ADP - Referat

Andre Brand

naives vs. komplexes Sortieren

naive Sortieralgorithmen

- InsertionSort
- SelectionSort

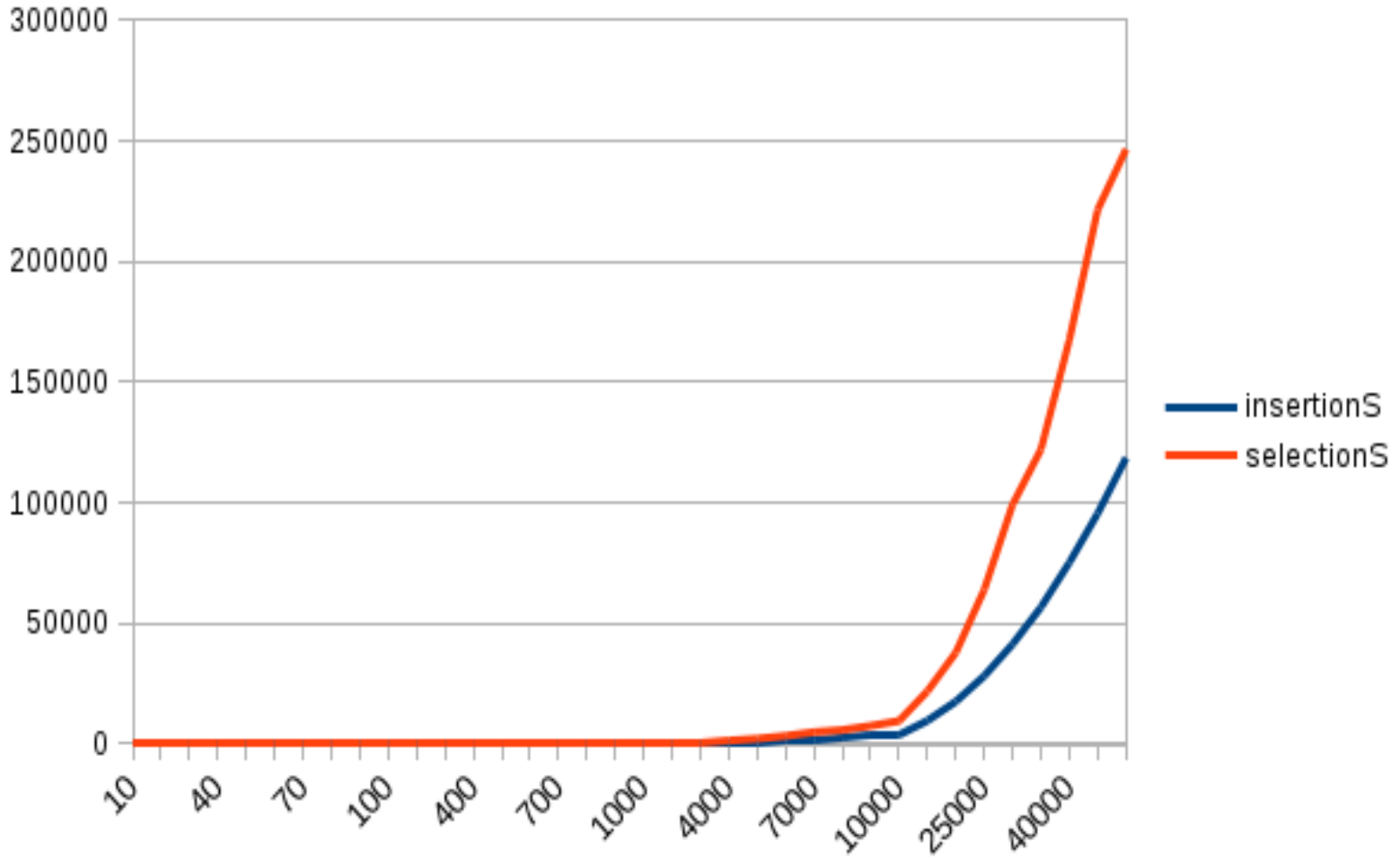
InsertionSort

- Entnahme des ersten Elements der unsortierten Liste
- Sortierung erfolgt beim Einfügen in neue Liste

SelectionSort

- Kleinstes Element wird aus der Liste entnommen und am das Ende des sortierten Bereiches gelegt
- Wiederholung mit dem Rest der unsortierten Liste bis kein Element mehr im unsortierten Bereich vorhanden ist

Vergleich


300000
250000
200000
150000
100000
50000
0
10
40
70
100
400
700
1000
4000
7000
10000
25000
40000
insertionS
selectionS

Fazit

- InsertionSort zumindest in meiner Implementierung deutlich effektiver

komplexe Sortieralgorithmen

- MergeSort
- QuickSort

MergeSort

- Die Liste wird in der Mitte in zwei Hälften geteilt
- Die beiden Hälften der Liste durchlaufen die selbe Prozedur, bis in jeder Liste nur noch maximal ein Element vorhanden ist
- Beim Zusammenführen der Liste werden diese sortiert

MergeSort

- Funktion Merge ist Hauptfunktion
- Ausschnitt:

```
do_merge(Merged, [LeftH | LeftT], [RightH | RightT]) ->
  if
    LeftH =< RightH ->
      do_merge([LeftH | Merged], LeftT, [RightH | RightT]);
    true ->
      do_merge([RightH | Merged], [LeftH | LeftT], RightT)
  end.
```

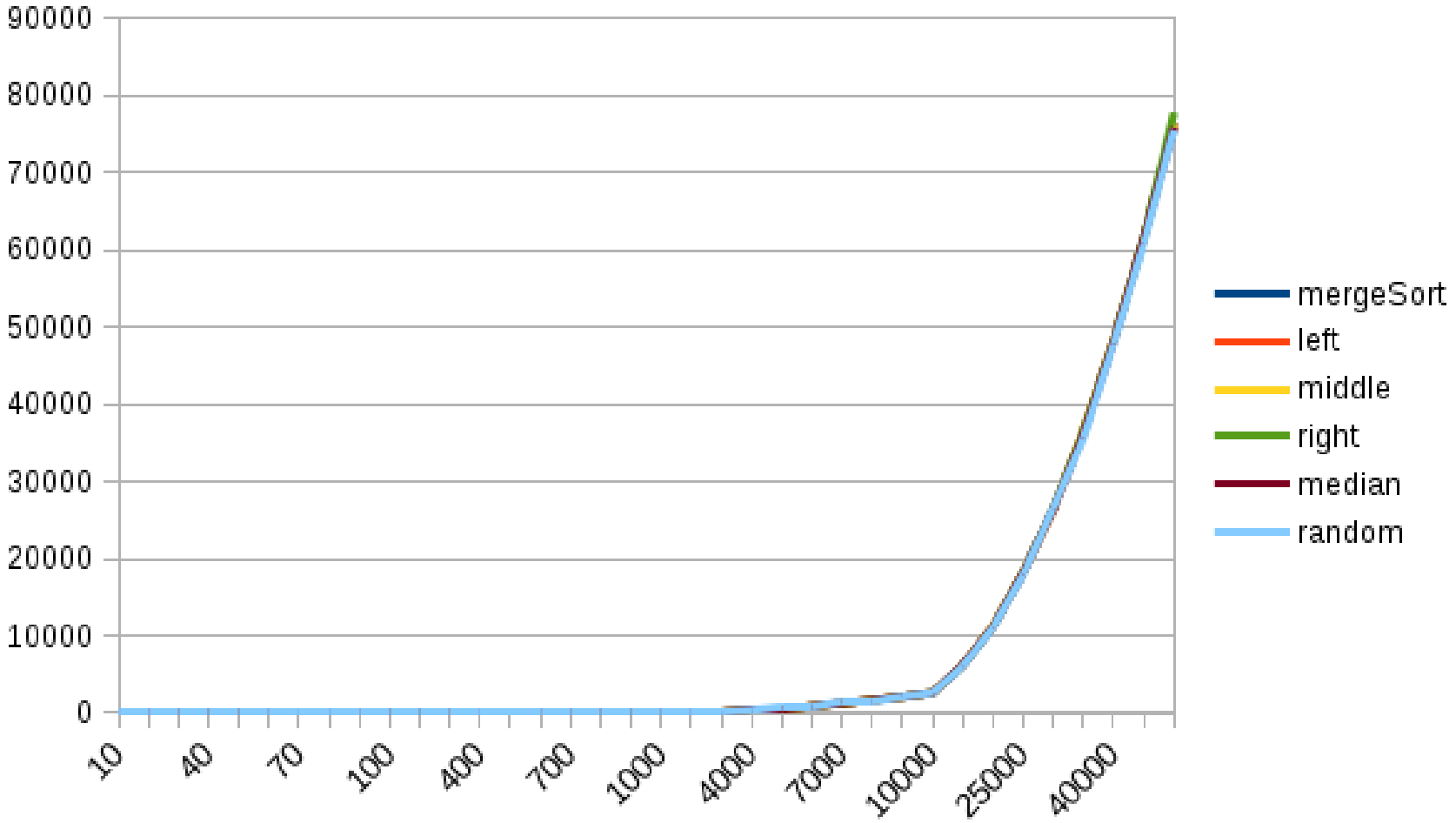
QuickSort

- Aufteilen der Liste anhand eines Pivot-Elements
- Kleinere Elemente werden links des Pivot-Elements gelegt, größere nach rechts
- Gleiches Vorgehen für die Teil-Listen links und rechts des Pivot-Elements, bis Teilliste maximal ein Element beinhaltet
- Unterschiedliche Elemente können als Pivot-Element genutzt werden

QuickSort

- Aufteilung der Liste mit zwei Funktionen
- Elemente, die gleich dem Pivot-Element sind, werden in einer dritten Liste gespeichert
  - Sonst Gefahr einer Endlosschleife
- Eigene Funktion zur Bestimmung des Pivot-Elements

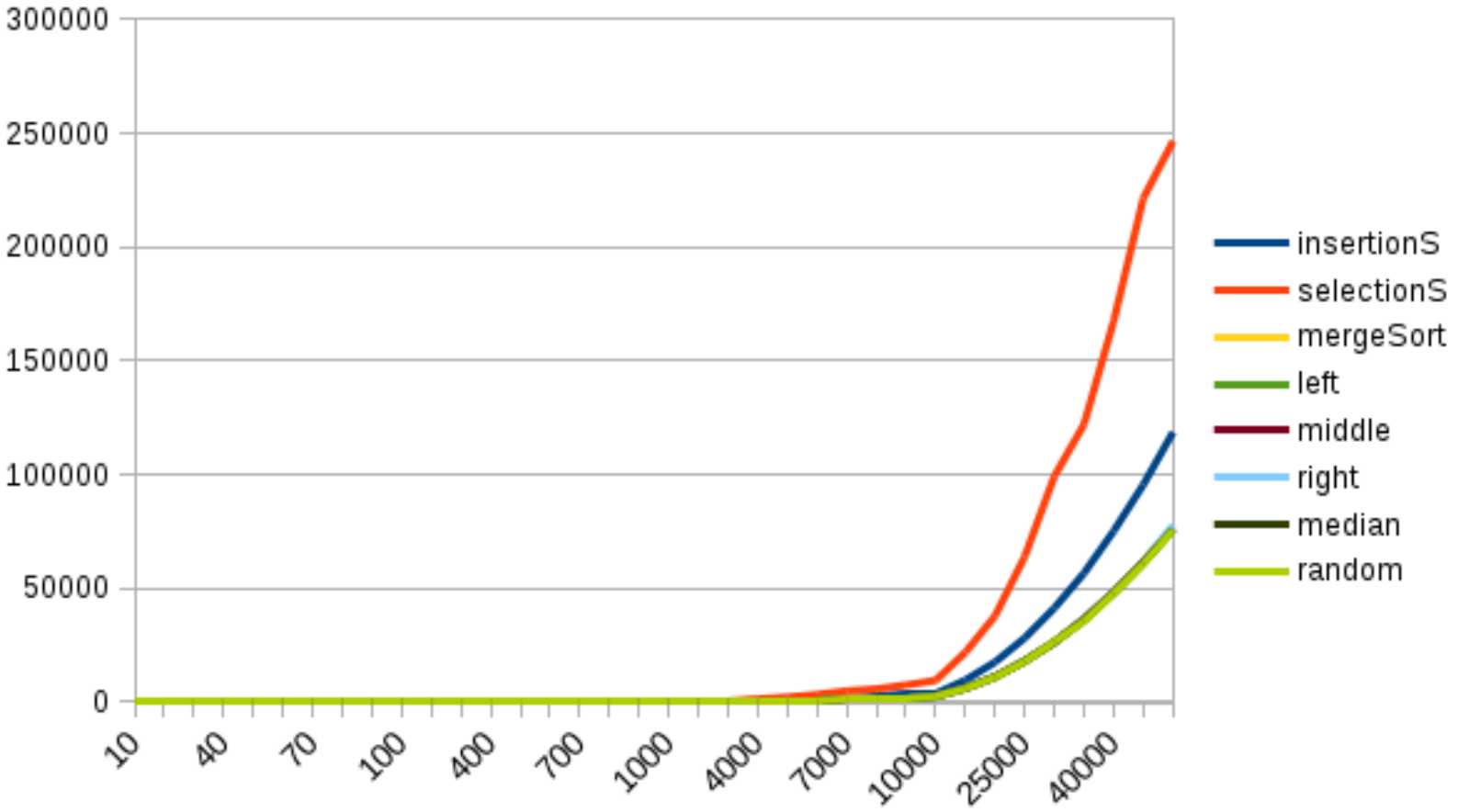
Vergleich


90000
80000
70000
60000
50000
40000
30000
20000
10000
0
10
40
70
100
400
700
1000
4000
7000
10000
25000
40000
mergeSort
left
middle
right
median
random

Fazit

- Alle Algorithmen sehr ähnlich im Zeitaufwand
- Auch wenig Unterschiede mit unterschiedlichen Pivot-Elementen bei QuickSort

Vergleich komplex und naiv

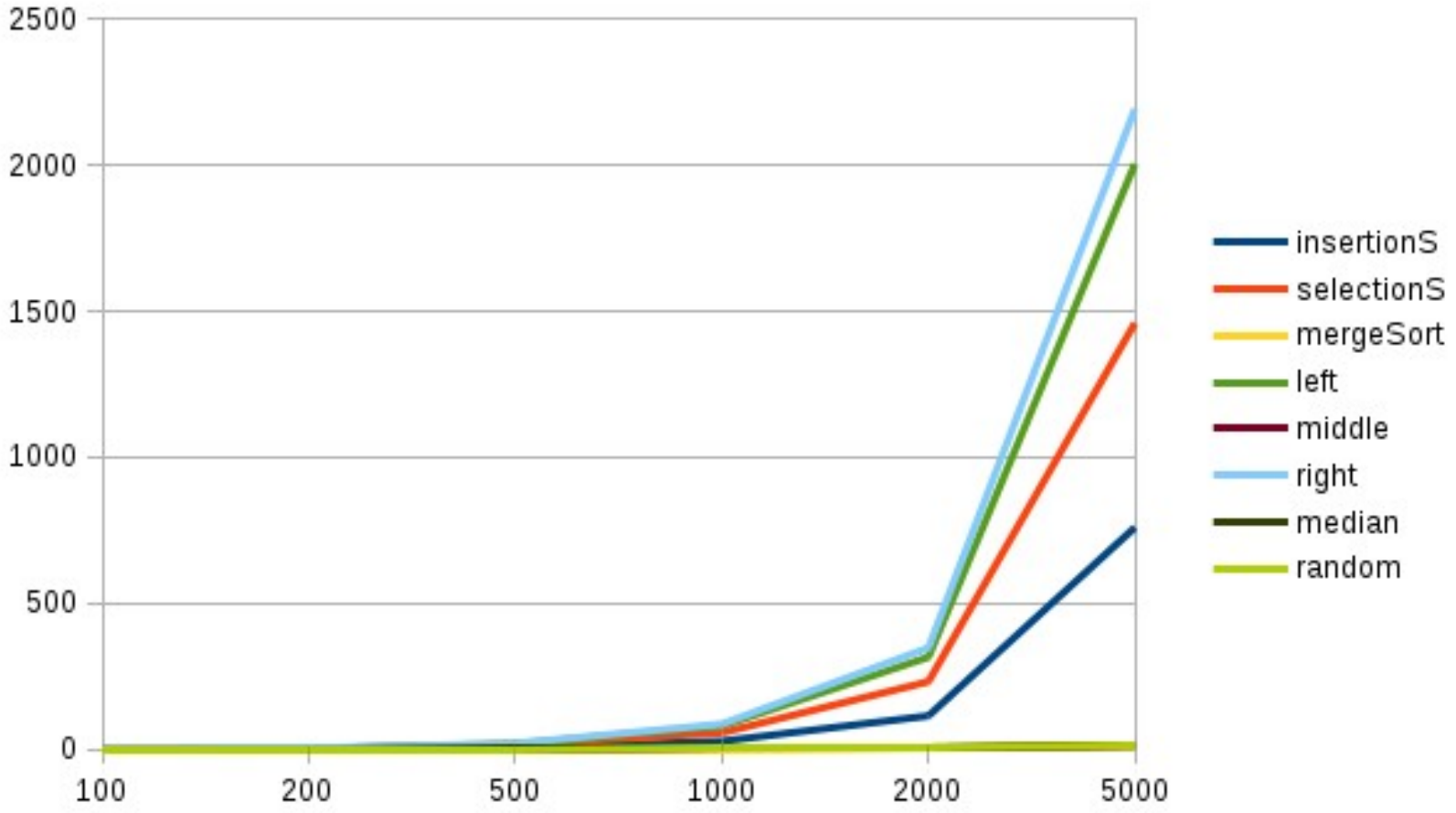


	Sortieralgorithmen / Angabe der Zeit in Millisekunden							
				quickSort				
Anzahl Elemer	insertionS	selectionS	mergeSort	left	middle	right	median	random
10	5	0	2	0	0	0	0	0
20	0	0	0	0	0	1	0	0
30	0	0	0	0	1	0	0	0
40	0	0	0	1	0	0	0	0
50	0	1	0	0	0	1	0	0
60	0	1	0	0	0	1	0	0
70	0	1	0	0	1	0	0	0
80	0	1	0	0	1	0	1	0
90	0	1	0	1	0	1	0	1
100	1	1	0	1	0	1	1	1
200	2	4	1	2	1	2	2	2
300	4	8	3	3	3	3	2	4
400	6	13	4	4	5	4	5	5
500	11	19	6	6	7	6	6	8
600	14	29	9	8	9	8	9	8
700	18	39	12	12	11	12	12	13
800	24	57	14	15	15	15	15	16
900	32	75	18	18	20	19	19	19
1000	38	84	22	24	23	23	23	28
2000	152	351	93	91	92	91	90	99
3000	334	786	208	205	205	209	210	211
4000	626	1518	386	380	373	378	383	388
5000	997	2247	622	632	631	618	630	662
6000	1436	3413	882	885	903	885	899	964
7000	1986	4976	1284	1256	1253	1236	1263	1329
8000	2701	5845	1657	1670	1658	1654	1646	1632
9000	3275	7598	2030	2059	2045	2101	2088	2104
10000	4087	9609	2552	2593	2606	2558	2568	2575
15000	9727	21950	6112	6148	6168	6151	6206	6038
20000	17632	37910	11196	11234	11267	11122	11114	11020
25000	28214	63889	18006	18104	18106	18224	18124	17844
30000	41422	99244	26371	26203	26420	26458	26382	26525
35000	56708	122070	36541	36331	36602	36434	36116	35543
40000	75235	167464	48271	48386	48232	48306	48114	47783
45000	95574	221475	61486	61373	61390	61587	61462	60744
50000	118622	246518	76522	76396	76432	77754	75850	75539

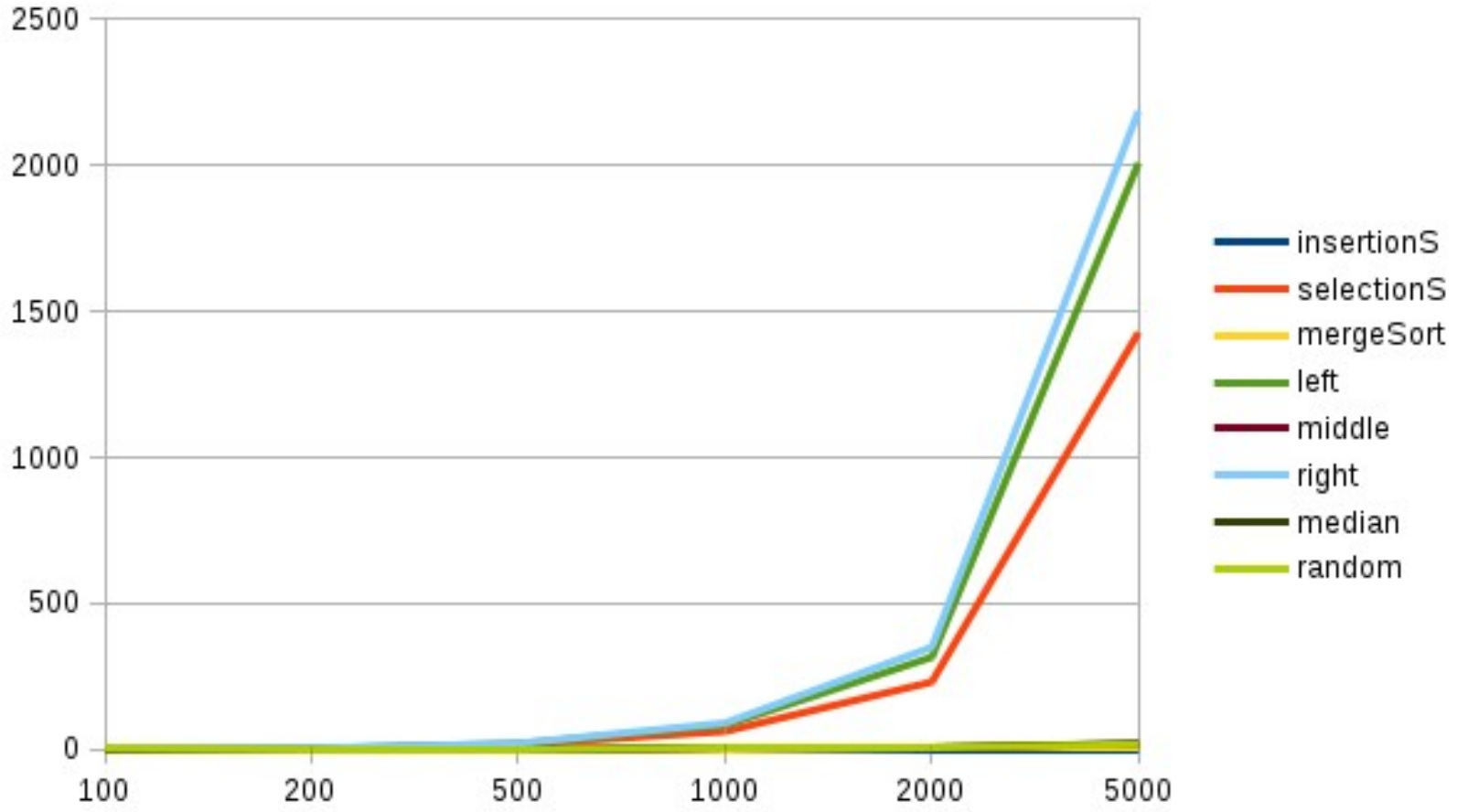
Fazit

- Komplexe Sortieralgorithmen gerade bei Listen ab einer Länge von 200 Elementen deutlich effektiver
- Davor kaum messbare zeitliche Unterschiede

Vergleich sortierte Listen



Vergleich gedrehte Listen



Sortiert	Sortieralgorithmen / Angabe der Zeit in Millisekunden							
				quickSort				
Anzahl Elemente	insertionS	selectionS	mergeSort	left	middle	right	median	random
100	1	1	0	1	0	1	1	0
200	1	3	0	3	0	4	1	0
500	6	17	1	19	1	21	2	1
1000	28	57	1	80	2	87	3	3
2000	115	232	5	316	5	348	6	6
5000	760	1460	11	2005	14	2193	15	15

Gedreht	Sortieralgorithmen / Angabe der Zeit in Millisekunden							
				quickSort				
Anzahl Elemente	insertionS	selectionS	mergeSort	left	middle	right	median	random
100	2	0	3	1	0	1	0	3
200	0	3	0	4	0	4	1	1
500	0	17	1	20	2	21	2	2
1000	0	62	2	83	3	91	3	3
2000	0	231	4	316	5	350	6	5
5000	0	1428	11	2008	19	2186	20	17

Mit Duplikaten	Sortieralgorithmen / Angabe der Zeit in Millisekunden							
				quickSort				
Anzahl Elemente	insertionS	selectionS	mergeSort	left	middle	right	median	random
100	0	1	0	0	1	0	0	0
200	0	3	0	1	0	1	0	1
500	4	17	2	1	1	1	2	1
1000	14	63	2	2	3	2	3	3
2000	64	237	4	5	6	5	6	6
5000	386	1458	12	13	19	15	16	16

Tests

- Jede Liste auf zwei Arten getestet
- Liste ohne doppelte Elemente
  - Sortierte Liste mit shuffle mischen
  - Gemischte Liste sortieren und mit sortierter Liste vergleichen
- Liste mit doppelten Einträgen
  - Vergleich von zwei Sortieralgorithmen

Fragen?