

3. Zeigen Sie daß die n -te Fibonacci-Zahl gegeben ist durch

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right].$$

Q_n ab $n=2$ zunehmend; $\frac{Q_{n+1}}{Q_n} \leq 2$

$f(x) = \sum_{n=0}^{\infty} Q_{n+1} x^n$ konv. für $|x| < \frac{1}{2}$

für $|x| < \frac{1}{2}$ gilt: $f(x) - x f(x) - x^2 f(x) = 1$

$$\Rightarrow f(x) = -\frac{1}{x^2 + x - 1} \quad x^2 + x - 1 \stackrel{!}{=} 0$$

$$x_{1,2} = \frac{-1 \pm \sqrt{1^2 - 4 \cdot 1 \cdot (-1)}}{2}$$

Nullstellen: $x_1 = \frac{-1 + \sqrt{5}}{2} \quad x_2 = \frac{-1 - \sqrt{5}}{2}$

$$f(x) = \frac{1}{\sqrt{5}} \left(\frac{1}{x - x_2} - \frac{1}{x - x_1} \right) = \frac{1}{\sqrt{5}} \left(\frac{1}{x_1} \cdot \frac{1}{1 - \frac{x}{x_1}} - \frac{1}{x_2} \cdot \frac{1}{1 - \frac{x}{x_2}} \right)$$

x_1^n x_2^n

$|x|$ hinreichend klein:

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \frac{1}{\sqrt{5}} \left(\frac{1}{x_1^{n+1}} - \frac{1}{x_2^{n+1}} \right) x^n \\ &= \frac{1}{\sqrt{5}} \sum_{n=0}^{\infty} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right] x^n \\ \Rightarrow Q_n &= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right] \end{aligned}$$

2. Ein lineares Feld $A[1 \dots n]$ enthalte ganzzahlige Einträge. Es soll so umgeordnet werden daß im ersten Teil des Feldes alle negativen Zahlen stehen, im mittleren Teil alle Nullen und im letzten Teil alle positiven Zahlen. Beschreiben Sie einen Algorithmus der eine solche Umordnung durchführt ohne zusätzlichen Speicher zu verwenden (außer Indizes und Zählvariablen). Der Zeitaufwand des Algorithmus soll linear in n sein.

$$A = [0 \ 0 \ | \ 0, 5, 3, \ | \ -4, -1]$$

$$A[1, \text{neg}], \quad A[\text{neg}+1, \text{zero}], \quad A[\text{zero}+1, n]$$

```
for i = 1 to n
  if A[i] < 0 then
    neg = neg + 1;
  if A[i] = 0 then
    zero = zero + 1;
```

} $\Theta(n)$

```
for i = 1 to neg
  j = 1;
  fin = false;
  if A[i] == 0 then
    while (j+neg <= n) && fin == false
      if (A[j+neg] < 0) then
        A[i] = A[j+neg];
        A[j+neg] = 0;
        fin = true;
        j = j + 1;
```

} #neg mal

} $\Theta(n)$

} $\Theta(n^2)$?

```
if A[i] > 0 then
  j = 1;
  fin = false;
  while (j+neg <= n) && fin == false
    if A[j+neg] < 0 then
      A[j+neg] = A[i] + A[j+neg];
      A[i] = A[j+neg] - A[i];
      A[j+neg] = A[j+neg] - A[i];
      fin = true;
      j = j + 1;
```

} $\Theta(n)$

```
for i = neg+1 to neg+zero
  j = 1;
  fin = false;
  if A[i] > 0 then
    while j+zero+neg <= n && fin == false
      if A[j+zero+neg] == 0 then
        A[j+zero] = A[i];
        A[i] = 0;
        fin = true;
        j = j + 1;
```

} $n - \#0$ mal

} $\Theta(n)$

} $\Theta(n^2)$?

1. Es soll ein einfaches Sortierverfahren (bubble sort) entwickelt und bezüglich seines Aufwands analysiert werden: Gegeben sei ein Array $A[1 \dots n]$ mit n ganzen Zahlen und folgender Pseudocode:

```

BubbleSort(A, n)
1: FOR j=1 TO n
2:   FOR i=1 TO n-1
3:     IF A[i]>A[i+1]
4:       tmp=A[i]
5:       A[i]=A[i+1]
6:       A[i+1]=tmp

```

$A = [1, 5, 3, 4, 6, 2]$

1, 3, 5, 4, 6, 2

1, 3, 4, 5, 6, 2

1, 3, 4, 2, 5, 6

1, 3, 2, 4, 5, 6

1, 2, 3, 4, 5, 6

1, 2, 3, 4, 5, 6

Es wird das aktuelle mit dem nächsten verglichen; sie werden nach aufsteigender Größe sortiert; da jedes Element n -mal verglichen wird hat das kleinste Element die Chance im Worst Case alle Stellen zu durchwandern;

k - kleinste Element nach $A[k]$ je nach seiner Startposition x : $\sqrt{(x-k)^2}$ Durchläufe (n Schritte) pro Durchlauf kann die Zahl um max einen Platz vorrücken

erste Verbesserung:

```

j = 1;
swep = true;
while j < n & (swep == true)
  j = j + 1
  swep = false
  for i = 1 to n-1
    if A[i]>A[i+1] then
      temp=A[i]
      A[i]=A[i+1]
      A[i+1]=temp
      swep = true

```

Abbruch wenn Durchlauf ohne Vertauschung;

\Rightarrow best case 1 Durchlauf, $\mathcal{O}(n)$
 worst case n Durchläufe

$\mathcal{O}(n \cdot n) = \mathcal{O}(n^2)$

Zweite Verbesserung:

```
j = 1;
swep = true;
lastSwep = n;
while j <= n & (swep == true)
  j = j + 1
  swep = false
  for i = 1 to lastSwep-1
    if A[i] > A[i+1] then
      temp = A[i]
      A[i] = A[i+1]
      A[i+1] = temp
      swep = true
  lastSwep = i
```

die letzte Stelle ist bei jedem Durchlauf die größte Zahl; sie braucht nicht mehr verglichen werden

⇒ best case 1 Durchlauf = $O(n)$
worst case n Durchläufe,
aber pro Durchlauf $n-1$ Schritte
 $O(n \cdot (n-1)) = O(n^2)$