beispiele

- Kochrezept
- Bauanleitung
- Schriftliches Rechnen
- Weg aus dem Labyrinth
- Zeichnen eines Kreises

Formalisierung (Informatik)



Abstrakter Datentyp und Datenstruktur

Abstrakter Datentyp

- legt fest, welche Operationen *was* tun (Semantik),
- aber nicht *wie* (konkrete Implementierung)

⇒ Kapselung durch Definition einer Schnittstelle

Beispiel: `PriorityQueue` mit Operationen `insert` und `deleteMin`

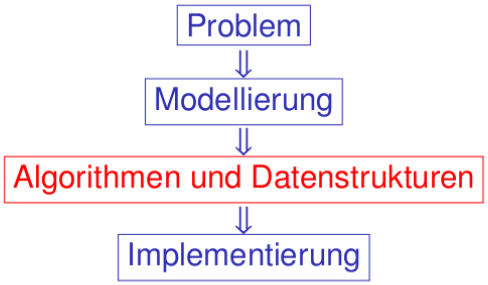
Datenstruktur: formalisiertes Objekt zur

- Speicherung,
- Verwaltung von bzw.
- Zugriff auf

Daten, die dabei geeignet angeordnet, kodiert und verknüpft werden.

Beispiel: `BinaryHeap` als konkrete Implementierung von `PriorityQueue`

Softwareentwicklung

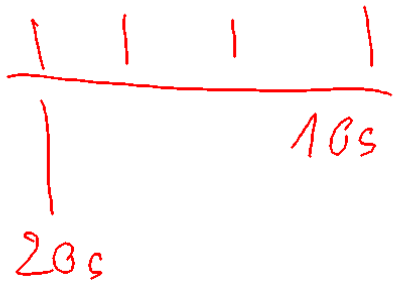


- Abstraktion vom genauen Problem (Vereinfachung)
- geeignete Auswahl von Algorithmen / Datenstrukturen
- Grundsätzliche Probleme: Korrektheit, Komplexität, Robustheit / Sicherheit, aber vor allem Effizienz

Effizienz

im Sinn von

- Laufzeit
- Speicheraufwand
- Festplattenzugriffe
- Energieverbrauch



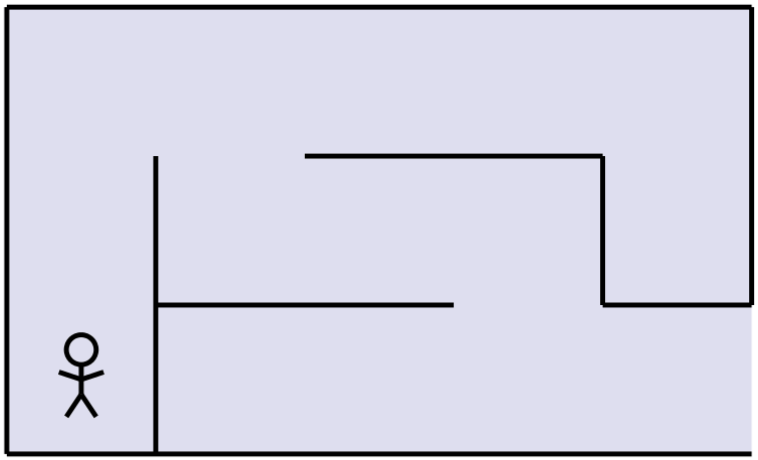
Kritische Beispiele:

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

Ziel der Vorlesung:

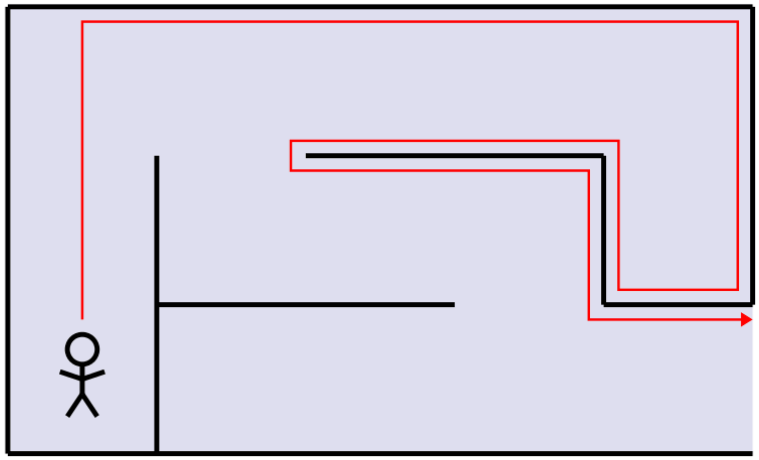
Grundstock an effizienten Algorithmen und Datenstrukturen für Standardprobleme

Weg aus dem Labyrinth



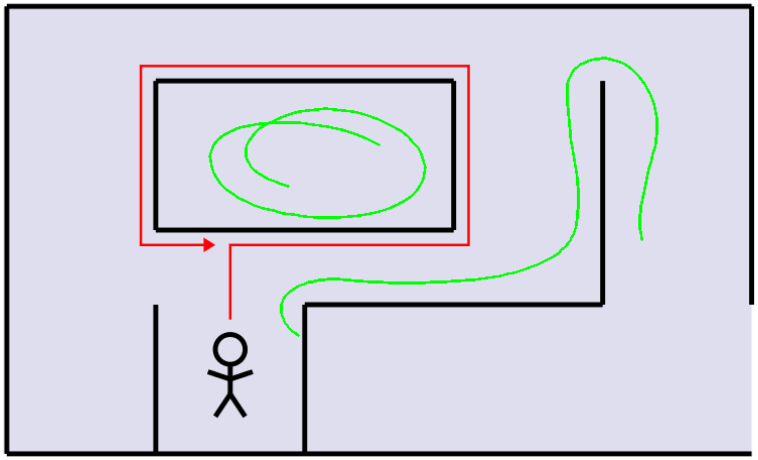
Problem: Es ist dunkel!

Weg aus dem Labyrinth



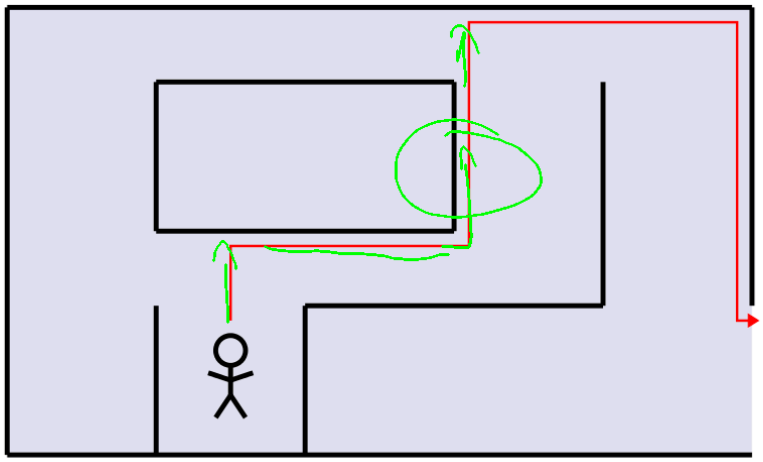
1. Versuch: mit einer Hand immer an der Wand lang

Weg aus dem Labyrinth



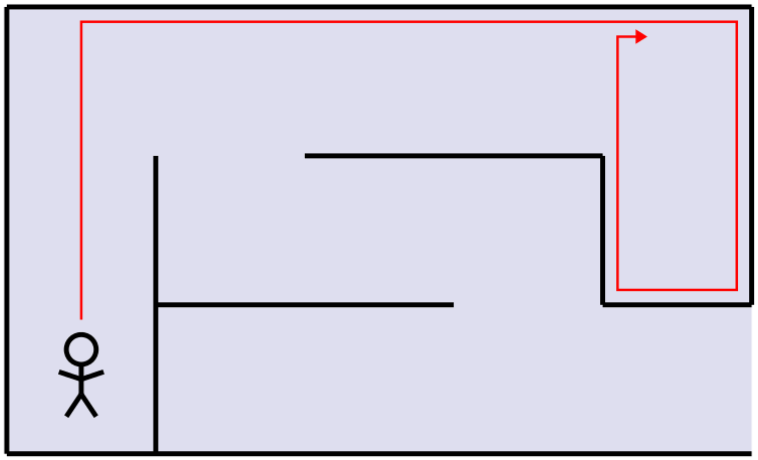
Problem: Inseln werden endlos umkreist

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



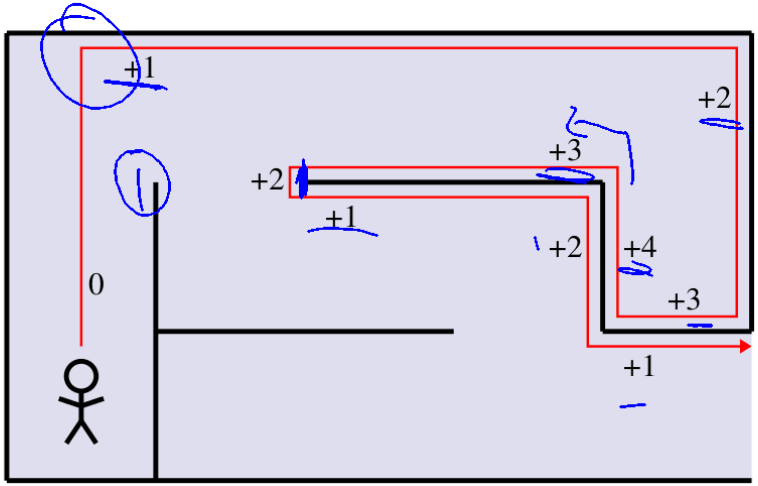
Problem: Jetzt laufen wir im ersten Beispiel im Kreis

Pledge-Algorithmus

Algorithmus Labyrinth: findet einen Ausgang

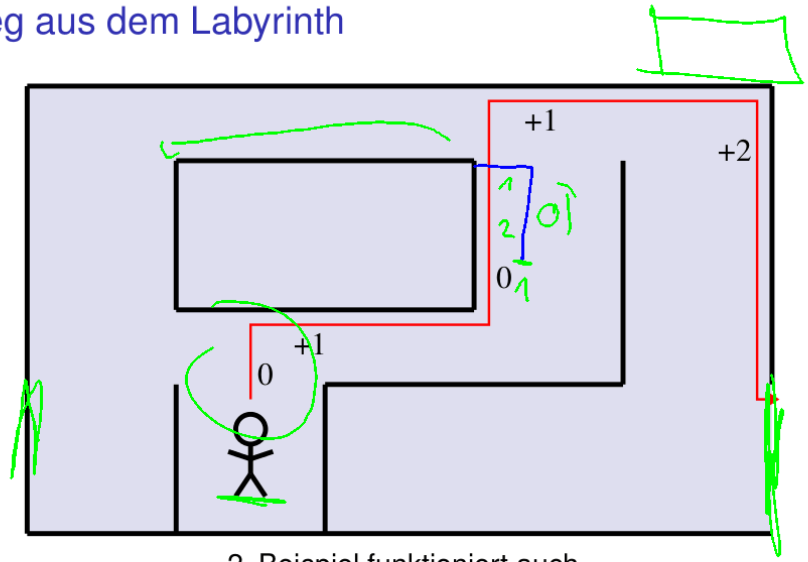
Setze Umdrehungszähler auf 0;
repeat
  repeat
    Gehe geradeaus;
  until *Wand erreicht*;
  Drehe nach rechts;
  Inkrementiere Umdrehungszähler;
  repeat
    Folge dem Hindernis mit einer Hand;
    dabei: je nach Drehrichtung Umdrehungszähler
      inkrementieren / dekrementieren;
  until *Umdrehungszähler=0*;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

Weg aus dem Labyrinth



1. Beispiel funktioniert

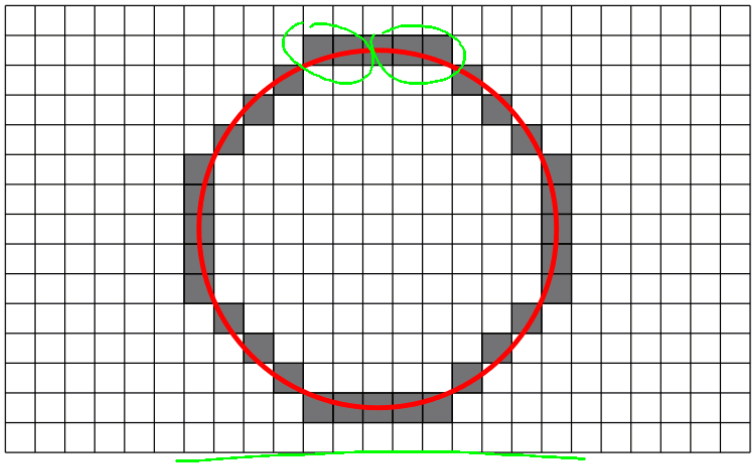
Weg aus dem Labyrinth



2. Beispiel funktioniert auch

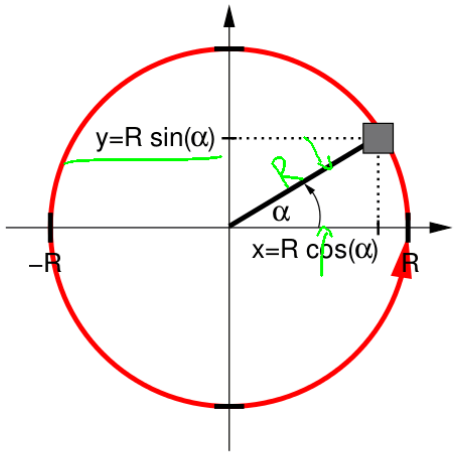
Kreis zeichnen

Wie kann ein Computer einen Kreis zeichnen?



Kreis zeichnen: mit Winkelfunktionen

Naiver Ansatz: eine Runde wie mit dem Zirkel



Verwendung von sin() und cos() für $\alpha = 0 \ldots 2\pi$

Kreis zeichnen: mit Winkelfunktionen

Algorithmus Kreis1: zeichnet Kreis mit Radius R aus n Pixeln

Eingabe : Radius R
Pixelanzahl n

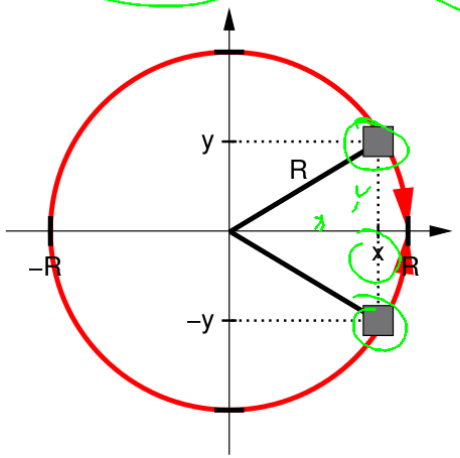
for $i = 0; i < n; i{+}{+}$ **do**
  plot($R * \cos(2\pi * i/n)$, $R * \sin(2\pi * i/n)$);

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

Problem: sin() und cos() sind teuer!

Kreis zeichnen: mit Wurzelfunktion

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



1 Pixel pro Spalte für oberen / unteren Halbkreis

Kreis zeichnen: mit Wurzelfunktion

Algorithmus Kreis2: zeichnet Kreis mit Radius R

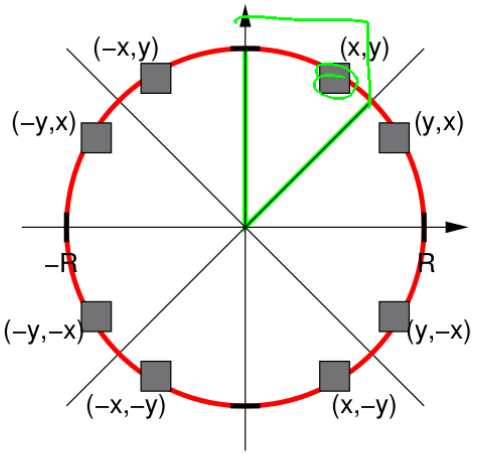
Eingabe : Radius R

```
for x = −R; x ≤ R; x++ do
    y = sqrt(R ∗ R − x ∗ x);
    plot(x, y);
    plot(x, -y);
```

Problem: sqrt() ist auch noch relativ teuer!

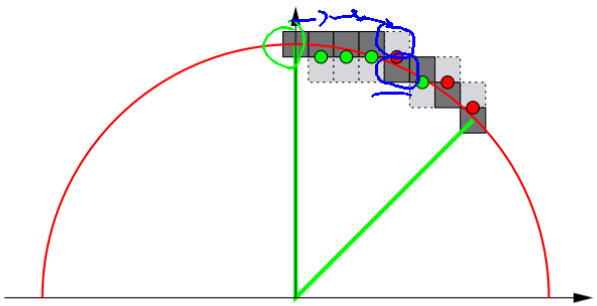
Kreis zeichnen: mit Multiplikation

Besserer Ansatz: Ausnutzung von Spiegelachsen



Kreis zeichnen: mit Multiplikation

- betrachtetes Kreissegment: Anstieg zwischen 0 und −1
- 2 Fälle für nächstes Pixel: nur rechts oder rechts unten
- Entscheidungskriterium:
  Grundlinienmittelpunkt des rechten Nachbarpixels innerhalb vom Kreis? ja: $x{+}{+}$ nein: $x{+}{+}$; $y{-}{-}$



Kreis zeichnen: mit Multiplikation

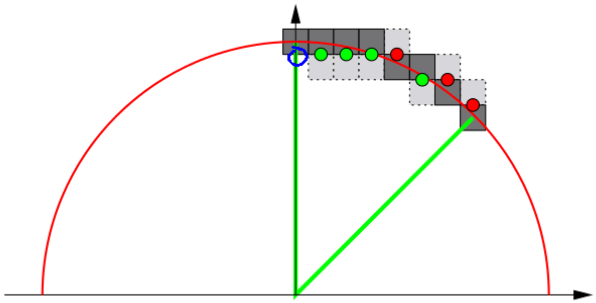
- Test, ob (x, y) innerhalb des Kreises:

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

- Mittelpunkt des ersten Quadrats: $(x, y) = (0, R)$
- Position seines Grundlinienmittelpunkts: $(0, R - \frac{1}{2})$
- Grundlinienmittelpunkt für Pixel rechts daneben:
  $F(1, R - \frac{1}{2}) = 1^2 + (R - \frac{1}{2})^2 - R^2 = \frac{5}{4} - R < 0$?

Kreis zeichnen: mit Multiplikation

- betrachtetes Kreissegment: Anstieg zwischen 0 und −1
- 2 Fälle für nächstes Pixel: nur rechts oder rechts unten
- Entscheidungskriterium:
  Grundlinienmittelpunkt des rechten Nachbarpixels innerhalb vom Kreis? ja: $x{+}{+}$ nein: $x{+}{+}$; $y{-}{-}$



Kreis zeichnen: mit Multiplikation

- Test, ob (x, y) innerhalb des Kreises:

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

- Mittelpunkt des ersten Quadrats: $(x, y) = (0, R)$
- Position seines Grundlinienmittelpunkts: $(0, R - \frac{1}{2})$
- Grundlinienmittelpunkt für Pixel rechts daneben:
  $F(1, R - \frac{1}{2}) = 1^2 + (R - \frac{1}{2})^2 - R^2 = \frac{5}{4} - R < 0$?

Kreis zeichnen: mit Multiplikation

- Test, ob (x, y) innerhalb des Kreises:

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

- Mittelpunkt des ersten Quadrats: $(x, y) = (0, R)$
- Position seines Grundlinienmittelpunkts: $(0, R - \frac{1}{2})$
- Grundlinienmittelpunkt für Pixel 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: mit Multiplikation

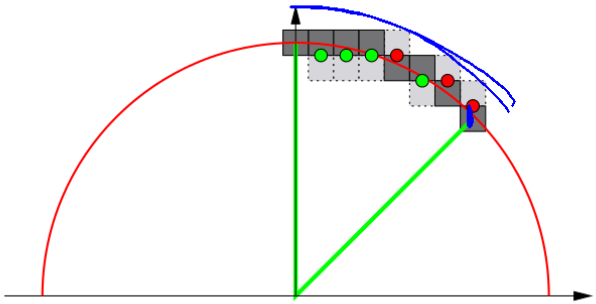
Algorithmus Bresenham1: zeichnet Kreis mit Radius R

```
x = 0;  y = R;
plot(0, R);  plot(R, 0);  plot(0, −R);  plot(−R, 0);
F = 5/4 − R;
while x < y do
    if F < 0 then
        F = F + 2 * x + 1;
    else
        F = F + 2 * x − 2 * y + 2;
        y = y − 1;
    x = x + 1;
    plot(x, y);  plot(−x, y);  plot(−y, x);  plot(−y, −x);
    plot(y, x);  plot(y, −x);  plot(x, −y);  plot(−x, −y);
```

Es geht sogar noch etwas schneller!

Kreis zeichnen: mit Multiplikation

- betrachtetes Kreissegment: Anstieg zwischen 0 und −1
- 2 Fälle für nächstes Pixel: nur rechts oder rechts unten
- Entscheidungskriterium:
  Grundlinienmittelpunkt des rechten Nachbarpixels innerhalb vom Kreis? ja: $x{+}{+}$ nein: $x{+}{+}$; $y{-}{-}$



Kreis zeichnen: mit Multiplikation

Algorithmus Bresenham1: zeichnet Kreis mit Radius R

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

Es geht sogar noch etwas schneller!

Kreis zeichnen: mit Multiplikation

- Test, ob (x, y) innerhalb des Kreises:

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

- Mittelpunkt des ersten Quadrats: $(x, y) = (0, R)$
- Position seines Grundlinienmittelpunkts: $(0, R - \frac{1}{2})$
- Grundlinienmittelpunkt für Pixel rechts daneben:
  $F(1, R - \frac{1}{2}) = 1^2 + (R - \frac{1}{2})^2 - R^2 = \frac{5}{4} - R < 0$?
- Update:

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