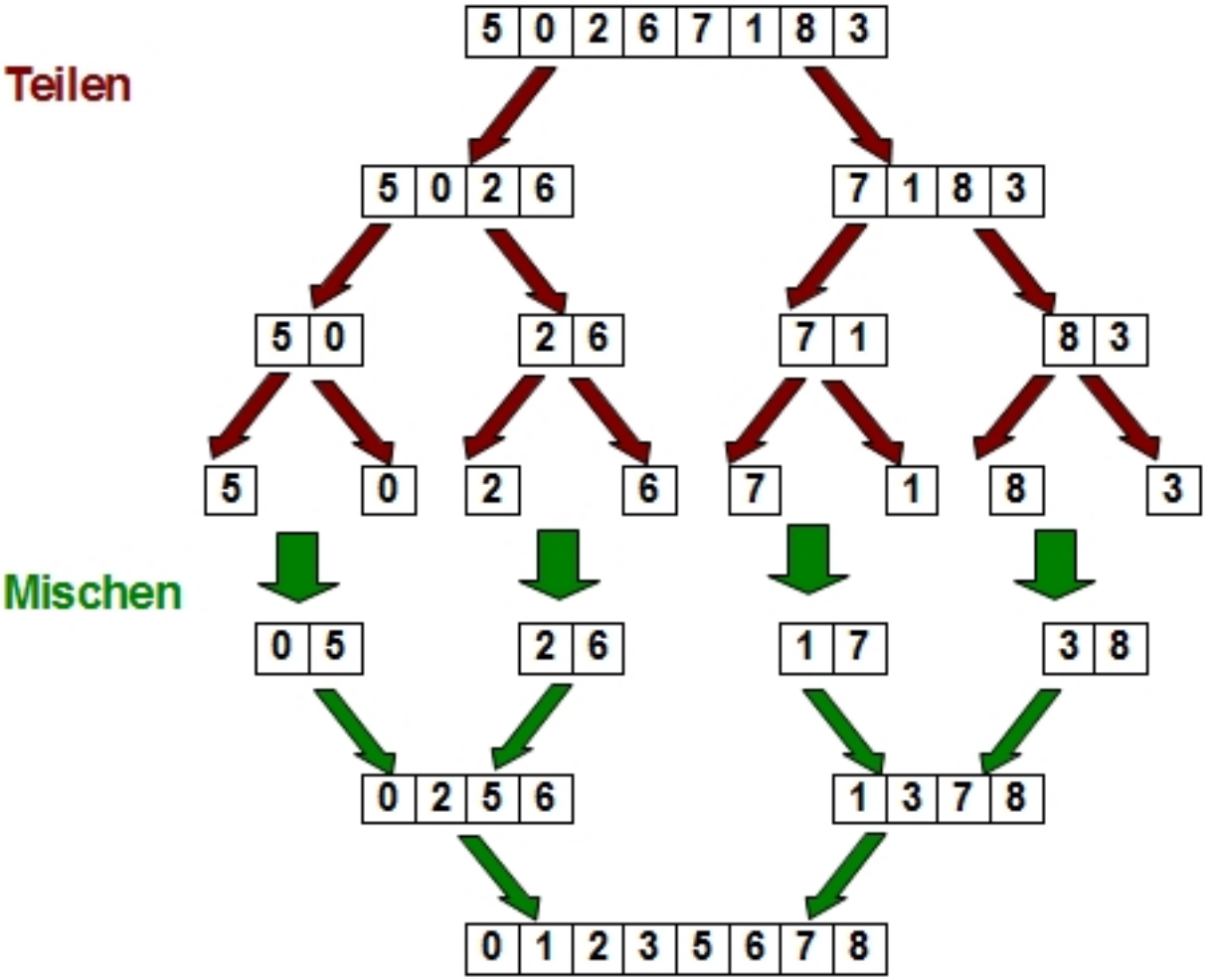
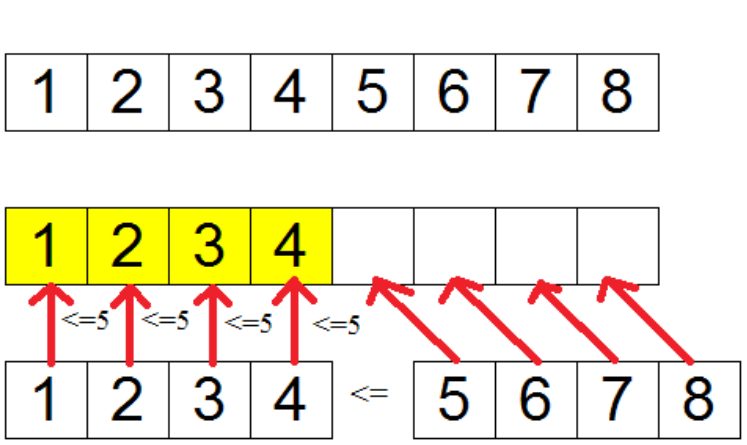
MergeSort


Teilen
Mischen
5 0 2 6 7 1 8 3
5 0 2 6
7 1 8 3
5 0
2 6
7 1
8 3
5
0
2
6
7
1
8
3
0 5
2 6
1 7
3 8
0 2 5 6
1 3 7 8
0 1 2 3 5 6 7 8

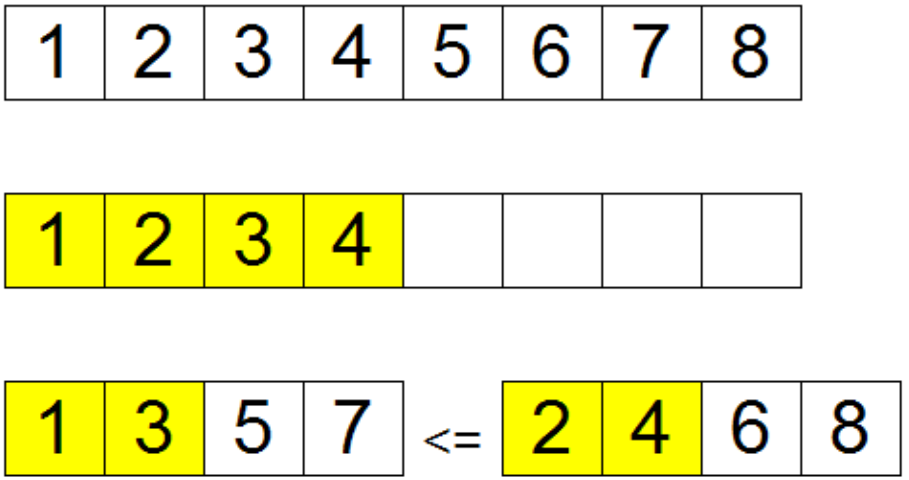
Best-Case

- O(n log n)
- Wenig Vergleiche beim Mergen
- z.B. Sortierte Liste

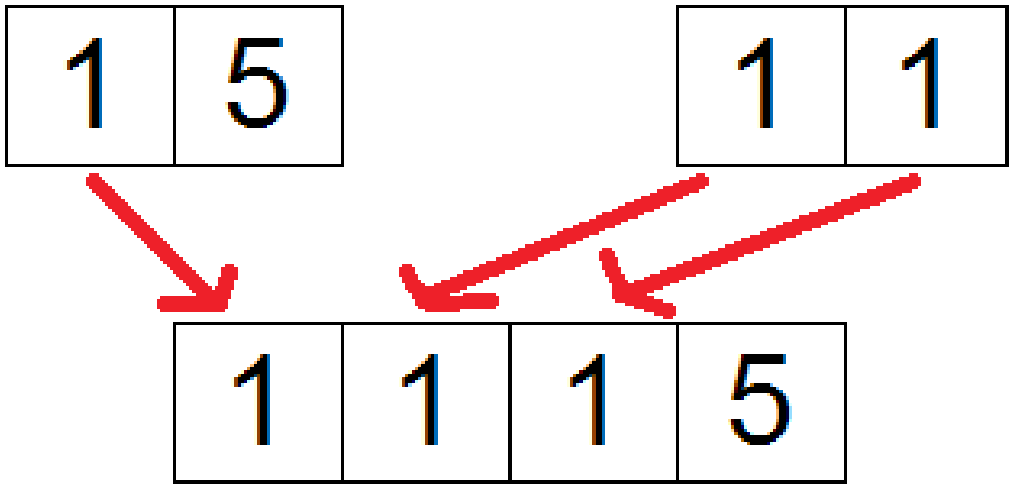


Worst-Case

- O(n log n)
- Maximale vergleiche beim Mergen

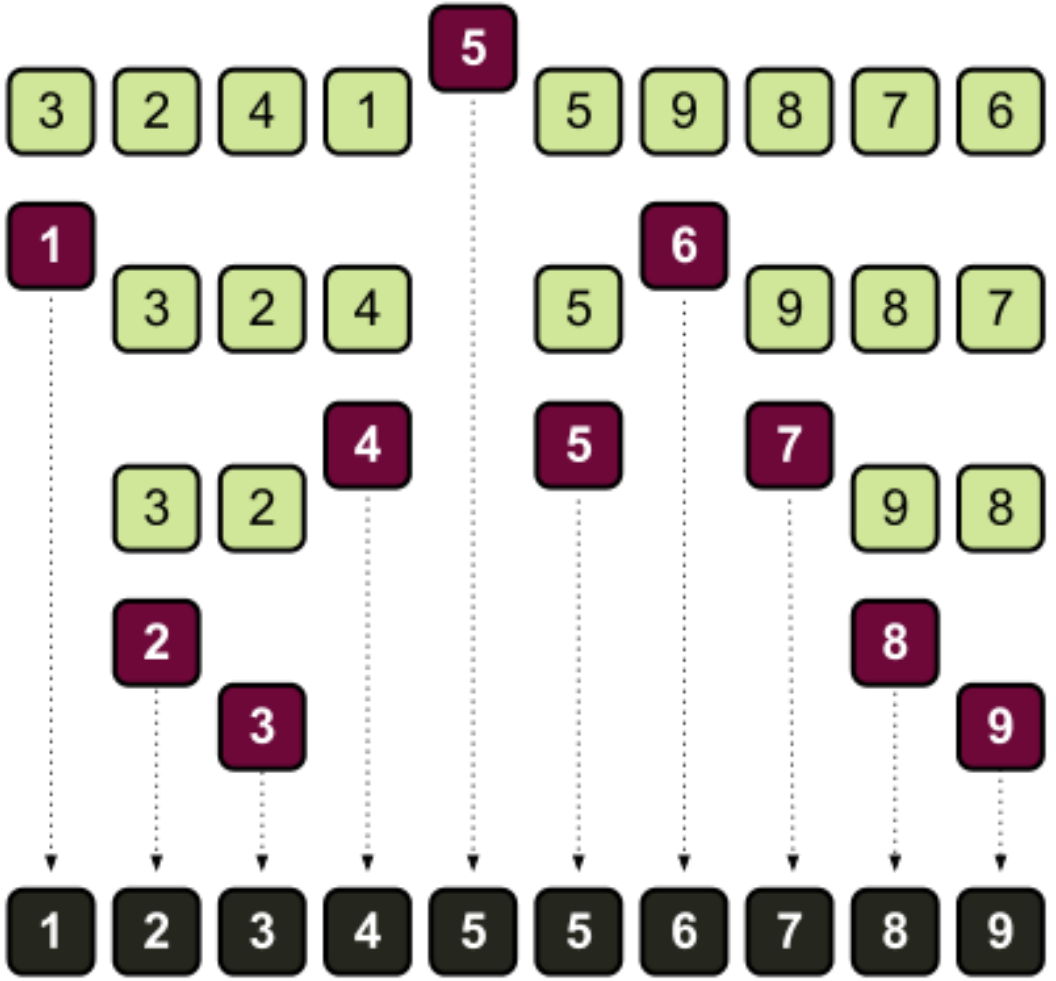


Stabilität & In-Place



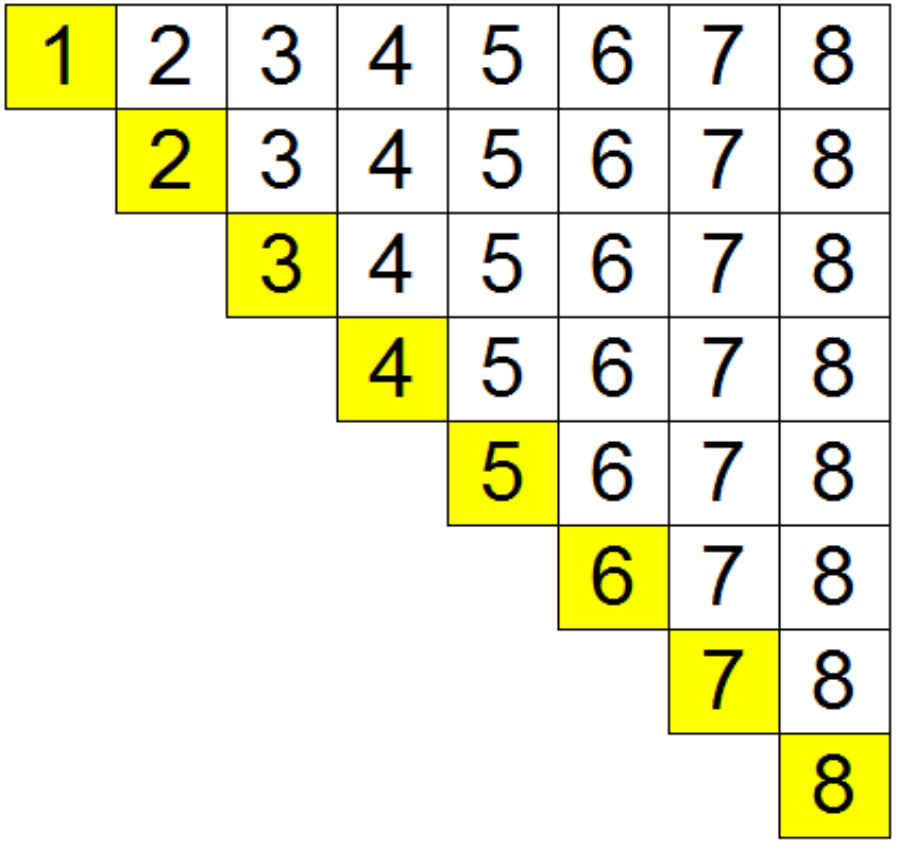
- Stabil: Ja
- In-Place: Nein
- Benötigt zusätzlich O(n) Hilfsspeicher

QuickSort



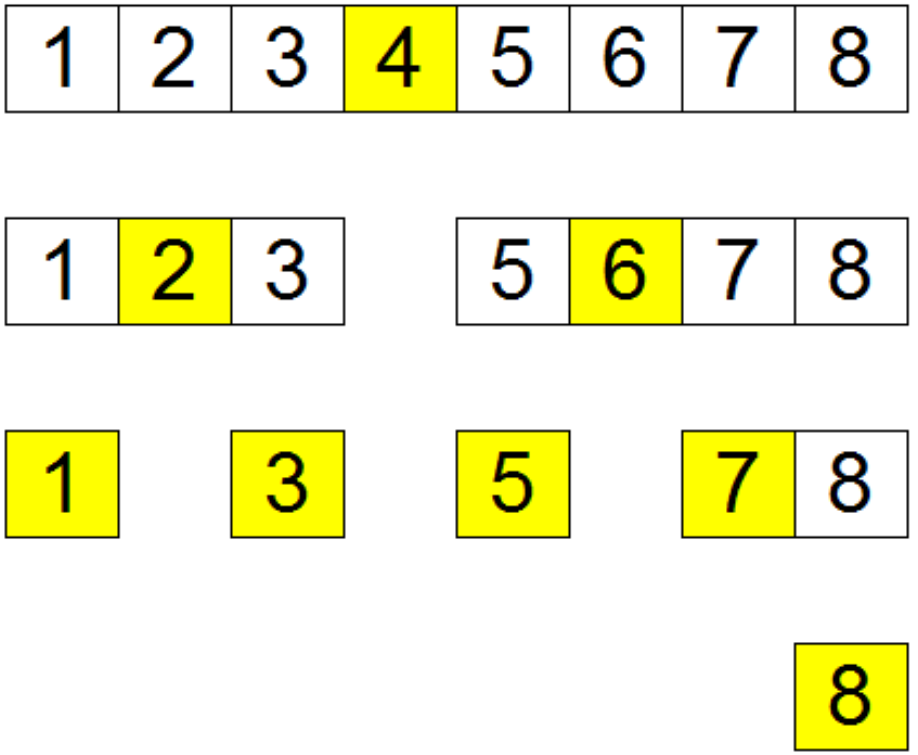
Worst-Case

Worst-Case $O(n^2)$



Best Case

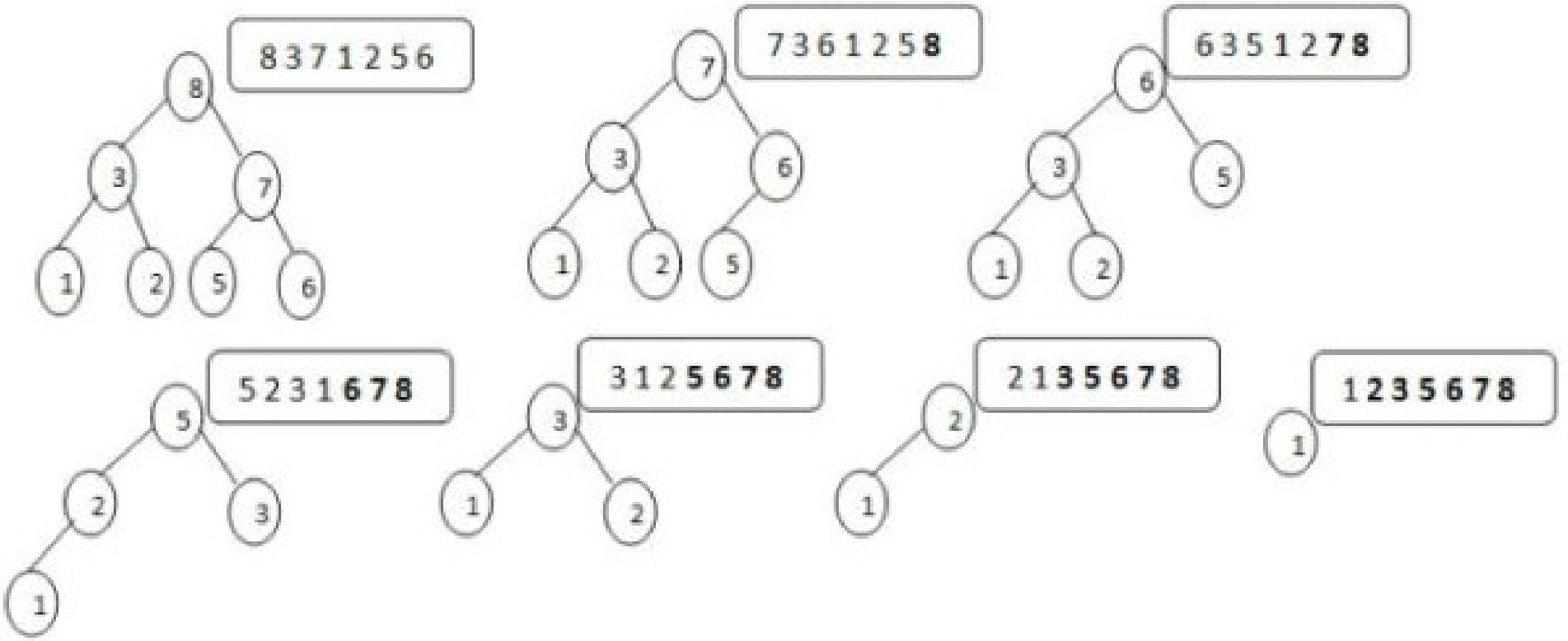
Best-Case O(n log n)



Stabil / In-Place

- Stabil: Nein
- In-Place
- Hilfsspeicher im Worst-Case O(n)

HeapSort


8371256
8
3
7
1
2
5
6
7361258
7
3
6
1
2
5
6351278
6
3
5
1
2
5231678
5
2
3
1
3125678
3
1
2
2135678
2
1
1235678
1

Worst-Case

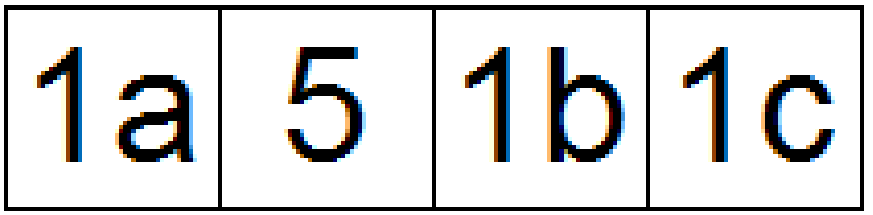
- O(n log n)
- Heapify: maximale vertauschungen
- z.B. nächst großes Element ist im Baum unten

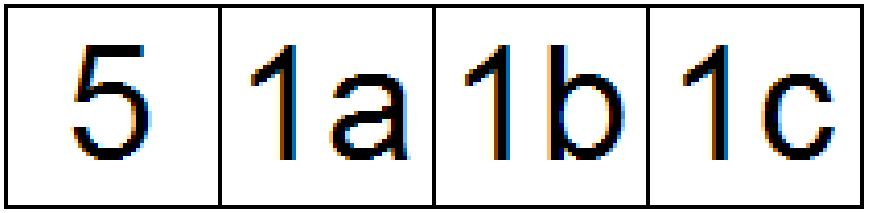
Best-Case

- O(n log n)
- Heapify: wenig vertauschungen
- Liste mit vielen gleichen Werten

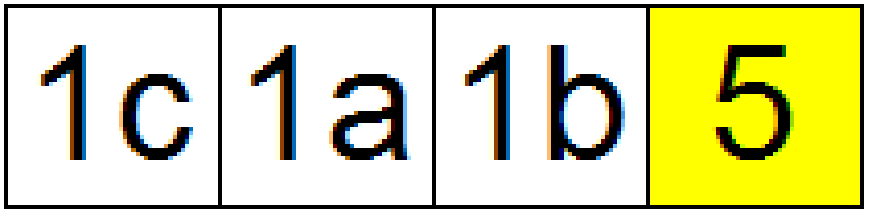
Stabil / In-Place

- Stabil: Nein
- Wurzel mit letzter Stelle vertauschen





- In-Place: Ja
- Hilfsspeicher O(1)



Evaluation

Zeit in ms	BubbleSort	SelectionSort	InsertionSort	QuickSort	MergeSort	Heapsort
Zufall(100k)	34793,6	14644	12765,4	15,8	46,4	31,4
aufsteigend(100k)	<1	14640,6	<1	6,4	31,2	34
absteigend(100k)	32427,8	15290,6	30072,2	9,2	31,2	40,8
aufsteigend, kleinste am Ende(100k)	13956,4	14553	<1	<1	34,4	31
gleiche werte(100k)	<1	14687	<1	6,4	31,2	<1

Zeit in ms	BubbleSort	SelectionSort	InsertionSort	QuickSort
absteigend(10k)	465,6	137,4	259,2	218,8

Fazit

- **Quadratische**
- Bei kleinen Werten
- Zu hohe Laufzeit

QuickSort

- Schneller als MS
- Worst-Case $O(n^2)$

- **MergeSort**
- Stabil
- Hoher Speicherplatzbedarf

- **HeapSort**
- Langsamer als MS
- Konstante Ergebnisse
- Hilfsspeicher von $O(1)$