Algorithmen für Kettenbrüche

Def.: Ein allgemeiner (unendlicher) Kettenbruch k ist ein fortgesetzter Bruch mit

$$k = z_0 + \cfrac{a_1}{z_1 + \cfrac{a_2}{z_2 + \cfrac{a_3}{z_3 + \cfrac{a_4}{z_4 + \cfrac{a_5}{\dots}}}}}$$

$z_0 \in \mathbb{Z}$ und $a_i, z_i \in \mathbb{N}^*$ für $i \in \mathbb{N}^*$.

Kurzschreibweise: $k = z_0 + \frac{a_1|}{|z_1} + \frac{a_2|}{|z_2} + \frac{a_3|}{|z_3} + \dots$

In der Praxis (z.B. Zahlentheorie) spielen reguläre Kettenbrüche eine besondere Rolle:

$$x = z_0 + \cfrac{1}{z_1 + \cfrac{1}{z_2 + \dots + \cfrac{1}{z_n}}}$$ (mit $x \in \mathbb{Q}$) heißt endlicher regulärer Kettenbruch .

Kurzschreibweise: $x = [\, z_0;\ z_1, z_2, \dots, z_n \,]$; $z_0 \in \mathbb{Z}$ heißt Anfangsglied

$z_i \in \mathbb{N}^*$ für $i = 1 \dots n$, wobei $z_n \neq 1$, da $\frac{1}{1}$ am Schluss kein echter Teilbruch wäre !

Das Wort "regulär" steht dafür, dass alle Zähler des KBs 1 sind.

Für einen nicht abbrechenden regulären KB (z.B. für π) gilt: $x = [\, z_0;\ z_1, z_2, z_3, \dots \,]$.

Rationale Zahlen ($x \in \mathbb{Q}$) besitzen eine endliche KB-Entwicklung, z.B. $x = 4{,}5$.

Irrationale Zahlen ($x \in \mathbb{R}$) besitzen ein unendliche KB-Entwicklung, z.B. $x = \pi$.

Die folgenden Algorithmen gelten für **positive** reguläre Kettenbrüche !

Wie wandelt man einen Bruch in einen Kettenbruch um (KB-Erzeugung) ?

Beispiel: 67 / 29 soll in einen KB umgewandelt werden:

67 / 29 = 2+9/29 = 2+1/(29/9) = 2+1/(3+2/9) = 2+1/(3+1/(9/2)) = 2+1/(3+1/(4+1/2))
= **2** + 1 / (**3** + 1 / (**4** + 1 / **2**))

Kurzschreibweise: 67 / 29 = [2; 3,4,2]

Algorithmisch liegt eine sukzessive Division mit Restbetrachtung vor (Euklidscher Algorithmus) :

```
Eingabe (zähler, nenner)
Solange  nenner > 0 wiederhole
    Ausgabe (zähler div nenner)
    Setze zähler = zähler mod nenner
    Vertausche zähler mit nenner
Ende Solange
```

Rechnung für obiges Beispiel (zae=zähler ; nen=nenner) :

Gegeben sind zae=67 nen=29
zae div nen = 67 div 29 = **2**
zae = 67 mod 29 = 9 vertauschen: zae = 29 nen = 9
zae div nen = 29 div 9 = **3**
zae = 29 mod 9 = 2 vertauschen: zae = 9 nen = 2
zae div nen = 9 div 2 = **4**
zae = 9 mod 2 = 1 vertauschen: zae = 2 nen = 1
zae div nen = 2 div 1 = **2**
zae = 2 mod 1 = 0 vertauschen: zae = 2 **nen = 0** **Abbruch !**

Die ermittelten Kettenbruchelemente sind: 2 3 4 2

Rechnung für das Beispiel 187 / 58 :

Gegeben sind zae=187 nen=58
zae div nen = 187 div 58 = **3**
zae = 187 mod 58 = 13 vertauschen: zae = 58 nen = 13
zae div nen = 58 div 13 = **4**
zae = 58 mod 13 = 6 vertauschen: zae = 13 nen = 6
zae div nen = 13 div 6 = **2**
zae = 13 mod 6 = 1 vertauschen: zae = 6 nen = 1
zae div nen = 6 div 1 = **6**
zae = 6 mod 1 = 0 vertauschen: zae = 1 **nen = 0** **Abbruch !**

Die ermittelten Kettenbruchelemente sind: 3 4 2 6

Rechnung für das Beispiel 11111111 / 86419753 :

Gegeben sind zae=11111111 nen= 86419753
zae div nen = 11111111 div 86419753 = **0**
zae = 11111111 mod 86419753 = 11111111 vertauschen: zae = 86419753 nen = 11111111
zae div nen = 86419753 div 11111111 = **7**
zae = 86419753 mod 11111111 = 8641976 vertauschen: zae = 11111111 nen = 8641976
zae div nen = 11111111 div 8641976 = **1**
zae = 11111111 mod 8641976 = 2469135 vertauschen: zae = 8641976 nen = 2469135
zae div nen = 8641976 div 2469135 = **3**
zae = 8641976 mod 2469135 = 1234571 vertauschen: zae = 2469135 nen = 1234571
zae div nen = 2469135 div 1234571 = **1**
zae = 2469135 mod 1234571 = 1234564 vertauschen: zae = 1234571 nen = 1234564

zae div nen = 1234571 div 1234564 = **1**
zae = 1234571 mod 1234564 = 7 vertauschen: zae = 1234564 nen = 7

zae div nen = 1234564 div 7 = **176366**
zae = 1234564 mod 7 = 2 vertauschen: zae = 7 nen = 2

zae div nen = 7 div 2 = **3**
zae = 7 mod 2 = 1 vertauschen: zae = 2 nen = 1

zae div nen = 2 div 1 = **2**
zae = 2 mod 1 = 0 vertauschen: zae = 1 **nen = 0** **Abbruch !**

Die ermittelten Kettenbruchelemente sind: 0 7 1 3 1 1 176366 3 2

Umkehrung: Wie wandelt man einen Kettenbruch in einen Bruch um ?

Dies geschieht durch Abarbeitung des Kettenbruchterms von rechts nach links.

```
Für i von 0 bis n wiederhole
    Eingabe(zi)              // zi = Kettenbruchelemente
x = 0                        // x ist der gesuchte Bruch
Für i von n ab bis 1 wiederhole
    x = 1/(x + zi)
x = x + z0
```

Rechnung für obiges (erstes) Beispiel (exakte Bruchrechnung) :

KB = [2; 3,4,2]
Eingabe: z0 = 2 z1 = 3 z2 = 4 z3 = 2
x = 0
x = 1 / (0 + 2) = 1/2
x = 1 / (1/2 + 4) = 1 / (9/2) = 2/9
x = 1 / (2/9 + 3) = 1 / (29/9) = 9/29
x = 9 / 29 + 2 = **67 / 29**

Weiteres Beispiel (exakte Bruchrechnung) :

KB = [1; 2,3,4,5,6,7,8,9,10]

Eingabe: z0 = 1 z1 = 2 z2 = 3 z3 = 4 z4 = 5 z5 = 6 z6 = 7 z7 = 8 z8 = 9 z9 = 10
x = 0
x = 1 / (0 + 10) = 1/10
x = 1 / (1/10 + 9) = 1 / (91/10) = 10/91
x = 1 / (10/91 + 8) = 1 / (738/91) = 91/738
x = 1 / (91/738 + 7) = 1 / (5257/738) = 738/5257
x = 1 / (738/5257 + 6) = 1 / (32280/5257) = 5257/32280
x = 1 / (5257/32280 + 5) = 1 / (166657/32280) = 32280/166657
x = 1 / (32280/166657 + 4) = 1 / (698908/166657) = 166657/698908
x = 1 / (166657/698908 + 3) = 1 / (2263381/698908) = 698908/ 2263381
x = 1 / (698908/2263381 + 2) = 1 / (5225670/2263381) = 2263381/5225670
x = 2263381/5225670 + 1 = **7489051 / 5225670**

Anmerkung: Für einen allgemeinen (endlichen) Kettenbruch lautet der Algorithmus „KB zu Bruch“:

```
Für i von 0 bis n wiederhole
    Eingabe(zi, ai)          // zi, ai = Kettenbruchelemente
x = 0                        // x ist der gesuchte Bruch
Für i von n ab bis 1 wiederhole
    x = ai/(x + zi)
x = x + z0
```

Beispiel: $x = 0 + \frac{1|}{|4} + \frac{2|}{|5} + \frac{3|}{|8}$

x = 3/8 → x = 2/(3/8 + 5) = 16/43 → **x** = 1/(16/43 + 4) = **43/188**

Wie wandelt man eine Dezimalzahl in einen Kettenbruch um ?

Dies geschieht durch Abspaltung von ganzzahligen Anteilen der Teilbrüche des Kettenbruchs.
Der Kettenbruch wird auf diese Weise von links nach rechts berechnet.
Zu beachten ist, dass die am weitesten rechts stehende Zahl (z_n) nicht 1 sein darf !
Mit dieser Methode lassen sich auch **Bruchnäherungen reeller Zahlen** bilden .

Achtung: Hier kann auch ein unendlicher Kettenbruch entstehen !
Deshalb sollte man eine Grenze nmax für die Anzahl der Stellen angeben !

```
Eingabe(x, nmax)       // x ist die gegebene Dezimalzahl
n=1
Solange (x <> int(x)) und (n <= nmax) wiederhole
   Ausgabe(int(x))
   fracx = x - int(x)
   Falls fracx > 0
      dann x = 1 / fracx
   n = n+1
Ende Solange
Falls letzte Zahl des KBs = 1, dann addiere sie zur vorletzten Zahl
```

Achtung: Durch die fehlerbehaftete Gleitkommaarithmetik kann es zu groben Rundungsfehlern kommen, insbesondere bei der Division 1 / fracx . Daher wird bereits nach wenigen Schleifendurchläufen mit Sicherheit ein falsches Kettenbruchelement ermittelt werden ! (vgl. Beispiele unten)

Wegen der Gleikommaarithmetik ist auch ein Test auf Gleichheit bzw. Ungleichheit wie bei fracx > 0 sinnlos. Man muss hier auf abs(**fracx) > eps** (mit z.B. eps = 5E-16) testen .

Beispiele:

(Exakte) Rechnung für x = 9,7:

int(9,7) = **9**
fracx = 9,7-9 = 0,7 <> 0 daher x = 1/0,7 = 10/7 = 1,42857...
int(10/7) = **1**
fracx = 10/7-1 = 3/7 = 0,428571428571... (Periode) <> 0 daher x = 1/(3/7) = 7/3 = 2,333...
int(7/3) = **2**
fracx = 7/3-2 = 1/3 = 0,3333... (Periode) <> 0 daher x = 1/(1/3) = 3
int(3) = **3**
fracx = 3-3 = 0 Abbruch !
Der Kettenbruch lautet daher [9;1,2,3]. Als Bruch 9+1/(1+1/(2+1/3)) = 97/10

(Exakte) Rechnung für x = 0,18:

int(0,18) = **0**
fracx = 0,18-0 = 0,18 <> 0 daher x = 1/0,18 = 5,55555555.. (eine Periode !)
int(5,555..) = **5**
fracx = 5,555...-5 = 0,555... daher x = 1/0,555... = 1,8
int (1,8) = **1**
fracx = 1,8-1 = 0,8 daher x = 1/0,8 = 1,25
int (1,25) = **1**
fracx = 1,25-1 = 0,25 daher x = 1/0,25 = 4
int (4) = **4**
fracx = 4-4 = 0 Abbruch !

Kettenbruch x = [0;5,1,1,4]. Als Bruch 1/(5+1/(1+1/(1+1/4))) = 9/50

Wegen der oben erläuterten Rundungsfehler ist eine andere Methode besser geeignet:

Die Dezimalzahl in einen **Bruch** umwandeln und diesen dann (s.o.) in einen KB überführen!

```
Eingabe(x)          // x ist die gegebene Dezimalzahl
x in Bruch z umwandeln mit Zähler zae und Nenner nen  ( siehe unten)
Solange  nen > 0 wiederhole
    Ausgabe(zae div nen)
    Setze zae = zae mod nen
    Vertausche zae mit nen
Ende Solange
```

x in Bruch z umwandeln:

- **Anzahl n der Nachkommastellen von x ermitteln**
- **zae = x ohne Komma**
- **nen = 10^n**

Java - Methode dazu:

```java
void xZuBruch(double x) {
  String s = String.valueOf(x);
  int pos = s.indexOf('.');
  String sNk = s.substring(pos + 1);
  int nkStellen = sNk.length();
  long zae = Long.parseLong(s.substring(0, pos) + sNk);
  long nen = 1;
  for (int i = 1; i <= nkStellen; i++) nen = 10 * nen;
}
```

Anmerkung: In der Programmiersprache Java sind bei Verwendung des Datentyps Long (max 9223372036854775807L) für Zähler und Nenner jeweils bis zu 19-stellige Zahlen (10^{18}) möglich.

Beispiel: x = 3,141592653589793 (Näherung für Pi ; 15 Dezimalen)

Die Umwandlung in einen Bruch (in Java: Datentyp long) liefert :
zae = 3141592653589793
nen = 10^{15} = 1000000000000000

Umwandlung des Bruches in einen Kettenbruch :
zae div nen = 3141592653589793 div 1000000000000000 = **3**
zae = 3141592653589793 mod 1000000000000000 = 141592653589793
zae div nen = 1000000000000000 div 141592653589793 = **7**
zae = 1000000000000000 mod 141592653589793 = 8851424871449
zae div nen = 141592653589793 div 8851424871449 = **15**
zae = 141592653589793 mod 8851424871449 = 8821280518058
zae div nen = 8851424871449 div 8821280518058 = **1**

usw.

Es entsteht schließlich der Kettenbruch für die Pi-Approximation 3,141592653589793 (15 Dezimalen) : [3; 7,15,1,292,1,1,1,2,1,3,1,14,**4**,2,3,1,12,5,1,5,20,1,11,1,1,1,2] .

Zum Vergleich (exakte Zahlen für den Kettenbruch von Pi):
[3; 7,15,1,292,1,1,1,2,1,3,1,14,**2**,1,1,2,2,2,2,1,84,2,1,1,15,3, ...]

Die Zahl 4 ist somit das erste falsche Element des für Pi ermittelten KBs, was vor allem auf die Beschränkung auf 15 Dezimalen zurückzuführen ist; demnach sind auch die restlichen Elemente des KBs falsch !

KB-Anwendung: Konstruktion von Zahnrädern (**Christaan Huygens**)

1682 baute **Christaan Huygens** (berühmter niederländischer Astronom / Physiker ; 1629-1695) ein Planetarium, in dem das Verhältnis der Umlaufzeiten von Saturn und Erde um die Sonne dargestellt werden sollte.

Huygens fand für die Umlaufzeiten von Saturn und Erde das Verhältnis
77708431 : 2640858 ≈ 29,42544847 .

Für eine passende Astronomische Uhr mussten die Zahnräder mit wesentlich weniger Zähnen als die hier genannten Zahlen auskommen, weil kleine Zahnräder mit einer hohen Zähnezahl technisch schwer zu realisieren sind. Deswegen suchte Huygens für dieses Verhältnis optimale Näherungen mithilfe von Kettenbrüchen.

Gesucht waren also möglichst kleine natürliche Zahlen a und b mit a / b ≈ 29,43 .

Mithilfe einer Kettenbruchentwicklung für 77708431 / 2640858 gelang Huygens die folgende Näherung:

77708431 / 2640858 = 29 + 1123549 / 2640858 = 29 + 1 / (2640858 / 1123549) =
29 + 1 / (2 + 393760 / 1123549)

Vernachlässigt man hier den Bruch 393760 / 1123549, so erhält man als Näherung
29 + 1 / 2 = 59 / 2.
Der Fehler bezüglich 77708431 / 2640858 beträgt 0,25% , ist also noch zu groß.

Weitere Zerlegung:
29 + 1 / (2 + 393760 / 1123549) = 29 + 1 / (2 + 1 / (1123549 / 393760)) =
29 + 1 / (2 + 1 / (2 + 336029 / 393760))

Die Näherung ist dann 29 + 1 / (2 + 1 / 2) = 29 + 1 / (5/2) = 29 + 2/5 = 147 / 5 .
Der Fehler bezüglich 77708431 / 2640858 beträgt 0,086% . Dies war Huygens immer noch zu viel.

Daher ersetzte er den Restbruch 336029 / 393760 durch 1 und erhielt als Näherung:
29 + 1 / (2 + 1 / (2 + 1)) = 29 + 1 / (2 + 1 / 3) = 29 + 1 / (7 / 3) = 29 + 3 / 7 = **206 / 7** .

Mit dem **Fehler von 0,01%** bezüglich 77708431 / 2640858 war Huygens dann zufrieden.

Er verwendete das Zahnradverhältnis 206 : 7 !

KB-Anwendung: Näherungsbrüche für reelle Zahlen finden :

Wenn man beim Taschenrechner TI84 45*78/99 ausrechnet und dann auf das Ergebnis 35.4545... MATH Frac anwendet, so erhält man den Bruch 390/11 . Dieser Bruch ist exakt berechnet !
Ebenso kann man 35.45454545 oder 35.45454545 45 eintippen und nach ENTER wieder MATH Frac aufrufen. Tippt man 35.45454546 ein, so erhält man nach ENTER MATH Frac überraschenderweise wieder 390/11. Dies zeigt, dass der Rechner TI84 versucht, für eingegebene Dezimalzahlen einen bestmöglichen Näherungsbruch zu finden.
Dies gelingt aber nicht immer, wie das Beispiel 35.45454547 zeigt . Es wird kein Bruch erzeugt !

Wie findet man einen solchen Näherungsbruch ?

Einen Näherungsbruch zu einer in Gleitkomma-Darstellung gegebenen Dezimalzahl x findet man durch Entwicklung von x in einen endlichen Kettenbruch (KB).
Die Anzahl der Glieder des KBs wird so gewählt, dass der KB sich von x nur noch um eine vorgegebene positive Schranke ε unterscheidet ; $|x - KB(x)| < \varepsilon$.

Ist x in der Normalform gegeben, z.B. x = 2,7145063, so muss man nur prüfen, ob der Dezimalwert des jeweiligen Kettenbruchs sich um höchstens $\varepsilon = 10^{-16}$ von x unterscheidet.
Die Doublewerte von KB und x stimmen dann bis auf die letzte Dezimale überein.

Beispiel: x = 0,1285714274142857 (Angabe in Gleitpunktarithmetik; 16 Dezimalen)

Entwicklung von x in einen Kettenbruch (Rechnung mit 16 Dezimalen, also mit Rundungsfehlern):

```
x = 0,1285714274142857 = 0+1/(1/0,1285714274142857) =
0+1/7,7777778477777793 = 0+1/(7+0,7777778477777793) =
0+1/(7+1/(1/0,7777778477777793)) =
0+1/(7+1/(1,2857141700000079)) = 0+1/(7+1/(1+0,2857141700000079)) =
0+1/(7+1/(1+1/(1/0,2857141700000079))) = 0+1/(7+1/(1+1/
(3,5000014175004773))) =
0+1/(7+1/(1+1/(3+0,5000014175004773))) = 0+1/(7+1/(1+1/(3+1/
(1/0,5000014175004773)))) =
0+1/(7+1/(1+1/(3+1/(1,9999943300141652)))) = 0+1/(7+1/(1+1/(3+1/
(1+0,9999943300141652)))) =0+1/(7+1/(1+1/(3+1/(1+1/
(1/0,9999943300141652))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1,0000056700179837))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+0,0000056700179837))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(1/0,0000056700179837)))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366,2836475599236996)))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+0,2836475599236996)))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(1/0,2836475599236996))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3,5255018596634401))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+0,5255018596634401))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+1/(1/0,5255018596634401)))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+1/(1,9029428376151785)))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+1/(1+0,9029428376151785)))))))) =
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+1/(1+1/
(1/0,9029428376151785))))))))) =0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/
(3+1/(1+1/(1,1074898192239561))))))))))=
0+1/(7+1/(1+1/(3+1/(1+1/(1+1/(176366+1/(3+1/(1+1/
(1+0,1074898192239561))))))))))
```

Nach 10 Gliedern kann der KB abgebrochen werden, denn
[0; 7,1,3,1,1,176366,3,1,1] = [0; 7,1,3,1,1,176366,3,2] (Korrektur am Ende !)
= 11111111 / 86419753 ≈ 0,12857142741428571 ,
also $|x - 0{,}12857142741428571| \leq 10^{-17}$.
Damit gibt es in der letzten Nachkommastelle keine Abweichung mehr.

Weiteres Beispiel(ohne Rechnung): x = 123456789 / 987654321 = 0,1249999988609375
Der Kettenbruch ist [0; 8,13717421] = 13717421 / 109739369 (= 123456789 / 987654321) .

Beispiele für Kettenbrüche

Alle angezeigten Elemente sind exakt !!

Kreiszahl: π =
[3;7,15,1,292,1,1,1,2,1,3,1,14,2,1,1,2,2,2,2,1,84,2,1,1,15,3,13,1,4,2,6,6,99,
1,2,2,6,3,5,1,1,6,8,1,7,1,2,3,7,1,2,1,1,12,1,1,1,3,1,1,8,1,1,2,1,6,1,1,5,2,2,3
,1,2,4,4,16,1,161,45,1,22,1,2,2,1,4,1,2,24,1,2,1,3,1,2,1,1,10,2,5,4,1,2,2,8,1,
5,2,2,26,1,4,1,1,8,2,42,2,1,7,3,3,1,1,7,2,4,9,7,2,3,1,57,1,18,1,9,19,1,2,18,1,
3,7,30,1,1,1,3,3,3,1,2,8,1,1,2,1,15,1,2,13,1,2,1,4,1,12,1,1,3,3,28,1,10,3,2,20
,1,1,1,1,4,1,1,1,5,3,2,1,6,1,4,1,120,2,1,1,3,1,23,1,15,1,3,7,1,16,1,2,1,21,2,1
,1,2,9,1,6,4,127,14,5,1,3,13,7,9,1,1,1,1,1,5,4,1,1,3,1,1,29,3,1,1,2,2,1,3,1,1,
1,3,1,1,10,3,1,3,1,2,1,12,1,4,1,1,1,1,7,1,1,2,1,11,3,1,7,1,4,1,48,16,1,4,5,2,1
,1,4,3,1,2,3,1,2,2,1,2,5,20,1,1,5,4,1,436,8,1,2,2,1,1,1,1,1,5,1,2,1,3,6,11,4,3
,1,1,1,2,5,4,6,9,1,5,1,5,15,1,11,24,4,4,5,2,1,4,1,6,1,1,1,4,3,2,2,1,1,2,1,58,5
,1,2,1,2,1,1,2,2,7,1,15,1,4,8,1,1,4,2,1,1,1,3,1,1,1,2,1, ...]

Eulerzahl: e =
[2;1,2,1,1,4,1,1,6,1,1,8,1,1,10,1,1,12,1,1,14,1,1,16,1,1,18,1,1,20,1,1,22,1,
1,24,1,1,26,1,1,28,1,1,30,1,1,32,1,1,34,1,1,36,1,1,38,1,1,40,1,1,42,1,1,44,1,1
,46,1,1,48,1,1,50,1,1,52,1,1,54,1,1,56,1,1,58,1,1,60,1,1,62,1,1,64,1,1,66,1,1,
68,1,1,70,1,1,72,1,1,74,1,1,76,1,1,78,1,1,80,1,1,82,1,1,84,1,1,86,1,1,88,1,1,9
0,1,1,92,1,1,94,1,1,96,1,1,98,1,1,100,1,1,102,1,1,104,1,1,106,1,1,108,1,1,110,
1,1,112,1,1,114,1,1,116,1,1,118,1,1,120,1,1,122,1,1,124,1,1,126,1,1,128,1,1,13
0,1,1,132,1,1,134,1,1,136,1,1,138,1,1,140,1,1,142,1,1,144,1,1,146,1,1,148,1,1,
150,1,1,152,1,1,154,1,1,156,1,1,158,1,1,160,1,1,162,1,1,164,1,1,166,1,1,168,1,
1,170,1,1,172,1,1,174,1,1,176,1,1,178,1,1,180,1,1,182,1,1,184,1,1,186,1,1,188,
1,1, ...]

Hier ist ein Bildungsgesetz zu erkennen ! Man könnte also die Folge fortsetzen.

Quadratwurzeln:

√(2) = [1; 2,2,2,2,2,2,2,2,2,2, ...] ≈ 1,4142135623730951
√(3) = [1; 1,2,1,2,1,2,1,2,1,2, ...] ≈ 1,7320508075688772
√(5) = [2; 4,4,4,4,4,4,4,4,4,4, ...] ≈ 2,23606797749979
√(6) = [2; 2,4,2,4,2,4,2,4,2,4, ...] ≈ 2,449489742783178
√(7) = [2; 1,1,1,4,1,1,1,4,1,1, ...] ≈ 2,6457513110645907
√(8) = [2; 1,4,1,4,1,4,1,4,1,4, ...] ≈ 2,8284271247461903
√(10)= [3; 6,6,6,6,6,6,6,6,6,6, ...] ≈ 3,1622776601683795

Die KBs der Quadratwurzeln haben stets eine Periode (rot markierte Zahlen !).

Allgemeine Kettenbruch-Formel für Quadratwurzeln natürlicher Zahlen n

$$\sqrt{n} = [\, a_0\, ;\, \overline{a_1, a_2, \ldots , a_k, 2\cdot a_0}\,] \qquad \text{mit}\ \ a_0 = \operatorname{int}(\sqrt{n})$$

Beweis für $\sqrt{2}$:

Wegen $(\sqrt{2}-1)(\sqrt{2}+1) = 1$ gilt $\boxed{\sqrt{2} = 1 + \dfrac{1}{1+\sqrt{2}}}$

Setzt man den Term für $\sqrt{2}$ sukzessive in die Formel ein, so erhält man:

$$\sqrt{2} = 1 + \cfrac{1}{1+1+\cfrac{1}{1+\sqrt{2}}} = 1 + \cfrac{1}{2+\cfrac{1}{1+\sqrt{2}}} = 1 + \cfrac{1}{2+\cfrac{1}{1+1+\cfrac{1}{1+\sqrt{2}}}} = 1 + \cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{1+\sqrt{2}}}} = \ldots$$

Man kann erkennen, dass sich das Element 2 immer wieder reproduziert und somit eine Periode entsteht !

Kubikwurzeln:

$\sqrt[3]{(2)}$ = [1; 3,1,5,1,1,4,1,1,8,1,14,1,10,2,1,4,12,2,3,2,1,3, ...] ≈ 1,2599210498948732
$\sqrt[3]{(3)}$ = [1; 2,3,1,4,1,5,1,1,6,2,5,8,3,3,4,2,6,4,4,1,3, ...] ≈ 1,4422495703074083

Kein Gesetz erkennbar !   Die KBs der Kubikwurzeln besitzen keine Periode !

Goldene Zahl (Verhältnis des "Goldenen Schnitts"):

Die "Goldene Zahl" entsteht aus dem Ansatz der harmonischen Teilung einer Strecke, was auch als "Goldener Schnitt" bekannt ist .

Die Länge der Strecke sei mit 1 (=100%) angenommen !
Für die beiden Teile a, b (mit a > b; a+b=1) der Strecke soll gelten: $\Phi = a / b = (a+b) / a$.
Hieraus ergibt sich $a^2 = (a+b)\cdot b = (a+1-a)\cdot(1-a) = 1-a$, und somit $a^2 + a - 1 = 0$.

Die beiden Lösungen dieser quadratischen Gleichung sind $a = (-1 \pm \sqrt{5}) / 2$.
Die negative Lösung kommt als Streckenlänge nicht in Betracht, somit ist

der größere Teil $a = (-1 + \sqrt{5}) / 2 \approx 0{,}618$ und

der kleinere Teil $b = 1 - a = (3 - \sqrt{5}) / 2 \approx 0{,}382$.

Für die Goldene Zahl gilt dann:

$\Phi = a / b = (-1 + \sqrt{5}) / (3 - \sqrt{5}) = (2 + 2\cdot\sqrt{5}) / 4 = (1 + \sqrt{5}) / 2 \approx 1{,}618$.

$$\boxed{\Phi = \frac{1+\sqrt{5}}{2} = [1;\, 1,1,1,1,\, \ldots\,] = [1;\, \bar{1}] \approx 1{,}6180339887498948482045868343656381}$$

Dies ist der einfachste mögliche periodische Kettenbruch (Periode = 1) !

Nachteil:

Die Berechnung einer hinreichend guten Approximation dauert jedoch ziemlich lange, z.B.

[1; 1,1,1,1,1,1,1,1,1,1] = 144 / 89 ≈ 1,6179775

[1; 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] = 17711 / 10946 ≈ 1,6180339850

[1; 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] = 2178309 / 1346269 ≈ 1,6180339887496481

Erweiterung (**BigIntegers**) :

Verwendet man eine Bruchklasse mit BigInteger-Elementen, so kann die Genauigkeit zur Ermittlung von Kettenbruchdarstellungen beliebig gesteigert werden.

Gibt man z.B. die Zahl π mit 1000 Dezimalen vor, so liefert ein entsprechendes Programm mit Kettenbrucherzeugung folgendes:

```
π ≈ [3;
7,15,1,292,1,1,1,2,1,3,1,14,2,1,1,2,2,2,2,1,84,2,1,1,15,3,13,1,4,2,6,6,99,1,2,2,6,3,5,1,1,
6,8,1,7,1,2,3,7,1,2,1,1,12,1,1,1,3,1,1,8,1,1,2,1,6,1,1,5,2,2,3,1,2,4,4,16,1,161,45,1,22,1,
2,2,1,4,1,2,24,1,2,1,3,1,2,1,1,10,2,5,4,1,2,2,8,1,5,2,2,26,1,4,1,1,8,2,42,2,1,7,3,3,1,1,7,
2,4,9,7,2,3,1,57,1,18,1,9,19,1,2,18,1,3,7,30,1,1,1,3,3,3,1,2,8,1,1,2,1,15,1,2,13,1,2,1,4,1
,12,1,1,3,3,28,1,10,3,2,20,1,1,1,1,4,1,1,1,5,3,2,1,6,1,4,1,120,2,1,1,3,1,23,1,15,1,3,7,1,1
6,1,2,1,21,2,1,1,2,9,1,6,4,127,14,5,1,3,13,7,9,1,1,1,1,1,5,4,1,1,3,1,1,29,3,1,1,2,2,1,3,1,
1,1,3,1,1,10,3,1,3,1,2,1,12,1,4,1,1,1,1,7,1,1,2,1,11,3,1,7,1,4,1,48,16,1,4,5,2,1,1,4,3,1,2
,3,1,2,2,1,2,5,20,1,1,5,4,1,436,8,1,2,2,1,1,1,1,1,5,1,2,1,3,6,11,4,3,1,1,1,2,5,4,6,9,1,5,1
,5,15,1,11,24,4,4,5,2,1,4,1,6,1,1,1,4,3,2,2,1,1,2,1,58,5,1,2,1,2,1,1,2,2,7,1,15,1,4,8,1,1,
4,2,1,1,1,3,1,1,1,2,1,1,1,1,1,9,1,4,3,15,1,2,1,13,1,1,1,3,24,1,2,4,10,5,12,3,3,21,1,2,1,34
,1,1,1,4,15,1,4,44,1,4,20776,1,1,1,1,1,1,1,23,1,7,2,1,94,55,1,1,2,1,1,3,1,1,32,5,1,14,1,1,
1,1,1,3,50,2,16,5,1,2,1,4,6,3,1,3,3,1,2,2,2,5,2,2,2,28,1,1,13,1,5,43,1,4,3,5,3,1,4,1,1,2,2
,1,1,19,2,7,1,72,3,1,2,3,7,11,1,2,1,1,2,2,1,1,2,1,1,1,1,1,33,7,19,1,19,3,1,4,1,1,1,1,2,3,1
,3,2,2,2,2,4,1,1,1,4,2,3,1,1,1,1,11,1,1,2,1,2,1,2,2,1,7,2,27,1,1,6,2,1,9,6,26,1,1,3,2,1,1,
1,1,1,15,1,36,4,2,2,1,22,2,1,106,2,2,1,3,1,12,10,7,1,2,1,1,1,1,8,2,4,5,3,2,1,4,23,1,18,2,1
0,3,1,6,6,13,8,6,2,2,2,2,1,1,1,3,1,7,17,1,1,1,2,5,5,1,1,2,11,1,6,1,6,1,29,4,29,3,5,3,1,141
,1,2,7,7,2,2,7,1,1,7,1,7,1,2,4,1,1,1,30,1,12,4,18,10,2,8,1,2,2,2,4,13,1,5,4,1,6,1,1,11,2,4
,2,1,1,3,3,12,1,1,39,5,1,1,16,125,1,4,1,2,1,19,1,4,1,1,2,1,4,1,10,1,4,2,1,1,1,5,10,4,14,1,
13,41,1,4,1,8,1,1,2,1,3,1,6,1,3,2,2,2,1,4,1,14,1,2,8,1,8,3,3,3,1,37,4,2,4,1,3,4,25,4,27,2,
7,1,1,2,6,1,1,1,12,1,2,2,2,13,12,1,3,1,6,1,1,33,1,5,3,1,5,15,8,8,47,1,3,2,12,2,12,1,12,1,2
,5,3,1,1,1,1,2,3,5,4,2,1,1,5,1,9,14,1,1,3,2,1,9,3,22,13,1,1,3,20,1,1,61,1,376,2,107,1,10,3
,2,2,31,1,2,10,2,2,62,2,2,7,4,5,6,1,1,1,1,2,8,2,73,3,5,42,1,3,2,1,1,59,6,1,1,1,5,1,6,1,2,6
,1,1,1,1,3,2,1,3,1,8,1,4,2,5,4,7,1,4,2,2,6,1,1,2,2,1,1,1,1,1,2,1,2,2,5,1,2,1,1,10,1,6,1,12
9,1,4,65,2,4,4,3,2,13,1,57,1,1,1,1,2,2,3,3,1,4,1,6,21,1,2,1,5,1,6,3,16,1,9,2,1,2,1,10,6,1]
```
die blauen Ziffern sind falsch !

"Optimale" Bruch-Approximationen für $\varepsilon = 10^{-16}$ (Java) :

Dies sind für den Java-Datentyp "double" optimierte Bruch-Approximationen !
Voraussetzung ist, dass eine genügend genaue dezimale Approximation der Zahl x bekannt ist,
z.B. `x = e = 2,71828182845904523536028747135266249775724709369995957496696762772407663035354759457 .`
Nur dann kann auch eine genügend genaue Kettenbruchentwicklung von x garantiert werden !

π = [3; 7,15,1,292,1,1,1,2,1,3,1,14,3] =
245850922 / 78256779 ≈ 3,1415926535897932
Math.PI ≈ 3,141592653589793

$$3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{292+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\cfrac{1}{14+\cfrac{1}{3}}}}}}}}}}}}}$$

e = [2; 1,2,1,1,4,1,1,6,1,1,8,1,1,10,1,1,12,1,1,14] =
410105312 / 150869313 ≈ 2,7182818284590452
Math.E ≈ 2,718281828459045

√(2) = [1; 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2] =
131836323 / 93222358 ≈ 1,414213562373095048
Math.sqrt(2) ≈ 1,4142135623730951

√(3) = [1; 1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2] =
138907099 / 80198051 ≈ 1,732050807568877203
Math.sqrt(3) ≈ 1,7320508075688772

√(5) = [2; 4,4,4,4,4,4,4,4,4,4,4,4,4,4] = 299537289 / 133957148 ≈ 2,236067977499789
Math.sqrt(5) ≈ 2,23606797749979

√(6) = 2; 2,4,2,4,2,4,2,4,2,4,2,4,2,4,2,4] = 205117922 / 83739041 ≈
2,44948974278317803
Math.sqrt(6) ≈ 2,449489742783178

√(7) = [2; 1,1,1,4,1,1,1,4,1,1,1,4,1,1,1,4,1,1,1,4,1,1,1,4,1,2] =
130576328 / 49353213 ≈ 2,64575131106459066
Math.sqrt(7) ≈ 2,6457513110645907

√(10) = [3; 6,6,6,6,6,6,6,6,6,6,6] = 243289797 / 76934989 ≈ 3,1622776601683793
Math.sqrt(10) ≈ 3,1622776601683795

(1+√(5))/2 = [1; 1,1,1,1,1, ...] = 165580141 / 102334155 ≈ 1,61803398874989484
(1+Math.sqrt(5))/2 ≈ 1,618033988749895

∛(2) = 1; 3,1,5,1,1,4,1,1,8,1,14,1,10,2,1,4,12] = 186150494 / 147747745 ≈
1,259921049894873184
Math.cbrt(2) ≈ 1,2599210498948732

∛(3) = [1; 2,3,1,4,1,5,1,1,6,2,5,8,3,3,4,2] =
59472423 / 41235875 ≈ 1,442249570307408329
Math.cbrt(3) ≈ 1,4422495703074083

∛(5) = [1; 1,2,2,4,3,3,1,5,1,1,4,10,17,1,14] =
115943811 / 67804352 ≈ 1,70997594667669709
Math.cbrt(5) ≈ 1,709975946676697

Beispiele (Ergebnisse wurden ermittelt mit Java 8 ; Datentyp double):

Bruch in Kettenbruch umwandeln:

67/29 = [2; 3, 4, 2]

77708431 / 2640858 = [29; 2, 2, 1, 5, 1, 4, 1, 1, 2, 1, 6, 1, 10, 2, 2, 3]

Und rückwärts: Kettenbruch in Bruch umwandeln:

[29; 2,2,1,5,1,4,1,1,2,1,6,1,10,2,2,3] = 77708431 / 2640858

Kettenbruch in Dezimalzahl umwandeln:

e = [2; 1,2,1,1,4,1,1,6,1,1,8,1,1,10,1,1,12,1,1,14,1,1,16,1,1,18,1,1,20,1,1,22,1,1,24,1,1,
26,1,1,28,1,1,30,1,1] = 2.7182818284590455

(√(5)-1)/2 = [0; 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] = 0.6180339887498969

-77708431 / 2640858 = [-29; 2,2,1,5,1,4,1,1,2,1,6,1,10,2,2,3] = -29.42544847167095

√(2) = [1; 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2] =
1.4142135623730951

pi = [3; 7,15,1,292,1,1,1,2,1,3,1,14,2,1,1,2,2,2,2,1,84,2,1,1,15,3,13,1,4,2,6,6,99,1,2,2,6,3,5,1,1,6,8,1,7,1,
2,3,7,1,2,1,1,12,1,1,1,3,1,1,8,1,1,2,1,6,1,1,5,2,2,3,1,2,4,4,16,1,161,45,1,22,1,2] = 3.141592653589793

log$_{10}$(2) = [0; 3,3,9,2,2,4,6,2,1,1,3,1,18,1,6,1,2,1,1,5,2,1,12,12,2,11,1,141] = 0.3010299956639812

ln(2) = [0; 1,2,3,1,6,3,1,1,2,1,1,1,1,3,10,1,1,1,2,1,1,1,1,3,2,3,1,13,7,4,1,1,1,7,2,4,1,1,2,5,
14,1,10,1,4,2,18,3,1,4,1,6,2,7,3,3,1,13,3,1,4,4,1,3,1,1,1,1,2,17,3,1,2,32,1,1,1,1,3] = 0.6931471805599453

99999999 / 777777777 = [0; 7,1,3,1,1,176366,3,1,1] = 0.1285714274142857

Dezimalzahl in Kettenbruch umwandeln (rot = falsche Elemente ; blau = Periode) :

pi = 3.141592653589793 = [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 3, 3, 23, 1, 1, 7, 4, 35, 1, 1, 1, 2, 3, 3, 3,
3, 1, 1, 14, 6, 4, 5, 1, 7, 1, 5, 1, 1, 3, 18, 2, 1, 2, 4, 2, 96, 2, 3]
genauer: pi =
[3; 7,15,1,292,1,1,1,2,1,3,1,14,2,1,1,2,2,2,2,1,84,2,1,1,15,3,13,1,4,2,6,6,99,1,2,2,6,3,5,1,1,6,8,1, ...]

e = 2.718281828459045 = [2; 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 11, 3, 2, 1, 3, 1, 73, 6, 1,
1, 1, 1, 1, 2, 31, 1, 1, 1, 2, 1, 1, 2, 1, 2, 15, 9, 1, 3, 1, 4, 2]
genauer: e =
[2;1,2,1,1,4,1,1,6,1,1,8,1,1,10,1,1,12,1,1,14,1,1,16,1,1,18,1,1,20,1,1,22,1,1,24,1,1,26,1,1,28,1,1, ...]

√(2) = 1.4142135623730951 = [1; 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 3, 3]
genau: [1; 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...] ; Periode !

√(3) = 1.7320508075688772 = [1; 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 3]
genau: [1; 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, ...] ; Periode !

√(31) = 5.5677643628300215 = [5; 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 7, 8, 3, 1, 2, 2]
genau: [5; 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 3, 5, 3, 1, 1, 10, 1, 1, 3, 5,
3, 1, 1, 10, 1, 1, ...] ; Periode !

9,7 = [9; 1,2,3,15011998757901,1,1,1,8,3,6,1477558932864,2,1,41,1,4139337892803,1,3,4,59,1,
19628659463,1,4] genau: [9; 1, 2, 3] ; abbrechend !

Dezimalzahl in bestmöglichen Näherungsbruch(KB) umwandeln:

0.5151515151515151 = 17 / 33
29.42544847167095 = 77708431 / 2640858
8000.005840104263 = 9876543210 / 1234567