Algorithmen und Datenstrukturen im WS 2005/06

Numerische Algorithmen

Numerische Algorithmen

- 'Rechnen' um Zahlenwerte zu ermitteln
  - Beispiele
    - Newton-Verfahren
      - Newton-Raphton
      - Newton-Sekantenverfahren
    - Intervallhalbierung
  - Berechnung von Ausdrücken
    - Beispiel Potenz: b^e.
    - Literatur: http://ls5-www.cs.uni-dortmund.de/teachings/courses/dap1.ht
    - Hier
      - in Java
      - in 3AA

Numerische Algorithmen

```
 1            // Potenz O(logarithmisch)
 2
 3            if  val basis == con 0.0 then goto con Null
 4            adr result := con  1.0     // x^0 = 1 (x>0)
 5 NochMal    if val exponent == con 0.0 then goto con Fertig
 6            adr h := val exponent & con 1
 7            if val h == con 0 then goto con Weiter
 8            adr result *f= val basis  // result*basis^(2^(k-1))
 9 Weiter     adr basis *f= val basis   // basis^(2^k)
                                        // b^2^(k-1)*b^2^(k-1)=b^2*2^(k-1)=b^2^k
10            adr exponent >>>= con 1   // exponent/2
11            goto con NochMal
12 Null       adr result := con 0.0     // 0^e = 0
13 Fertig     stop
14
15 basis      con 2.0
16 exponent   con 1000
17 h
18 result     con 0.0
```

Numerische Algorithmen

```
 1            // Potenz O(linear)
 2
 3            if  val basis == con 0.0 then goto con Null
 4            adr result := con  1.0 // x^0=1!
 5 NochMal    if val exponent == con 0.0 then goto con Fertig
 6            adr result *f= val basis
 7 Weiter     adr exponent -= con 1
 8            goto con NochMal
 9 Null       adr result := con 0.0 // 0^e=0!
10 Fertig     stop
11
12 basis      con -2.0
13 exponent   con 9
14 result     con 0.0
```

Numerische Algorithmen

```
 1            // Wurzelziehen (ganzzahlige Wurzel iterativ)
 7
 8 NochMal    if val b > val Q then goto con Fertig // @=k++
 9            adr b += val a     // b = (k+1)²
10            adr a += con 2     // a = 3+2*k
11            adr w += con 1     // w = k
12            goto con NochMal
13            Fertig stop        // b = (k+1)² > Q ≥ k²
14 a          con 3 // 3 5 7 9       //.. k = w ⇒ w+1 > √Q ≥ k
15 b          con 1 // 1 4 9 16
16 w          con 0 // 0 1 2 3 4
17 Q          con 123456787654321
```

Numerische Algorithmen

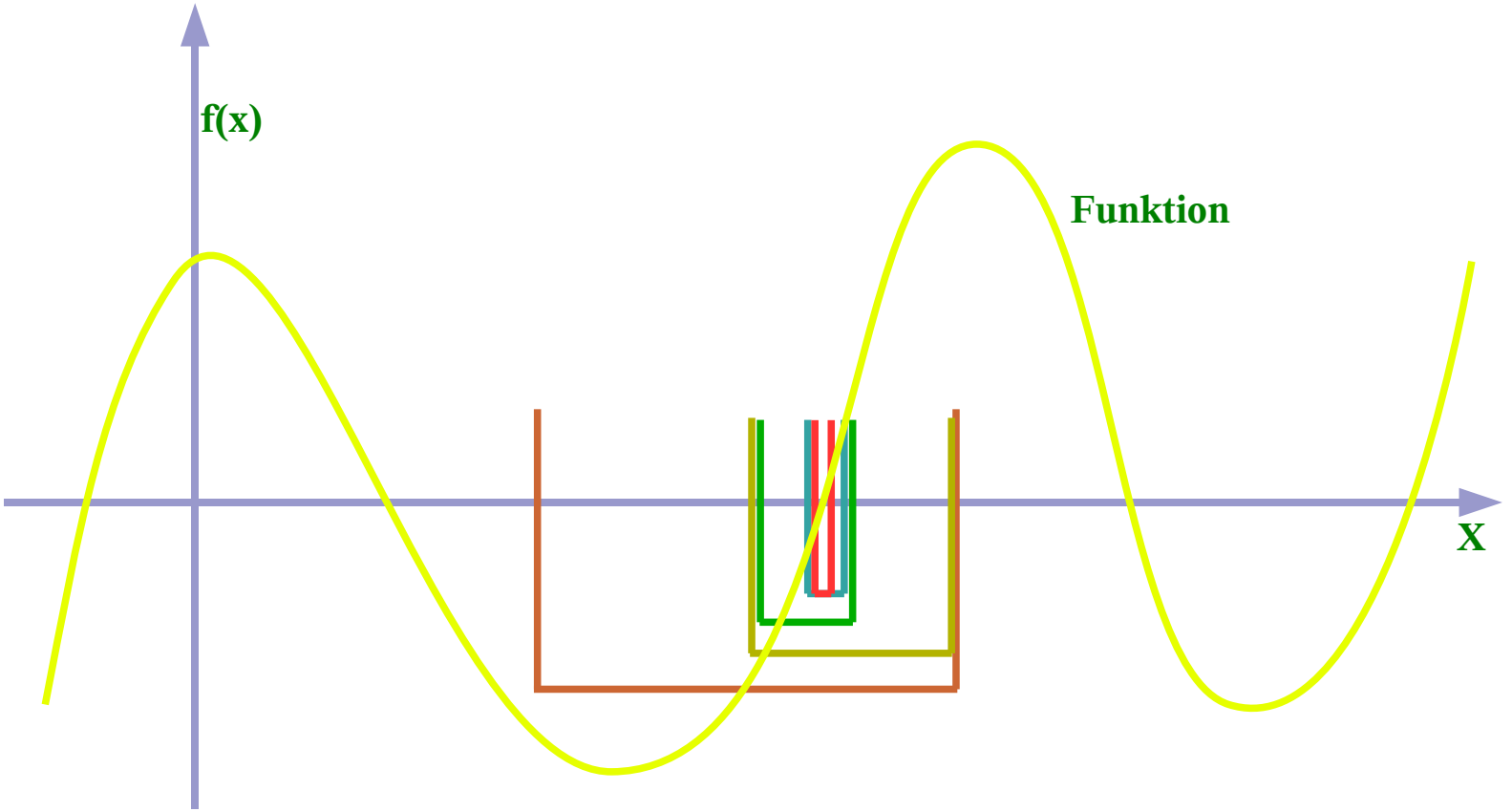
```
1           // Wurzel: Q = 2^31;
2           // while Q>0 {
            //   hilf := (W^2+2WQ+Q^2)
3           //   if hilf<=Zahl && hilf>0 then
4           //     W^2 := hilf
4           //     W |= Q
4           //     2WQ |= Q^2 / 2
5           //   Q /= 2
5           //   2WQ /= 2
5           //   Q^2 /= 4
6
7           nop
8  While    if val Q <= con 0 then goto con Fertig      // while Q>0 {
9           adr hilf := val Wurzel2 + val wq2
10          adr hilf += val Q2                          //   hilf := (W^2+2WQ+Q^2)
11          if val hilf > val Zahl then goto con Sprung // if (Q+W)*(Q+W)<=Zahl
12          if val hilf < con 0 then goto con Sprung    // if (Q+W)*(Q+W)>0
13          adr Wurzel2 := val hilf                     // W^2 := hilf
14          adr Wurzel := val Wurzel | val Q            // Wurzel = Wurzel' + Q
15          adr hilf := val Q2 << con 1
15          adr wq2 := val wq2 | val hilf   // wq2 = 2*Wurzel*Q = 2*Q*(Wurzel'+Q) =
    2*Q*Wurzel' + 2*Q*Q = wq2' + 2*Q2
16 Sprung   adr Q := val Q >>> con 1                    // Q   /= 2
17          adr wq2 := val wq2 >>> con 1                // 2WQ /= 2
18          adr Q2 := val Q2 >>> con 2                  // Q^2 /= 2
19          goto con While                              // Noch mal
20 Fertig   adr Q := val Wurzel * val Wurzel
21          adr Q := val Zahl - val Q                   // Q = Z - W*W
22          stop
23
24 Zahl     con 12345678987654321 // -1 >>> con 1 // 12345678987654322 //
25 Q        con 1 << con 31
26 Q2       con 1 << con 62 // Q2 = Q^2
27 Wurzel   con 0
28 Wurzel2  con 0
29 wq2      con 0
30 hilf
```

Numerische Algorithmen

```
 While  if val Q <= con 0 then goto con Fertig// while Q>0 { //(!Q<=0)
  9     adr hilf := val Wurzel2 + val wq2
 10     adr hilf += val Q2                        //hilf:=(W^2+2WQ+Q^2)
 11     if val hilf > val Zahl then goto con Sprung//if(Q+W)*(Q+W)<=Zahl
 12     if val hilf < con 0 then goto con Sprung   // if (Q+W)*(Q+W)>0
 13     adr Wurzel2 := val hilf                   // W^2 := hilf
 14     adr Wurzel := val Wurzel | val Q          // Wurzel = Wurzel' + Q
 15     adr hilf := val Q2 << con 1
 15     adr wq2 := val wq2 | val hilf
// wq2=2*Wurzel*Q = 2*Q*(Wurzel'+Q) = 2*Q*Wurzel' + 2*Q*Q = wq2' + 2*Q2
 16 Sprung adr Q := val Q >>> con 1              // Q   /= 2
 17     adr wq2 := val wq2 >>> con 1            // 2WQ /= 2
 18     adr Q2 := val Q2 >>> con 2              // Q^2 /= 4
 19     goto con While                          // Noch mal
 20 Fertig adr Q := val Wurzel * val Wurzel
 21     adr Q := val Zahl - val Q               // Q = Z - W*W
 22     stop
```

Numerische Algorithmen

- Regula Falsi, Binärteilung



Numerische Algorithmen

- Newton Verfahren
  - Schnittpunkt aus Tangente in Punkt x_k
  - Hohe Konvergenz-Geschwindigkeit
  - Nur bei möglichstglatten Kurven
  - Schlechte Konvergenz bei großer Entfernung vom Nullpunkt
  - Ableitung muss bekannt sein!