Script generated by TTT

Title: Seidl: GAD (14.04.2015)

Date: Tue Apr 14 13:47:57 CEST 2015

Duration: 143:55 min

Pages: 78

Grundlagen: Algorithmen und Datenstrukturen

Helmut Seidl

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Institut für Informatik
Technische Universität München

Sommersemester 2015

Vorlesungsdaten

- Titel: "Grundlagen: Algorithmen und Datenstrukturen" / GAD
- SWS: 3 (Vorlesung) + 2 (Übung)
- Modul: IN0007, ECTS: 6 Credit Points
- Vorlesungszeiten:
  Dienstag 13:45 – 16:15 Uhr (Hörsaal MW 0001)
- Webseite: http://www2.in.tum.de/hp/Main?nid=273
- Voraussetzung: IN0001 – Einführung in die Informatik 1
  Empfehlung: IN0015 – Diskrete Strukturen
- Klausur:
  Gesamtklausur: Samstag, 25.07.2015 (11:00–13:30 Uhr)
  Wiederholungsklausur: Freitag, 25.09.2015 (11:00–13:30 Uhr)

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
- 38 Gruppen an 17 verschiedenen Terminen
- jeweils maximal 16-30 Teilnehmer
- Anmeldung zur Vorlesung über TUMonline:
  https://campus.tum.de/
- Anmeldung zu den Übungsgruppen:
  über Matchingsystem
- Übungsleitung:

  Julian Kranz, Andreas Reuss, Ralf Vogler
- Webseite:

  https://www.moodle.tum.de/course/view.php?id=21664

Übung

- 2 SWS Tutorübungen
- 38 Gruppen an 17 verschiedenen Terminen
- jeweils maximal 16-30 Teilnehmer
- Anmeldung zur Vorlesung über TUMonline:
  https://campus.tum.de/
- Anmeldung zu den Übungsgruppen:
  über Matchingsystem
- Übungsleitung:

  Julian Kranz, Andreas Reuss, Ralf Vogler
- Webseite:

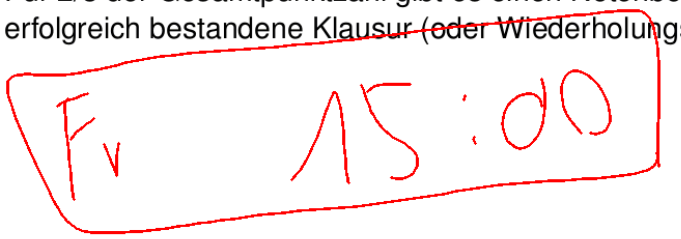
  https://www.moodle.tum.de/course/view.php?id=21664

Übung (Forts.)

- Die Bearbeitung der Übungen ist freiwillig, aber empfehlenswert.
- 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).

Übung (Forts.)

- Die Bearbeitung der Übungen ist freiwillig, aber empfehlenswert.
- 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).



Informationen zu den Programmieraufgaben für GAD

Philipp Hoffmann, Christian Müller, Chris Pinkau, Stefan Toman

Offizielles DOMjudge System

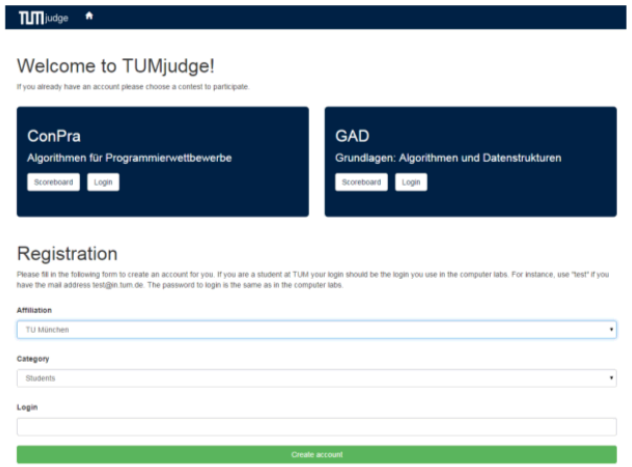
- In Gebrauch für Programmierwettbewerbe wie den GCPC oder den ICPC.
- Web-Oberfläche:

TUMjudge

`http://judge.informatik.tu-muenchen.de/`

Registrierung

- Registrierung erforderlich.

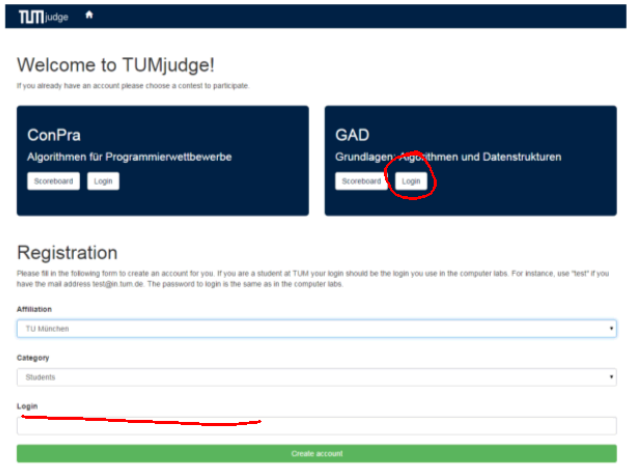


Login

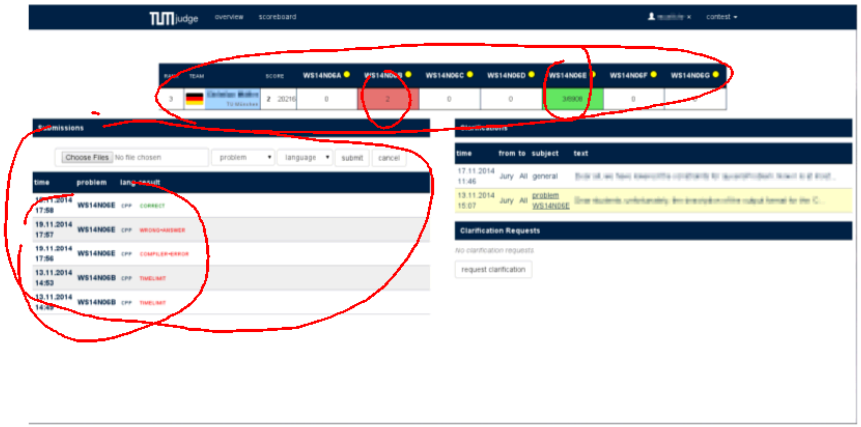
- Authentifizierung erforderlich.
- Funktioniert über LDAP und ist daher identisch mit dem Rechnerhallen-Login:
  - Login ist "name" in `name@in.tum.de`.
  - Passwort ist das zugehörige Passwort.
- Passwort vergessen/ändern/zurücksetzen?
  - Zur RBG gehen, nicht zu uns!

Registrierung

- Registrierung erforderlich.



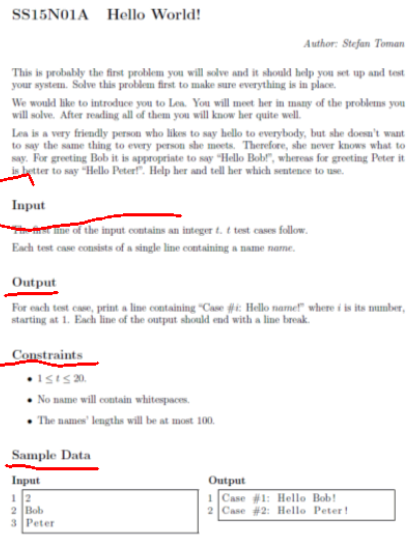
Overview



Aufgabenaufbau

Eine Aufgabe besteht aus mehreren Teilen:

- Name, Kürzel, Schwierigkeit,
- Aufgabentext,
- Angabe über das Eingabeformat,
- Angabe über das Ausgabeformat,
- Variablenbeschränkungen (Constraints),
- Beispiel-Ein- und Ausgabe.



SS15N01A Hello World!

Author: Stefan Toman

This is probably the first problem you will solve and it should help you set up and test your system. Solve this problem first to make sure everything is in place.

We would like to introduce you to Lea. You will meet her in many of the problems you will solve. After reading all of them you will know her quite well.

Lea is a very friendly person who likes to say hello to everybody, but she doesn't want to say the same thing to every person she meets. Therefore, she never knows what to say. For greeting Bob it is appropriate to say "Hello Bob!", whereas for greeting Peter it is better to say "Hello Peter!". Help her and tell her which sentence to use.

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a single line containing a name *name*.

Output

For each test case, print a line containing "Case #i: Hello *name*!" where i is its number, starting at 1. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$.
- No name will contain whitespaces.
- The names' lengths will be at most 100.

Sample Data

Input	Output
2 Bob Peter	Case #1: Hello Bob! Case #2: Hello Peter!

Programme hochladen

Programme hochladen (submit) funktioniert komplett über den TUMjudge.

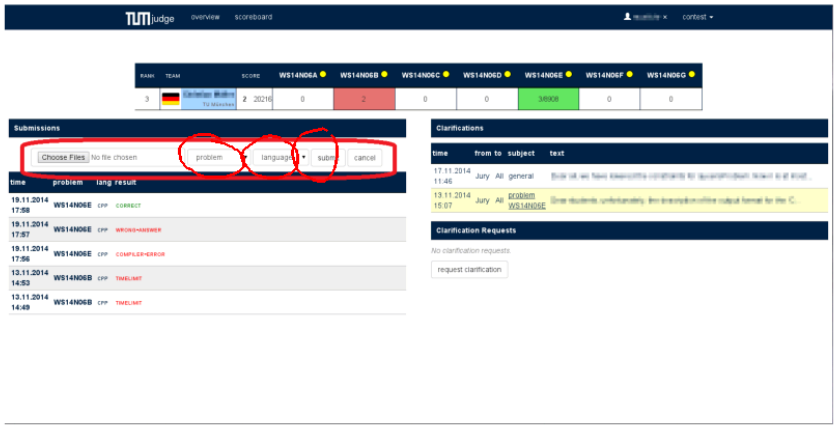
- Keine Dateien per Mail o.ä. hin- und herschicken.
- Hochgeladen wird nur der Quelltext, keine .class-Dateien o.ä.

Submit

- Per Drag-and-Drop oder per Auswahl im Menü ("Choose Files") Datei(en) auswählen,
- Problem auswählen,
- Sprache (soweit nicht automatisch erkannt) einstellen,
- "submit",
- F5, F5, F5, F5, ...

ACHTUNG: Bei Lösungen in mehreren Dateien muss im Auswahlmenü die Datei mit der main-Methode zuerst ausgewählt werden.

Aufgaben hochladen



Judging

Der TUMjudge

- kompiliert,
- führt aus,
- testet

das hochgeladene Programm gegen mehrere Testfälle. Solange der TUMjudge nicht fertig ist erscheint "PENDING" als Status der Submission.
Das Programm wird sofort getestet und der TUMjudge gibt (im Normalfall) die Antwort in wenigen Sekunden zurück.

Judging

Folgende Antworten können auftreten:

CORRECT

Die Submission hat alle Testfälle erfolgreich gelöst.

COMPILER-ERROR

Das Programm konnte nicht kompiliert werden. Auf der Detailseite der Submission kann man den genauen Fehler einsehen.

NO-OUTPUT

Das Programm produziert schlicht keine Ausgabe. Ausgabe zu "standard out" nicht vergessen!

Judging

TIMELIMIT

Das Programm läuft länger als die maximal erlaubte Zeit und wurde daher abgebrochen.
Mögliche Gründe:

- Das Programm hängt in einer Endlosschleife.
- Das Programm ist nicht effizient genug.

RUN-ERROR

Fehler während der Ausführung des Programms.
Mögliche Gründe:

- Division durch 0.
- Speicher-Indizierung mit falschem Index, z.B. ArrayIndexOutOfBounds.
- Speichernutzung über dem erlaubten Maximum.

Judging

WRONG-ANSWER

Die Ausgabe des Programms ist nicht korrekt.
Mögliche Gründe:

- Die Antwort ist einfach falsch.
- Die Antwort entspricht nicht der gleichen Formatierung wie in der Aufgabenstellung gegeben.
- Die Antwort ist nicht genau genug (bei Gleitkommazahl-Lösungen).

TOO-LATE

Das Programm wurde erst nach der Deadline hochgeladen. Sie wird zwar vom System gespeichert, aber zählt nicht mehr.

Scoreboard

Verschiedene Hintergrundfarben der Submissions:

- Problem gelöst.
- Problem als Erste/r gelöst.
- Falsche Submission(s).
- Submission in Bearbeitung: PENDING.
- Keine Submissions.

Scoreboard

Reihenfolge:

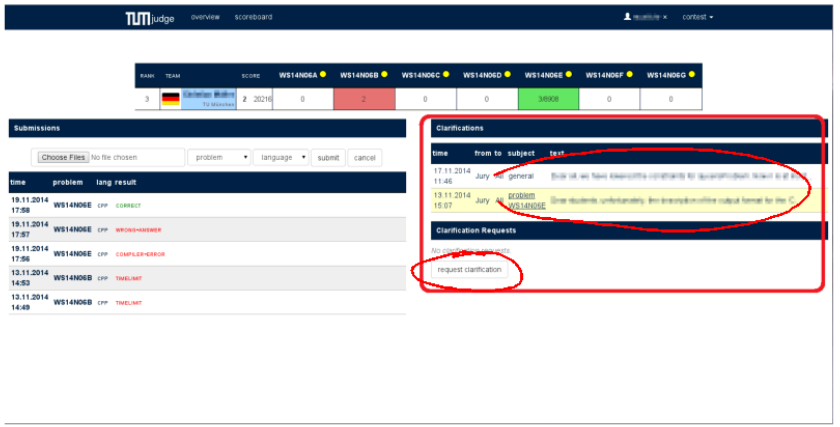
1. Anzahl gelöster Aufgaben,
2. Score:
   - pro Aufgabe: (Anzahl falscher Submissions) $*$ Strafzeit $+$ (Zeit der richtigen Submission),
   - z.B. 3/421 bedeutet: das Problem wurde mit der 3. Submission gelöst, mit einer Strafzeit von insgesamt 421.

Es dürfen zu einer Aufgabe beliebig viele Submissions eingeschickt werden!
Der Score beinflusst nicht den Bonus am Ende des Semesters, nur die Position im Scoreboard dieser Woche!

Scoreboard

Wer nicht im öffentlichen Scoreboard erscheinen will, wählt dies bei der Registrierung aus (bei “Category”).

Clarifications



Beschränkungen

- Die maximale Kompilierzeit beträgt 30 Sekunden, danach wird die Kompilierung abgebrochen und ein COMPILER-ERROR ausgegeben.
- Die maximal erlaubte Größe eines Quelltextes beträgt 256 Kilobyte. Ansonsten wird die Submission nicht angenommen.
- Der maximal verfügbare Speicherplatz beträgt 2 Gigabyte. Dies beinhaltet alles: Quelltext, Variablen, Stack, Java VM (bis zu 0,35 Gigabyte!),... Falls mehr Speicher gebraucht werde würde, wird das Programm beendet und erzeugt einen RUN-ERROR.
- Es ist kein Multi-Threading verfügbar. Jede Submission hat nur einen Prozessor voll zur Verfügung.

Beschränkungen

Jegliche Sabotage des Systems wird geahndet!

- Keine Dateien öffnen! Die Eingabe kommt immer über “standard in”.
- Keine Zugriffe auf lokale Dateien auf dem System! Das wäre ohnehin nicht möglich.
- Keine Netzwerkverbindungen öffnen!
- ...

Außerdem sollte die Anzahl an Submissions gering gehalten werden um die Bewertungszeit für alle Teilnehmer nicht unnötig zu verlangsamen.

JavaSubmission

```
import java.util.Scanner;

public class JavaSubmission {
   public static void main(String[] args) {
      // create scanner object
      Scanner s = new Scanner(System.in);

      // loop over all test cases
      int t = s.nextInt();
      for(int i = 1; i <= t; i++) {

         // read several types of input
         boolean b = s.nextBoolean();
         String st = s.next();

         // output: use the possibility you like more
         System.out.println("Case #"+i+": "+st);
         System.out.format("Case #%d: %s\n", i, s);
      }
      s.close();
   }
}
```

Informationen zu den Programmieraufgaben für GAD
Beispielquelltext

JavaSubmission

import java.util.Scanner;

public class JavaSubmission {
   public static void main(String[] args) {
      // create scanner object
      Scanner s = new Scanner(System.in);

      // loop over all test cases
      int t = s.nextInt();
      for(int i = 1; i <= t; i++) {

         // read several types of input
         boolean b = s.nextBoolean();
         String st = s.next();

         // output: use the possibility you like more
         System.out.println("Case #"+i+": "+st);
         System.out.format("Case #%d: %s\n", i, s);
      }
      s.close();
   }
}

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

Organisatorisches Inhalt

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

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

2 Einführung
- Begriffe: Algorithmus, Datenstruktur, Effizienz
- Beispiele

Algorithmus - Definition

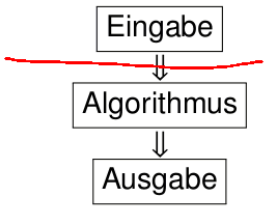
Definition

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

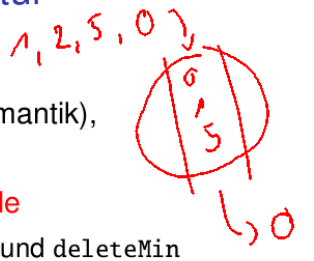
Die Bezeichnung ist abgeleitet aus dem Namen des persischen Gelehrten Muhammad ibn Musa al-Chwarizmi.

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

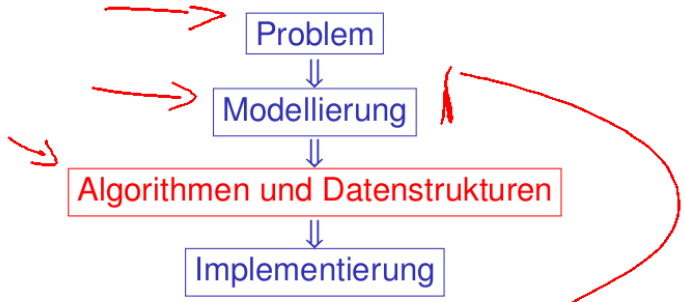
Beispiel: `PriorityQueue` mit Operationen `insert` und `deleteMin`

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)

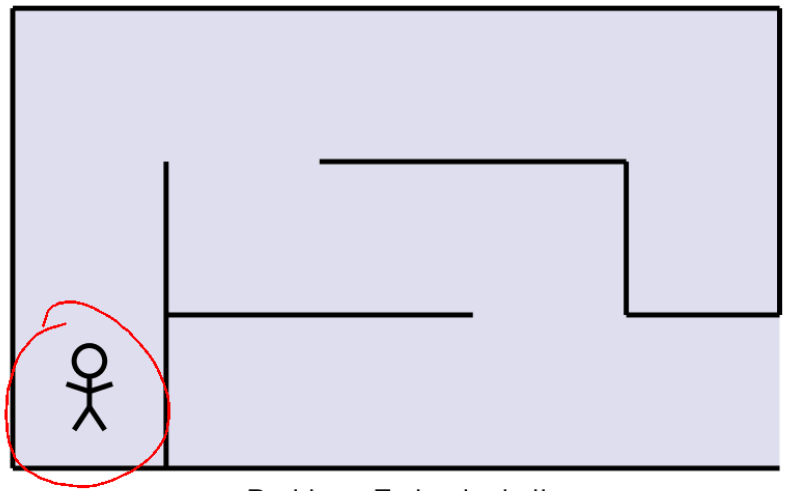
Ziel der Vorlesung:

Grundstock an effizienten Algorithmen und Datenstrukturen für Standardprobleme

Übersicht

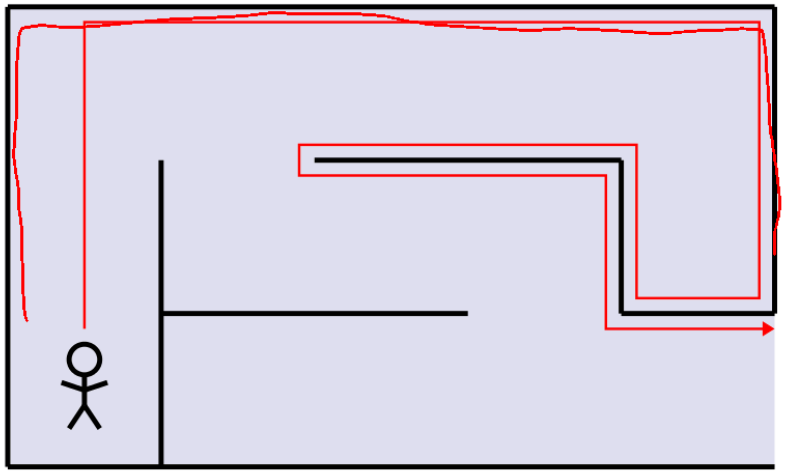
2 Einführung
- Begriffe: Algorithmus, Datenstruktur, Effizienz
- Beispiele

Weg aus dem Labyrinth



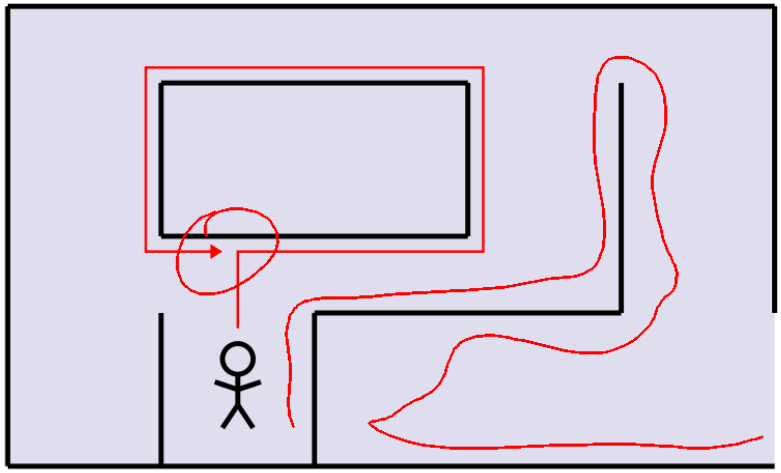
Problem: Es ist dunkel!

Weg aus dem Labyrinth



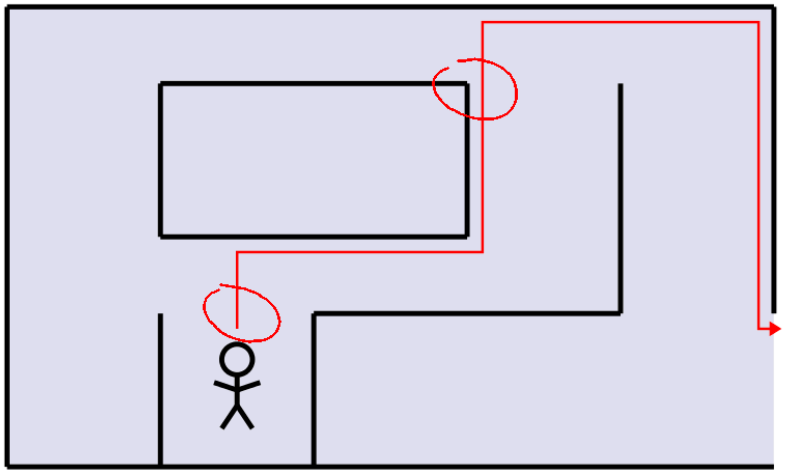
1. Versuch: mit einer Hand immer an der Wand lang

Weg aus dem Labyrinth



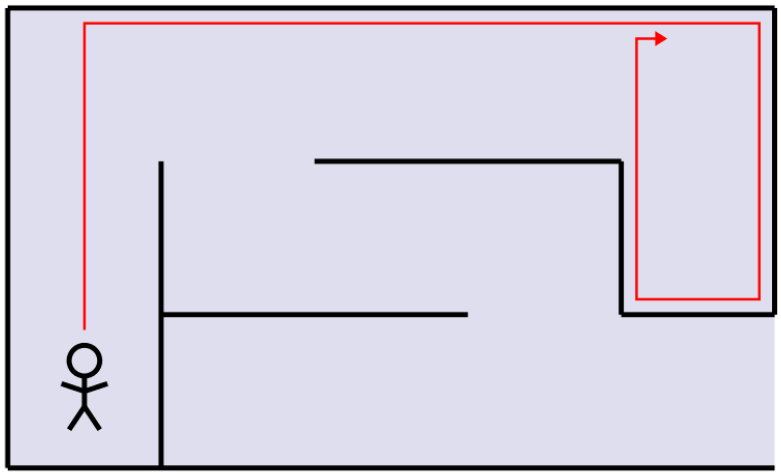
Problem: Inseln werden endlos umkreist

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

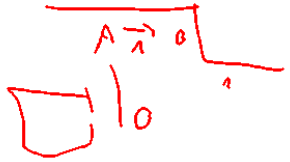


Problem: Jetzt laufen wir im ersten Beispiel im Kreis

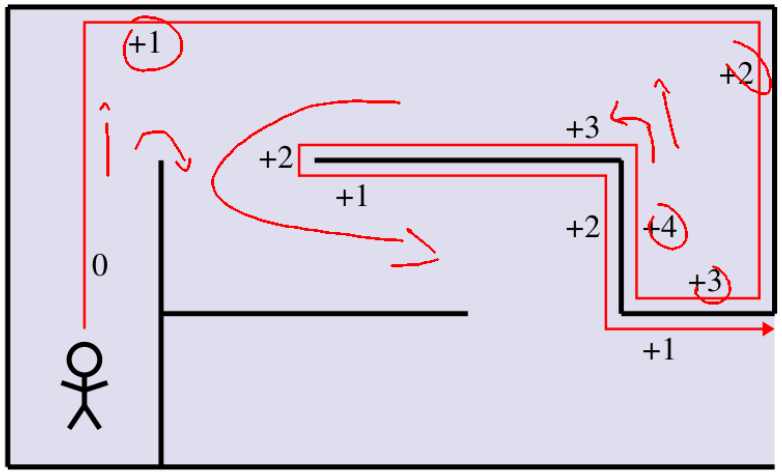
Pledge-Algorithmus

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*;

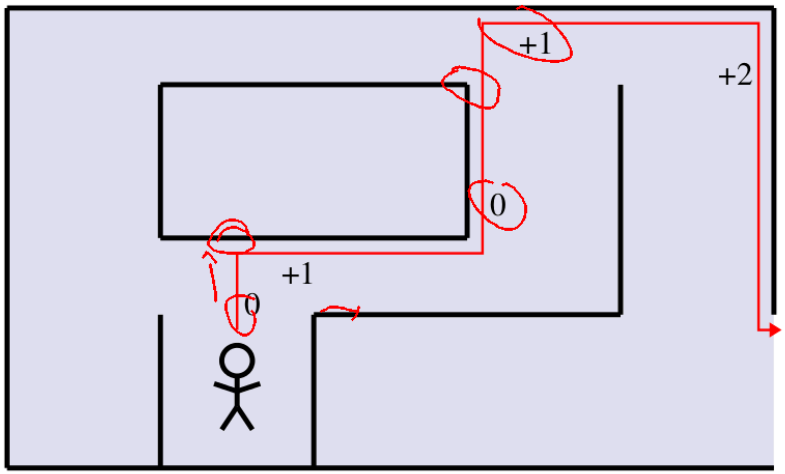


Weg aus dem Labyrinth

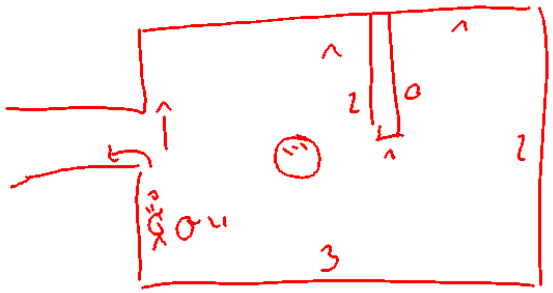


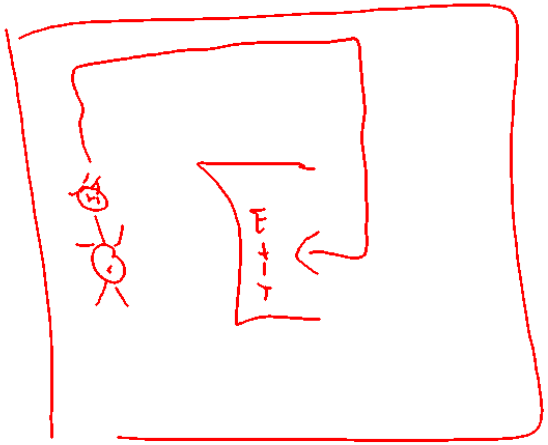
1. Beispiel funktioniert

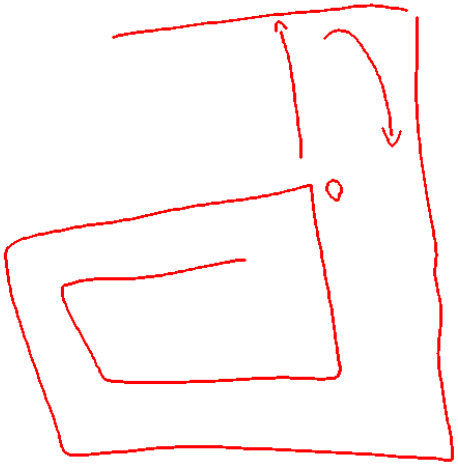
Weg aus dem Labyrinth



2. Beispiel funktioniert auch

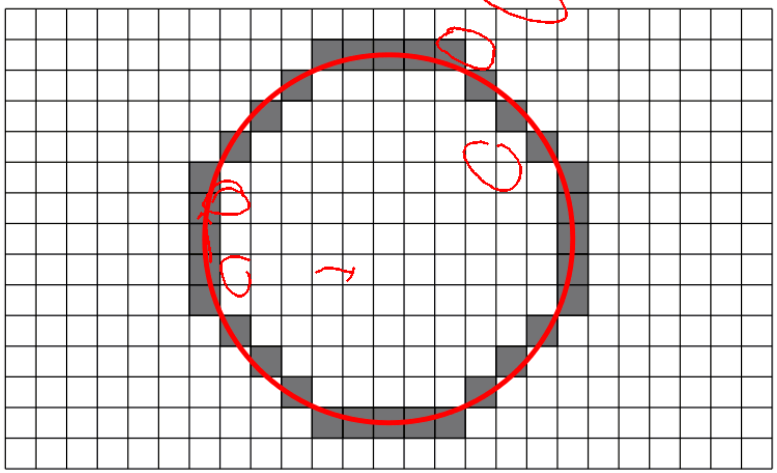






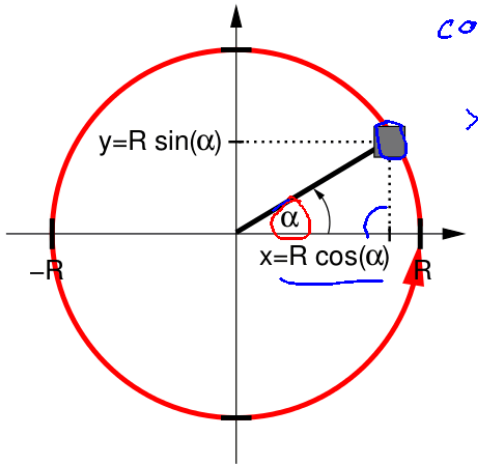
Kreis zeichnen

Wie kann ein Computer einen Kreis zeichnen?



Kreis zeichnen: mit Winkelfunktionen

Naiver Ansatz: eine Runde wie mit dem Zirkel



$\cos(\alpha) = \frac{x}{R}$

$x = R \cdot \cos(\alpha)$

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

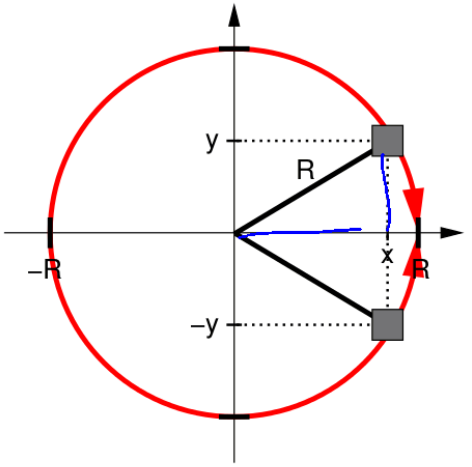
for $i = 0; i < n; i{+}{+}$ **do**
  plot($R * \cos(2\pi * i/n)$, $R * \sin(2\pi * i/n)$);

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

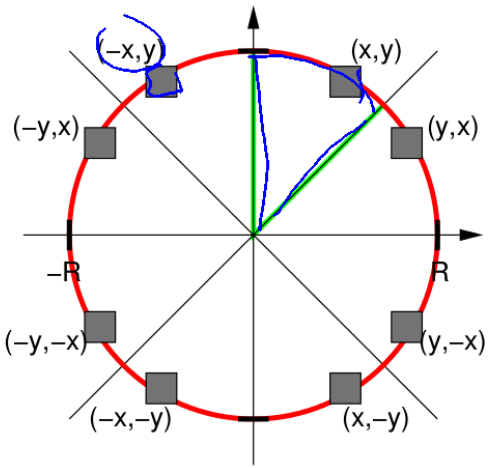
Eingabe : Radius R

for $x = -R; x \leq R; x{+}{+}$ **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

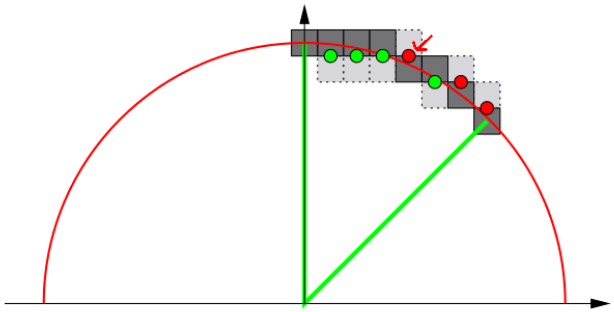
Algorithmus Bresenham1: zeichnet Kreis mit Radius R

$x = 0; \quad y = R;$
$\text{plot}(0, R); \quad \text{plot}(R, 0); \quad \text{plot}(0, -R); \quad \text{plot}(-R, 0);$
$F = \frac{5}{4} - R;$
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

- 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: $x++; \quad y--$



Kreis zeichnen: mit Multiplikation

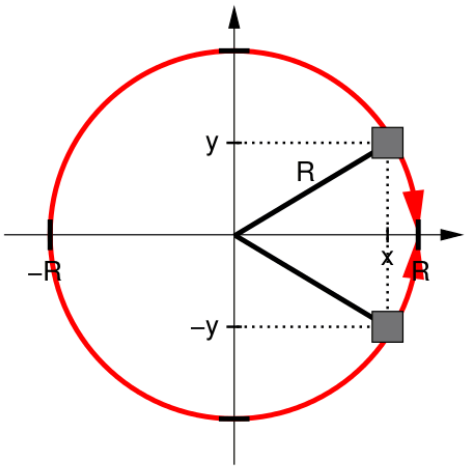
- Test, ob (x, y) innerhalb des Kreises:

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

- 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 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 Multiplikation

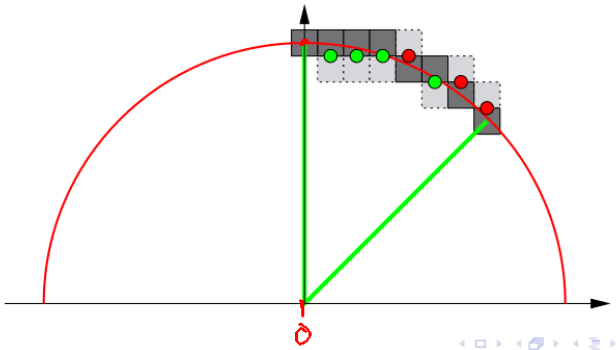
- Test, ob (x, y) innerhalb des Kreises:

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

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

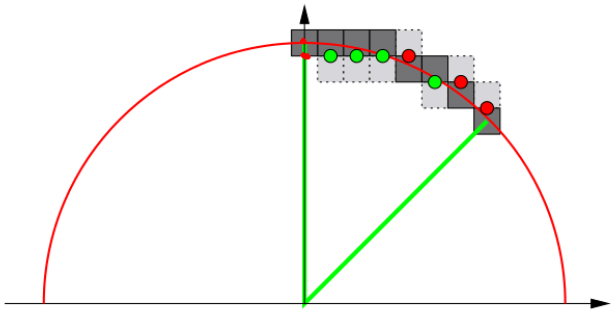
Kreis zeichnen: mit Multiplikation

- 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

- 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

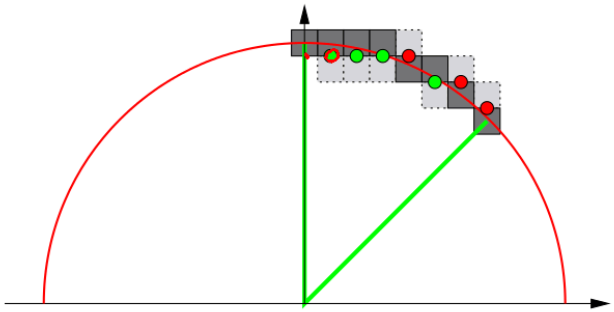
- Test, ob (x, y) innerhalb des Kreises:

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

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

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

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

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

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

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 Addition / Subtraktion

- 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:

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

- Updates nach rechts (E) und nach unten ~~rechts~~ (SE):

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

Kreis zeichnen: mit Addition / Subtraktion

- Der Bruch $\frac{5}{4}$ kann durch 1 ersetzt werden, weil sich F immer um eine ganze Zahl 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: mit Addition / Subtraktion

Algorithmus Bresenham2: zeichnet Kreis mit Radius R

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

Bresenham-Algorithmus

- Ab Anfang der 1960er Jahre hat Jack Bresenham Algorithmen zur Linien- und Kreisdarstellung entwickelt.
- Diese verwenden nur einfache Additionen ganzer Zahlen.
- Sie sind damit deutlich schneller als die naiven Ansätze.

Einführung Beispiele

Bresenham-Algorithmus

- Ab Anfang der 1960er Jahre hat Jack Bresenham Algorithmen zur Linien- und Kreisdarstellung entwickelt.
- Diese verwenden nur einfache Additionen ganzer Zahlen.
- Sie sind damit deutlich schneller als die naiven Ansätze.

H. Seidl (TUM) GAD SS'15 38

Multiplikation langer Zahlen

Schulmethode:

- gegeben Zahlen a und b
- multipliziere a mit jeder Ziffer von b
- addiere die Teilprodukte

$$
\begin{array}{cccccccc}
5 & 6 & 7 & 8 & \cdot & 4 & 3 & 2 & 1 \\
\hline
 & 2 & 2 & 7 & 1 & 2 & & & \\
 & & 1 & 7 & 0 & 3 & 4 & & \\
 & & & 1 & 1 & 3 & 5 & 6 & \\
 & & & & & 5 & 6 & 7 & 8 \\
\hline
 & 2 & 4 & 5 & 3 & 4 & 6 & 3 & 8 \\
\hline\hline
\end{array}
$$