Script generated by TTT

Title: Täubig: GAD (08.04.2014)

Date: Tue Apr 08 13:50:47 CEST 2014

Duration: 116:16 min

Pages: 51

Grundlagen: Algorithmen und Datenstrukturen

Hanjo Täubig

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

Sommersemester 2014

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://www14.in.tum.de/lehre/2014SS/gad/
- Voraussetzung: IN0001 – Einführung in die Informatik 1
  Empfehlung: IN0015 – Diskrete Strukturen
- Klausur:
  Gesamtklausur: Samstag, 19.07.2014 (11:30–14:00 Uhr)
  Wiederholungsklausur: Freitag, 26.09.2014 (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

- planmäßig im 2. Fachsemester

Dozent / Kontaktdaten

- Hanjo Täubig

  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 (3. Stock, Finger 9)
- Sprechstunde: Mittwoch 13-14 Uhr
  (oder nach Vereinbarung)

Übung

- 2 SWS Tutorübungen
- 28 Gruppen an 12 verschiedenen Terminen
- jeweils maximal 16-20 Teilnehmer
- Anmeldung über TUMonline:
  https://campus.tum.de/
- Übungsleitung:

  Jeremias Weihmann (weihmann@in.tum.de)
- Webseite:

  http://www14.in.tum.de/lehre/2014SS/gad/uebung/

Thematische Einordnung

- Einführung in die Informatik 1
  - grundlegende Begriffe und Konzepte
- Diskrete Strukturen
  - Kombinatorik, Graphen

Thematische Einordnung

- Einführung in die Informatik 1
  - grundlegende Begriffe und Konzepte
- Diskrete Strukturen
  - Kombinatorik, Graphen
- Grundlagen: Datenbanken
  - Indexstrukturen
- Diskrete Wahrscheinlichkeitstheorie
  - Erwartungswert, bedingte Wahrscheinlichkeit
- Einführung in die Theoretische Informatik
  - $\mathcal{NP}$-vollständige Probleme

Thematische Einordnung

- Einführung in die Informatik 1
  - grundlegende Begriffe und Konzepte
- Diskrete Strukturen
  - Kombinatorik, Graphen
- Grundlagen: Datenbanken
  - Indexstrukturen
- Diskrete Wahrscheinlichkeitstheorie
  - Erwartungswert, bedingte Wahrscheinlichkeit
- Einführung in die Theoretische Informatik
  - $\mathcal{NP}$-vollständige Probleme
- weiterführende Vorlesungen
  - z.B. Effiziente Algorithmen 1+2, Komplexitätstheorie, Online- und Approximationsalgorithmen, Fortgeschrittene Netzwerk- und Graphalgorithmen, Algorithmische Bioinformatik

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:

  SS'08: Prof. Dr. Christian Scheideler
  SS'09: Prof. Dr. Helmut Seidl

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 - Definition

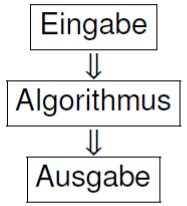
Definition

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

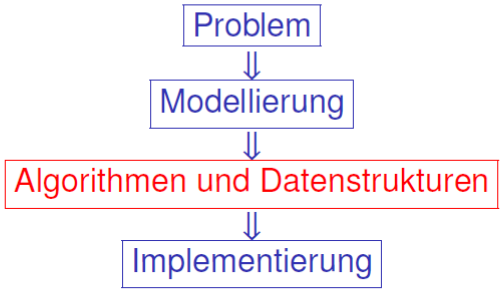
Beispiel: `PriorityQueue` mit Operationen `insert` und `deleteMin`

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:

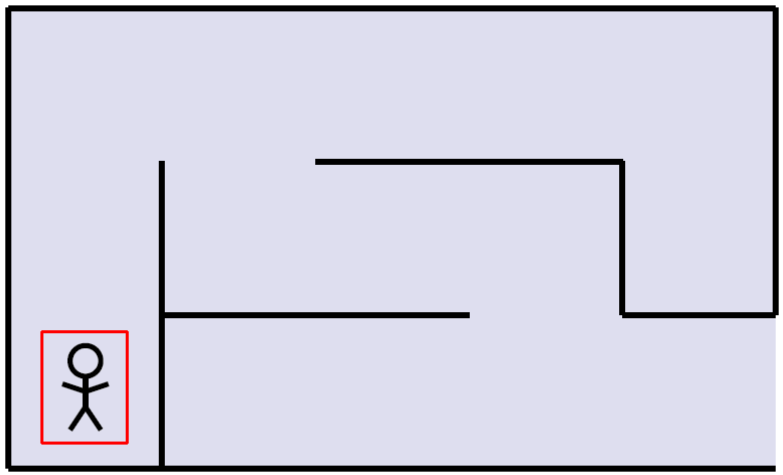
- Riesige Datenmengen (Bioinformatik)
- Echtzeitanwendungen (Spiele, Flugzeugsteuerung)

Ziel der Vorlesung:

Grundstock an effizienten Algorithmen und Datenstrukturen für Standardprobleme

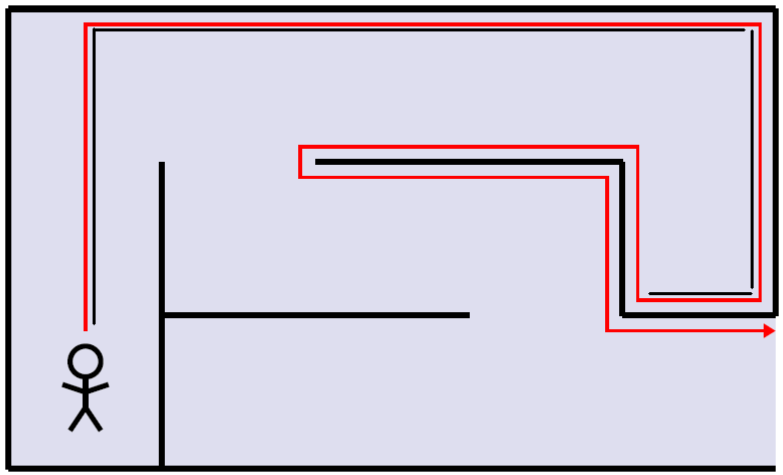
Übersicht

Weg aus dem Labyrinth



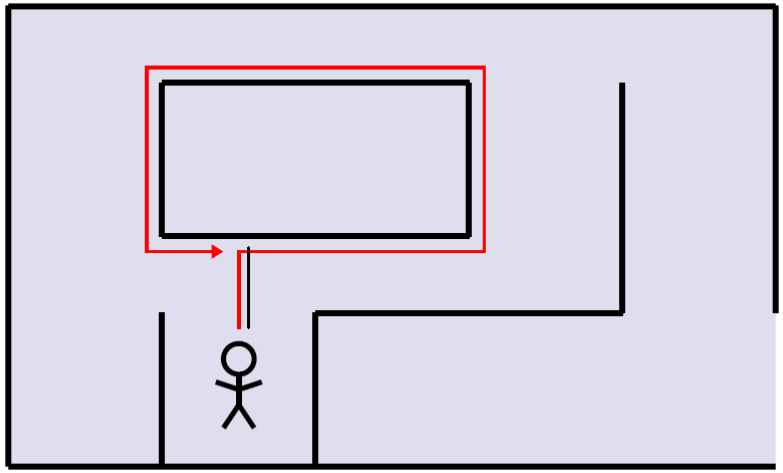
Problem: Es ist dunkel!

Weg aus dem Labyrinth



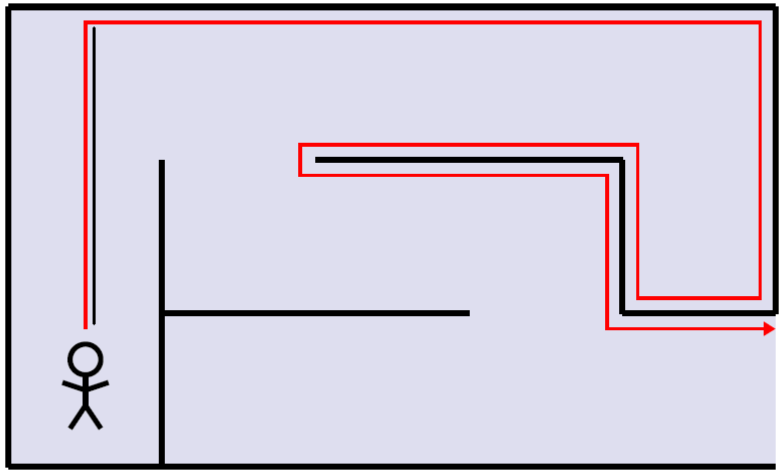
1. Versuch: mit einer Hand immer an der Wand lang

Weg aus dem Labyrinth



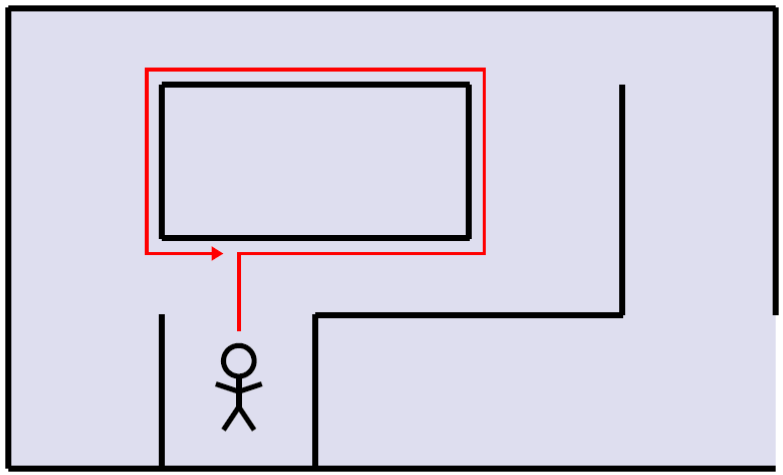
Problem: Inseln werden endlos umkreist

Weg aus dem Labyrinth



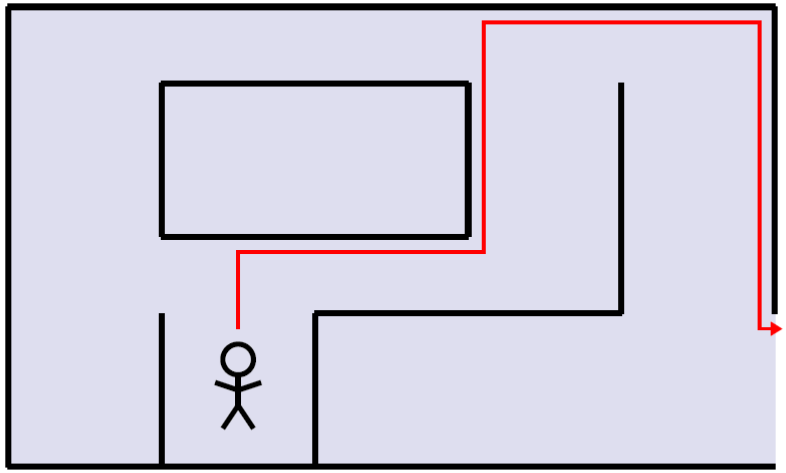
1. Versuch: mit einer Hand immer an der Wand lang

Weg aus dem Labyrinth



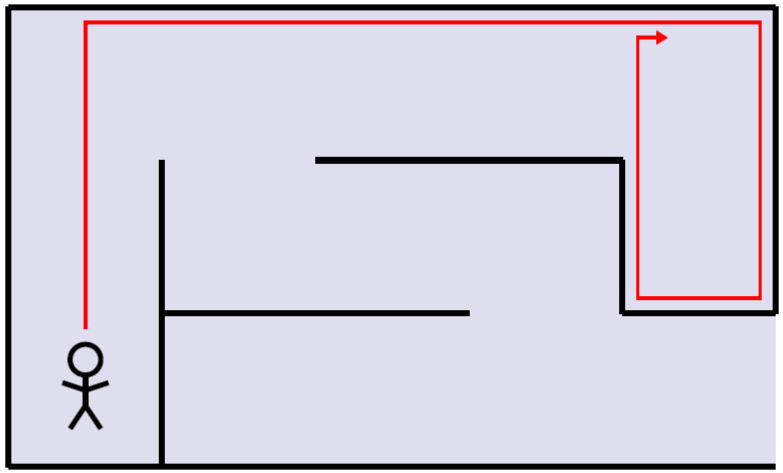
Problem: Inseln werden endlos umkreist

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



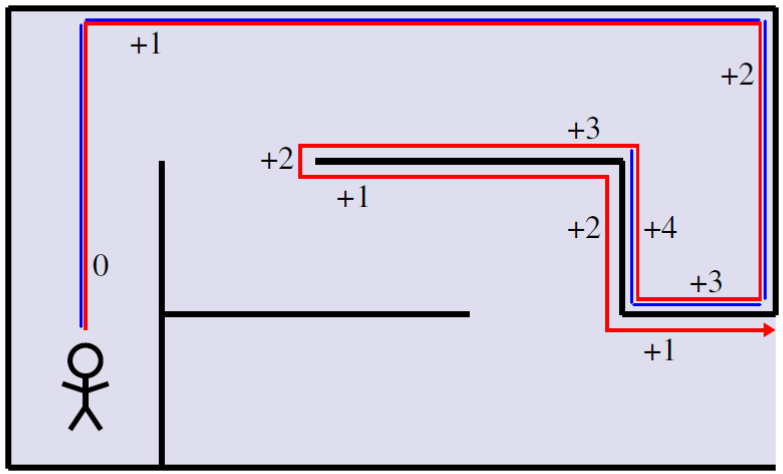
Problem: Jetzt laufen wir im ersten Beispiel im Kreis

Pledge-Algorithmus

Algorithmus Labyrinth: findet einen Ausgang

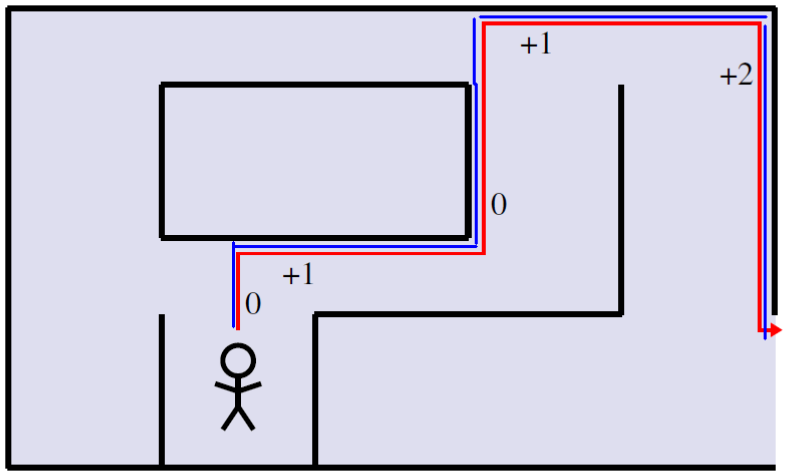
```
Setze Umdrehungszähler auf 0;
repeat
    repeat
        Gehe geradeaus;
    until Wand erreicht;
    Drehe nach rechts;
    Inkrementiere Umdrehungszähler;
    repeat
        Folge dem Hindernis mit einer Hand;
        dabei: je nach Drehrichtung Umdrehungszähler
        inkrementieren / dekrementieren;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


Weg aus dem Labyrinth



1. Beispiel funktioniert

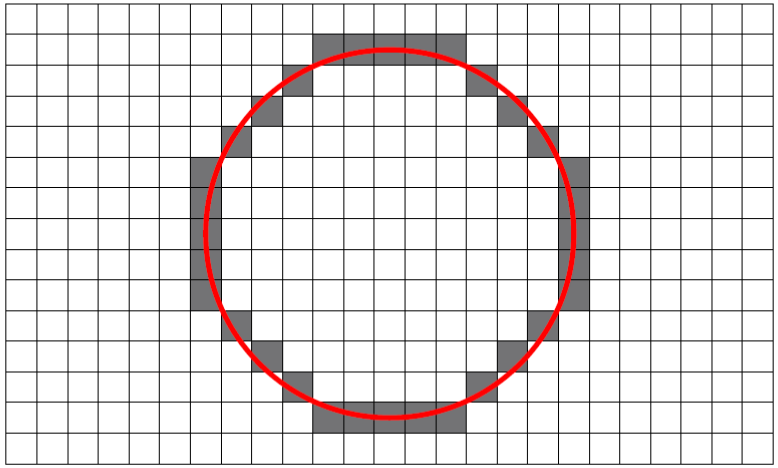
Weg aus dem Labyrinth



2. Beispiel funktioniert auch

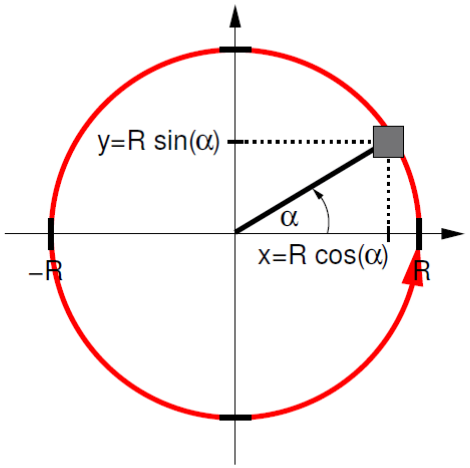
Kreis zeichnen

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

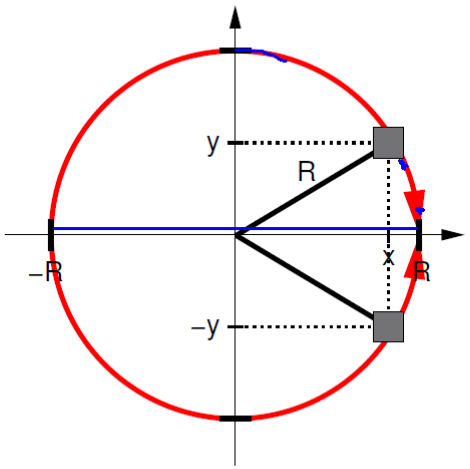
for $i = 0; i < n; i{+}{+}$ **do**
  plot($R * \cos(2\pi * i/n), R * \sin(2\pi * i/n)$);

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

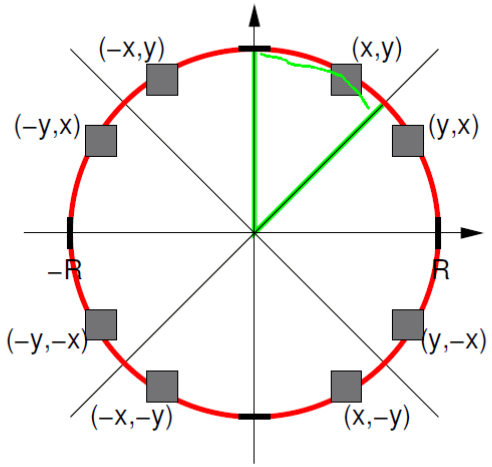
Eingabe : Radius R

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

Problem: sqrt() ist auch noch relativ teuer!

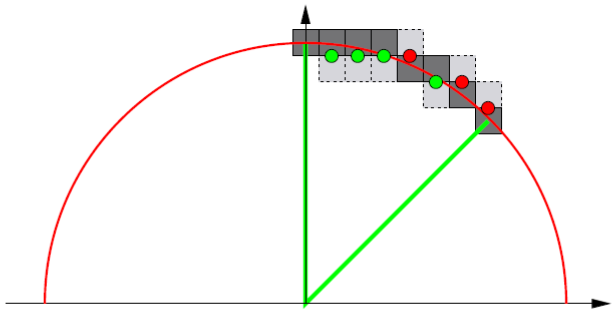
Kreis zeichnen: mit Multiplikation

Besserer Ansatz: Ausnutzung von Spiegelachsen



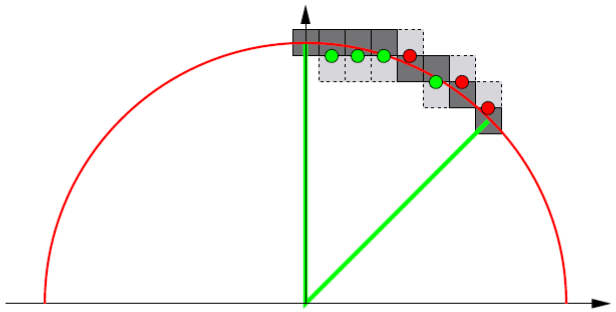
Kreis zeichnen: mit Multiplikation

- 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

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

- Ersetzung der Korrekturterme für F:

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

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

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

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

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

Kreis zeichnen: 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

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

Multiplikation langer Zahlen

Schulmethode:

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

5	6	7	8	·	4	3	2	1
	2	2	7	1	2			
		1	7	0	3	4		
			1	1	3	5	6	
					5	6	7	8
	2	4	5	3	4	6	3	8

Aufwand

- Wenn die Zahlen klein sind, ist der Aufwand ok.
- Aber wenn die Zahlen sehr lang sind, kann man das Produkt dann schneller ausrechnen als mit der Schulmethode?

$\Rightarrow$ Wie wollen wir die Zeit oder den Aufwand überhaupt messen?

- Am besten nicht in Sekunden, die irgendein Rechner braucht, denn das könnte für einen anderen Rechner eine ganz andere Zahl sein.
- Außerdem werden die Computer ja von Generation zu Generation immer schneller und leistungsfähiger.

$\Rightarrow$ Wir zählen Grundoperationen: Operationen, die man in einem einzigen Schritt bzw. in einer konstanten Zeiteinheit ausführen kann.

Grundoperation

- Multiplikation von zwei Ziffern: $x \cdot y = ?$

  Das Ergebnis besteht aus (höchstens) zwei Ziffern u (Zehnerstelle) und v (Einerstelle), also

  $$x \cdot y = 10 \cdot u + v$$

- Addition von drei Ziffern: $x + y + z = ?$

  Auch hier besteht das Ergebnis aus (höchstens) zwei Ziffern u (Zehnerstelle) und v (Einerstelle), also

  $$x + y + z = 10 \cdot u + v$$

  Wir benutzen hier drei Ziffern als Summanden, weil wir später Überträge berücksichtigen wollen.

Analyse der Addition

- *Zahl plus Zahl:*

	6	9	1	7
	4	2	6	9
1	1	0	1	
1	1	1	8	6

Zur Addition zweier Zahlen mit jeweils n Ziffern brauchen wir n Additionen von 3 Ziffern, also n Grundoperationen.

Ergebnis: Zahl mit $n + 1$ Ziffern

Analyse des Teilprodukts

- *Zahl mal Ziffer:*

	5	6	7	8	·	4
			3	2		
		2	8			
	2	4				
2	0					
2	2	7	1	2		

Zur Multiplikation einer Zahl bestehend aus n Ziffern mit einer einzelnen Ziffer brauchen wir

- n Multiplikationen von 2 Ziffern und
- $n + 1$ Additionen von 3 Ziffern, wobei in der letzten Spalte eigentlich nichts addiert werden muss,

also $2n[+1]$ Grundoperationen.

Ergebnis: Zahl mit $n + 1$ Ziffern

Analyse des Produkts

- *Zahl mal Zahl:*

5	6	7	8	·	4	3	2	1
	2	2	7	1	2	0	0	0
		1	7	0	3	4	0	0
			1	1	3	5	6	0
					5	6	7	8
	2	4	5	3	4	6	3	8

Zur Multiplikation zweier Zahlen mit jeweils n Ziffern brauchen wir

- n Multiplikationen einer n-Ziffern-Zahl mit einer Ziffer, also $n \cdot (2n[+1]) = 2n^2[+n]$ Grundoperationen
- Zwischenergebnisse sind nicht länger als das Endergebnis ($2n$ Ziffern), also $n-1$ Summen von Zahlen mit $2n$ Ziffern, also $(n-1) \cdot 2n = 2n^2 - 2n$ Grundoperationen

Insgesamt: $4n^2 - [2]n$ Grundoperationen

Analyse des Produkts

- *Zahl mal Zahl:*

5	6	7	8	·	4	3	2	1
	2	2	7	1	2			
		1	7	0	3	4		
			1	1	3	5	6	
					5	6	7	8
	2	4	5	3	4	6	3	8

Genauer:

- Beim Aufsummieren der Zwischenergebnisse muss man eigentlich jeweils nur Zahlen bestehend aus $n+1$ Ziffern addieren. Das ergibt $(n-1)(n+1) = n^2 - 1$ Grundoperationen.

Insgesamt hätte man damit $3n^2[+n] - 1$ Grundoperationen.

Geht es besser?

Frage:

- Ist das überhaupt gut?
- Vielleicht geht es ja schneller?
- Was wäre denn überhaupt eine signifikante Verbesserung?
- Vielleicht irgendetwas mit $2n^2$?
- Das würde die Zeit auf ca. 2/3 des ursprünglichen Werts senken.
- Aber bei einer Verdoppelung der Zahlenlänge hätte man immer noch eine Vervierfachung der Laufzeit.

- Wir werden diese Frage später beantworten ...

Algorithmen-Beispiele

- Rolf Klein und Tom Kamphans:
  Der Pledge-Algorithmus: Wie man im Dunkeln aus einem Labyrinth entkommt
- Dominik Sibbing und Leif Kobbelt:
  Kreise zeichnen mit Turbo
- Arno Eigenwillig und Kurt Mehlhorn:
  Multiplikation langer Zahlen (schneller als in der Schule)
- Diese und weitere Beispiele:

Taschenbuch der Algorithmen (Springer, 2008)