SAARLAND UNIVERSITY
COMPUTER SCIENCE

Vergleich von Laufzeiten von Algorithmen

Laufzeit: Funktion der Eingabegröße (und des Rechners)

Wir wollen Funktionen *„der Größe nach“* vergleichen. Wie?

Für diesen Vergleich betrachten wir das Wachstum der Funktion, wenn das Argument sehr groß wird, also gegen Unendlich geht.

„Asymptotische Betrachtung“

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Def: $f,g : \mathbb{N} \to \mathbb{R}$

$f \leq_{fü} g$: für alle n, bis auf endlich viele Ausnahmen gilt $f(n) \leq g(n)$

$(\exists\, n_0 \in \mathbb{N}\ \forall\, n \geq n_0 : f(n) \leq g(n))$

Def: $g \in FÜP = \{ h : \mathbb{N} \to \mathbb{R} \mid h \geq_{fü} 0 \}$

$O(g) = \{ f \in FÜP \mid \exists c>0 : f \leq_{fü} c \cdot g \}$

$o(g) = \{ f \in FÜP \mid \forall c>0 : f \leq_{fü} c \cdot g \}$

$\Omega(g) = \{ f \in FÜP \mid \exists c>0 : f \geq_{fü} c \cdot g \}$

$\omega(g) = \{ f \in FÜP \mid \forall c>0 : f \geq_{fü} c \cdot g \}$

$\Theta(g) = O(g) \cap \Omega(g)$

Def: $g \in FÜP = \{ h : \mathbb{N} \to \mathbb{R} \mid h \geq_{fü} 0 \}$

$O(g) = \{ f \in FÜP \mid \exists c>0 : f \leq_{fü} c \cdot g \}$

$o(g) = \{ f \in FÜP \mid \forall c>0 : f \leq_{fü} c \cdot g \}$

$\Omega(g) = \{ f \in FÜP \mid \exists c>0 : f \geq_{fü} c \cdot g \}$

$\omega(g) = \{ f \in FÜP \mid \forall c>0 : f \geq_{fü} c \cdot g \}$

$\Theta(g) = O(g) \cap \Omega(g)$

$f \in O(g)$ entspricht f "$\leq$" g
$f \in o(g)$ entspricht f "$<$" g
$f \in \Omega(g)$ entspricht f "$\geq$" g
$f \in \omega(g)$ entspricht f "$>$" g
$f \in \Theta(g)$ entspricht f "$=$" g

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Wichtige Regeln

$f \in O(g) \Leftrightarrow g \in \Omega(f)$
$f \in o(g) \Leftrightarrow g \in \omega(f)$
$f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$

$f \in O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$
$f \in o(g) \wedge g \in o(h) \Rightarrow f \in o(h)$

Wenn $f_1 \in O(g_1)$, $f_2 \in O(g_2)$

dann $f_1 + f_2 \in O(\max\{g_1,g_2\})$
$f_1 \cdot f_2 \in O(g_1 \cdot g_2)$

$0<a<b \quad 0<\alpha<\beta \quad 1<A<B$

$\log^\alpha n$ "<" $\log^\beta n$ "<" n^a "<" $n^a \log^\alpha n$ "<" n^b "<" A^n "<" $n^a A^n$ "<" B^n

Quicksort

Wollen „schnellen" Sortieralgorithmus, der wenig Zusatzspeicher braucht (Mergesort benötigt ein Hilfsfeld für das „Mergen")

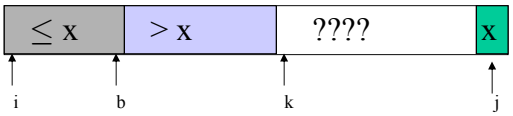
Idee: wähle „Pivotelement" x in Feld und stelle Feld so um:

sortiere Teilfeld der „kleinen" Elemente ($\leq$ x) rekursiv
sortiere Teilfeld der „großen" Elemente (> x) rekursiv

```
Quicksort( A , i , j )
        if  i < j then     p = Partition( A , i , j , A[j] )
                           Quicksort( A , i , p-1 )
                           Quicksort( A , p+1 , j )
```


Partitionieren

Invariante:




```
Partition( A , i , j , x )
        b = i-1
        for k = i to j do  swap( A[k] , A[b+1] )
                           if  A[b+1] ≤ x then b++
        return b
```

Überlege Korrektheit des Rückgabewertes im letzten Schritt !

Es wird kein Hilfsfeld verwendet.

Laufzeit ist O(j – i + 1), also linear in der Feldgröße

Laufzeitanalyse von Quicksort

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$T(n)$ Zeit, um ein Feld mit n Elementen zu sortieren

$T(n) = O(1)$ wenn $n \leq 2$

$T(n) \leq c \cdot n + T(n_1) + T(n_2)$ wenn $n>2$
dabei gilt $n_1 + n_2 = n-1$

Schlechtester Fall: $n_1=0$, $n_2=n-1$ $\Rightarrow$ $T(n) = \Theta(n^2)$

Bester Fall: $n_1 \approx n_2 \approx n/2$ $\Rightarrow$ $T(n) = \Theta(n \log n)$

„Durchschnittlicher Fall" ?

Erwartete Laufzeit Analyse von Quicksort

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Annahme:
Das gewählte Pivotelement ist mit gleicher Wahrscheinlichkeit $1/n$ das k-kleinste Element im Feld ($k=1,...,n$).
D.h. mit W'-keit $1/n$ gilt $n_1=k-1$ für $k=1,...,n$

$T(n)$ ... erwartete Laufzeit von Quicksort

$$T(n) \leq c \cdot n + \sum_{0 \leq k < n} ((1/n)(T(k) + T(n-1-k))) \text{ wenn } n>2$$
$$= c \cdot n + (2/n) \sum_{0 \leq k < n} T(k)$$

$\Rightarrow$ $T(n) = O(n \log n)$

Erwartete Laufzeit Analyse von Quicksort

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Annahme:
Das gewählte Pivotelement ist mit gleicher Wahrscheinlichkeit 1/n das k-kleinste Element im Feld (k=1,...,n).
D.h. mit W'-keit 1/n gilt n_1=k-1 für k=1,...,n

Wie kann man diese Annahme rechtfertigen?

Methode 1: Man nimmt einfach an, Eingaben kommen in "zufälliger" Reihenfolge (ohne wirkliche Rechtfertigung)

Methode 2: Man erzwingt die Richtigkeit der Annahme, indem man ein zufälliges Element des Feldes als Pivot wählt.

(Zufallszahlengenerator, randomisierter Algorithmus)