Sortierverfahren

	T_{best}	T_{avg}	T_{worst}	C_{best}	C_{avg}	C_{worst}	M_{best}	M_{avg}	M_{worst}	stabil	Speicher
Insertion Sort	n	n^2	n^2	n	n^2	n^2	n	n^2	n^2	1	-
Selection Sort	n^2	n^2	n^2	n^2	n^2	n^2	n	n	n	0	-
Merge Sort	n log n	n log n	n log n	n log n	n log n	n log n	n log n	n log n	n log n	1	$\Theta(n)$
Quick Sort	n log n	n log n	n^2	n log n	n log n [A]	n^2	n	-	n log n	0	mit Stack $\Theta(n)$
Heap Sort	n log n	n log n	n log n	n log n	n log n	n log n	n log n	n log n	n log n	0	
Bucket Sort	-	-	-	-	-	-	-	-	-	1 [B]	$\mathcal{O}(\mathrm{k})$[C]
Radix Sort	-	-	-	-	-	-	-	-	-	1 [D]	$\mathcal{O}(\mathrm{n})$[E]

[A] http://en.wikipedia.org/wiki/Quicksort [2010-04-29 02:30]
[B] http://de.wikipedia.org/wiki/Sortierverfahren [2010-04-29 02:30]
[C] http://de.wikipedia.org/wiki/Sortierverfahren [2010-04-29 02:30]
[D] http://de.wikipedia.org/wiki/Sortierverfahren [2010-04-29 02:30]
[E] http://de.wikipedia.org/wiki/Sortierverfahren [2010-04-29 02:30]

Abstrakte Datentypen und Datenstrukturen

	einfügen			entfernen			suchen			zugreifen			verketten		
	T_{best}	T_{avg}	T_{worst}	T_{best}	T_{avg}	T_{worst}	T_{best}	T_{avg}	T_{worst}	T_{best}	T_{avg}	T_{worst}	T_{best}	T_{avg}	T_{worst}
sequentiell	1	n	n	1	n	n	1	n	n	1	1	1	n2	n2	n2
verkettet	1	n	n	1	n	n	1	n	n	1	n	n	1	1	1

Erwartet man viele Operationen vom Typ Zugriff, sind Arrays schneller. Ein Nachteil entsteht beim Einfügen und Löschen.
Doppelt verkettete Listen sind beim Verketten (Einfügen am Anfang oder Ende der Liste) schneller.

Suchverfahren

		suchen			ins	entf	min	max	suc	pre
		C_{best}	C_{avg}	C_{worst}	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$	$\mathcal{O}$
Suche in **sequentiellen** Folgen	lineare (naive) Suche	1	n	n					∄	∄
	binäre Suche	1	$\log n$	$\log n$					∄	∄
binäre **Suchbäume**	natürliche binäre	1	$\log n$ [F]	n[G]						
	AVL	1[H]	$\log n$	$\log n$	$\log n$ [I]	$\log n$ [J]	$\log n$	$\log n$	$\log n$	$\log n$
	B	1[K]	$\log n$ [L]	$\log n$ [M]	$\log n$ [N]	$\log n$ [O]				
	B*	[P]	[Q]							

- Binäre Suche ist nur sinnvoll, wenn sich die Daten nicht zu oft ändern, sonst sind binäre Suchbäume besser geeignet.

[F] 1,38629…*$\log_2 n$

[G] wenn zu einer linearen Liste degeneriert

[H] Autor

[I] da die Rebalancierung in konstanter Zeit ausführbar ist

[J] da die Rebalancierung in konstanter Zeit ausführbar ist

[K] Autor

[L] $\mathcal{O}(\log_{\lceil \frac{m}{2} \rceil}(N+1))$

[M] $\mathcal{O}(\log_{\lceil \frac{m}{2} \rceil}(N+1))$

[N] $\mathcal{O}(\log_{\lceil \frac{m}{2} \rceil}(N+1))$

[O] $\mathcal{O}(\log_{\lceil \frac{m}{2} \rceil}(N+1))$

[P] im Unterschied zum B-Baum muss in jedem Fall bis zu einem Blatt gegangen werden, was die mittlere Anzahl an Schritten jedoch kaum erhöht

[Q] im Unterschied zum B-Baum muss in jedem Fall bis zu einem Blatt gegangen werden, was die mittlere Anzahl an Schritten jedoch kaum erhöht