3AA

Sortieren
Direkte Sortierverfahren

30.01.06

Prof. Dr. Wolfgang P. Kowalk
Universität Oldenburg

WS 2005/2006

Laufzeitkomplexität

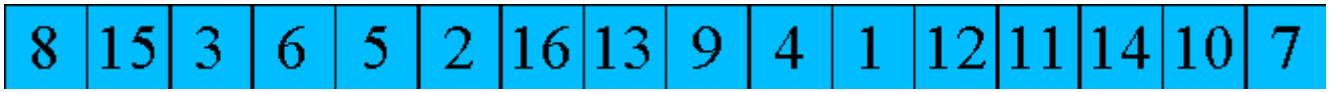
- Hängt die Bearbeitungszeit $Z(N)$ eines Problems von Komplexitätsparameter N ab, so dass für ein $N_0 > 0$ und c, $C > 0$ gilt:
  - $c \cdot f(N) \leq Z(N) \leq C \cdot g(N)$ für alle $N > N_0$, $C > 0$, $c > 0$,
- so wird $Z(N)$ bezeichnet als
  - „$Z(N)$ ist nach unten durch o($f(N)$) und nach oben durch O($g(N)$) beschränkt“
- O($g(N)$) heißt „Von der Ordnung $g(N)$“

Laufzeitkomplexität

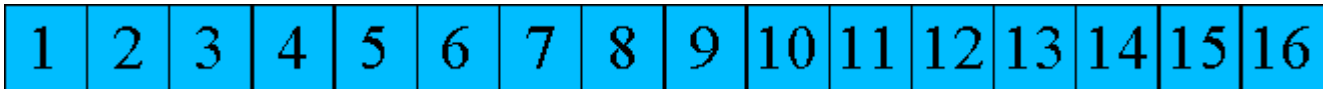
- Spezielle Bezeichnungen
  - O(1) oder **konstant**, falls $g(N) = 1$ eine Konstante ist;
  - O(N) oder **linear**, falls $g(N) = N$;
  - O(N^2) oder **quadratisch**, falls g(N) = N^2;
  - **polynomial**, falls $g(N)$ ein Polynom in N ist;
  - O(log(N)) oder **logarithmisch**, falls g(N) = log(N);
  - O(a^N) oder exponentiell, falls $g(N) = a^N$ ist für eine Konstante $a > 1$;
  - (N·log(N)) oder „N mal log N“, falls $g(N) = N$·log(N).
- Komplexität eines Problems
  - Komplexität des „schnellsten“ Algorithmus, der dieses Problem löst.

Sortieren – Grundbegriffe

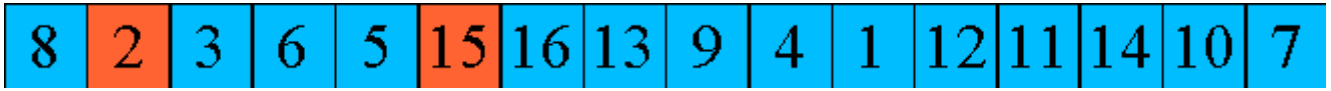
- Eine Folge von *N* Zahlen lässt sich in einem Feld abspeichern:
  - `integer array Feld [1..N];`
  - Unsortiertes Feld



  - Sortiertes Feld



- **Permutieren**
  Vertauschen zweier Elemente des Feldes



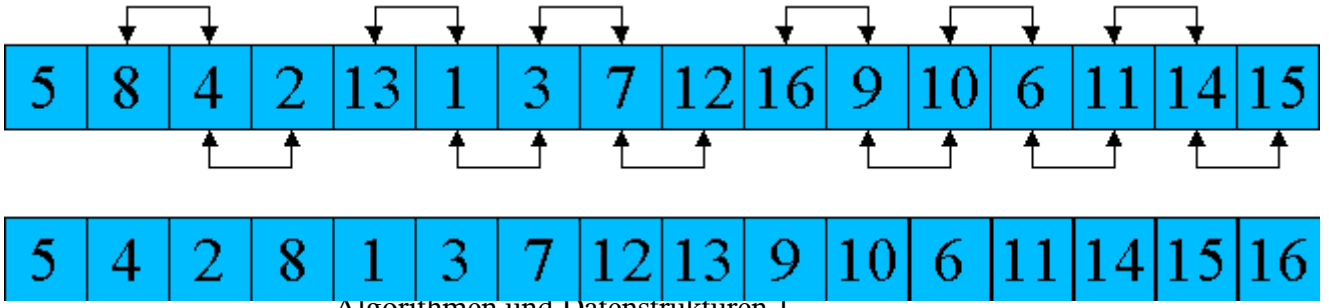
Sortieren – Grundbegriffe

- **Stabiles Sortierverfahren**
  - $i < j \ \wedge$ Feld[i] = Feld[j] $\Rightarrow \ i_S < j_S$

    Ordnung gleicher Feldelemente bleibt erhalten.
  - Wichtig für
    - Sortierung bei verschiedenen Schlüsseln (Vorname, Nachname)
    -
- **Ordnungsverträgliches Sortierverfahren**
  - Laufzeit eines Sortierverfahrens wächst monoton mit Inversionszahl jeder Permutation
  - Effizientere Verfahren sind häufig nicht oder nur teilweise ordnungsverträglich

Direkte Sortierverfahren

- Sortieren durch Vertauschen
  - Bubblesort
- Vertausche benachbarte Element, solange noch welche vertauscht sind
- Verschiedene Verbesserungsmöglichkeiten, aber
  - Bubblesort 'schlechtestes' Sortierverfahren
  - nicht einfacher zu implementieren als andere Verfahren
  - unprofessionell



Direkte Sortierverfahren

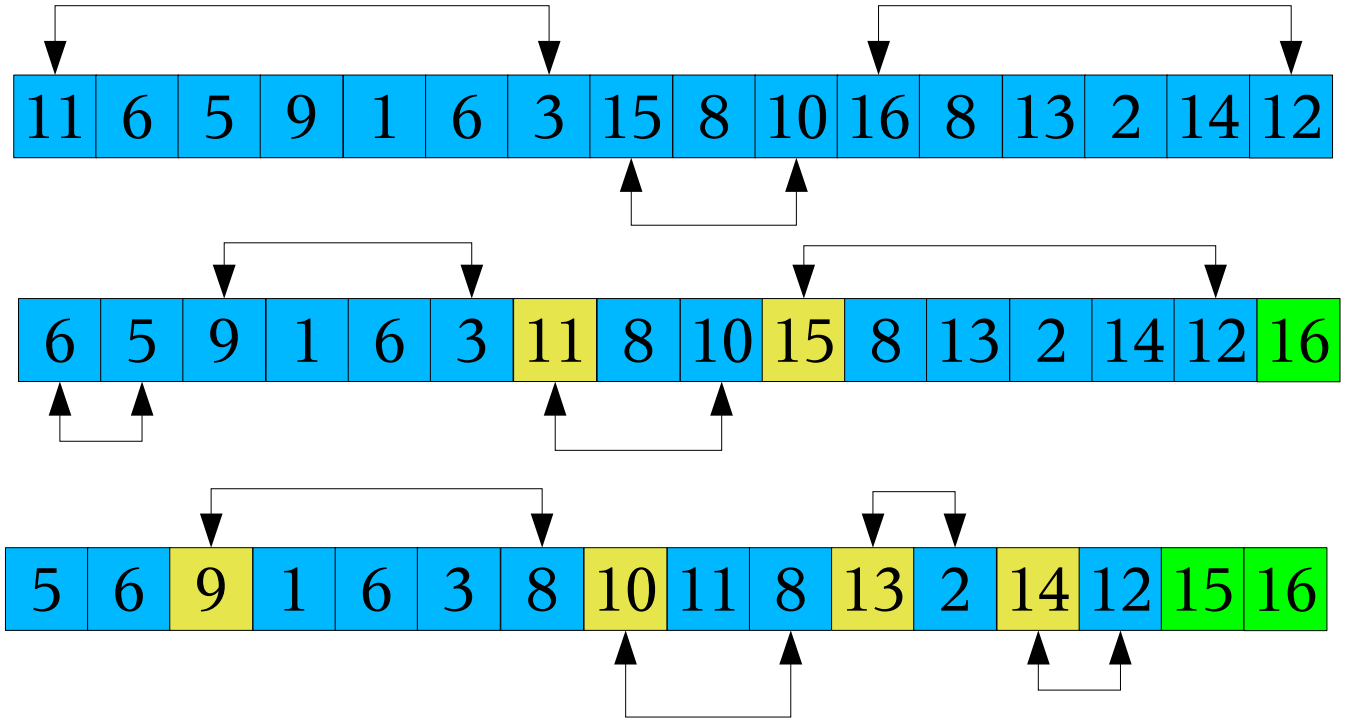
```
 1            // sortieren durch Vertauschen
 2
 3            adr ende := con feld0  // Feld [0]
 4 FuegHinzu  adr ende += con 1      // Feld [0,...,ende]
 5            if val ende >= con feldEnde then goto con Fertig
 6            adr vergleich := val ende       // vergleich = ende
 7            // vertausche bis an richtiger Stelle
 8 Vertausch  if val vergleich <= con feld0 then goto con FuegHinzu
 9            adr v1 := val vergleich - con 1 // v1 = i-1
10            // prüfe, ob an richtiger Stelle
11            if val val v1 <= val val vergleich then goto con FuegHinzu
12            adr wert1 := val val v1         // vertausche w<=[v1]
13            adr val v1 := val val vergleich // [v1] = [vergleich]
14            adr val vergleich := val wert1  // [vergleich] = w
15            adr vergleich -= con 1          // vergleich--
16            goto con Vertausch
17 Fertig     stop

26 feld0      con 12
27 feld1      con 4
...
47 feldEnde
```

Direkte Sortierverfahren

- Die Laufzeit von Bubblesort ist quadratisch
  - Vergleiche in jeder Zeile (N-1) Mal
  - Nach jedem Durchlauf liegt größtes Element an richtiger Stelle
  - Nach N-1 Durchläufen stehen alle Elemente an richtiger Stelle
  - Laufzeit: $(N\text{-}1)^2$ Vergleiche, ca. $(N\text{-}1)^2/2$ Vertauschungen
  - Laufzeitkomplexität: $O(N^2)$.

Direkte Sortierverfahren

- Verbesserungen der Laufzeit von Bubblesort
  - Verkleinere obere Grenze:
    N-1 + N-2 + N-3 +...+ 2 + 1 = $N \cdot (N\text{-}1)/2$ Vergleiche
  -



Direkte Sortierverfahren

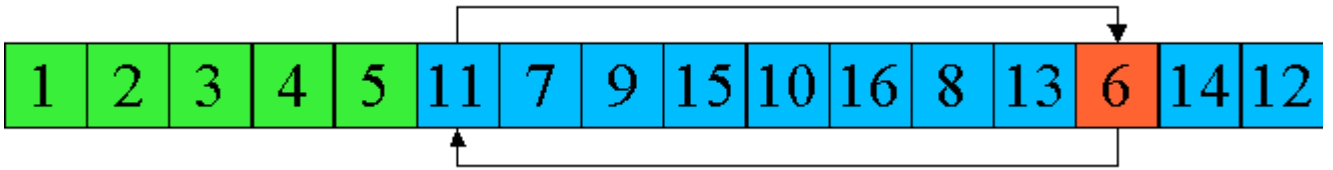
- Verbesserungen der Laufzeit von Bubblesort
  - Beende Durchlauf, wenn alle Elemente sortiert
    - meist keine Verbesserung, da zusätzlicher Overhead

- Weitere Verbesserungen
  - Statt Vertauschen: Suche größtes Element und bringe es nach oben → **Sortieren durch Auswählen**
  - Vergrößere Menge um jeweils ein Element; vertausche dieses an richtige Stelle → **Sortieren durch Einfügen**

- Alle 'direkten' Sortierverfahren haben quadratische Laufzeitkomplexität!

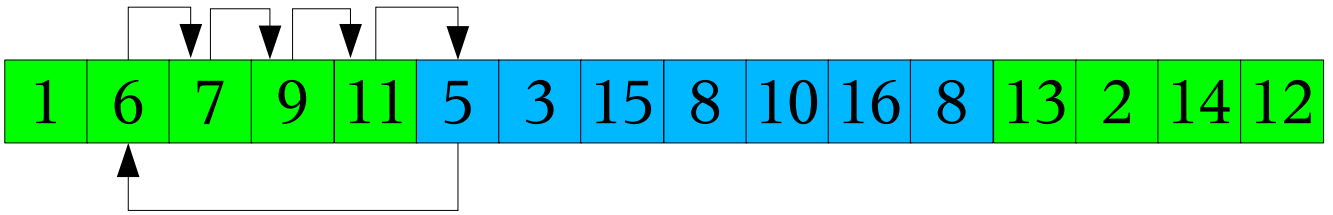
Direkte Sortierverfahren

- Sortieren durch Auswählen
  - Vertausche kleinstes Element an den Anfang der Liste
    i. Suche kleinstes Element
    ii. Vertausche kleinstes Element mit erstem Element
    iii. Verkleinere Liste um richtig positioniertes Element
    iv. Verfahre bei i. fort, bis nur noch ein Element übrig



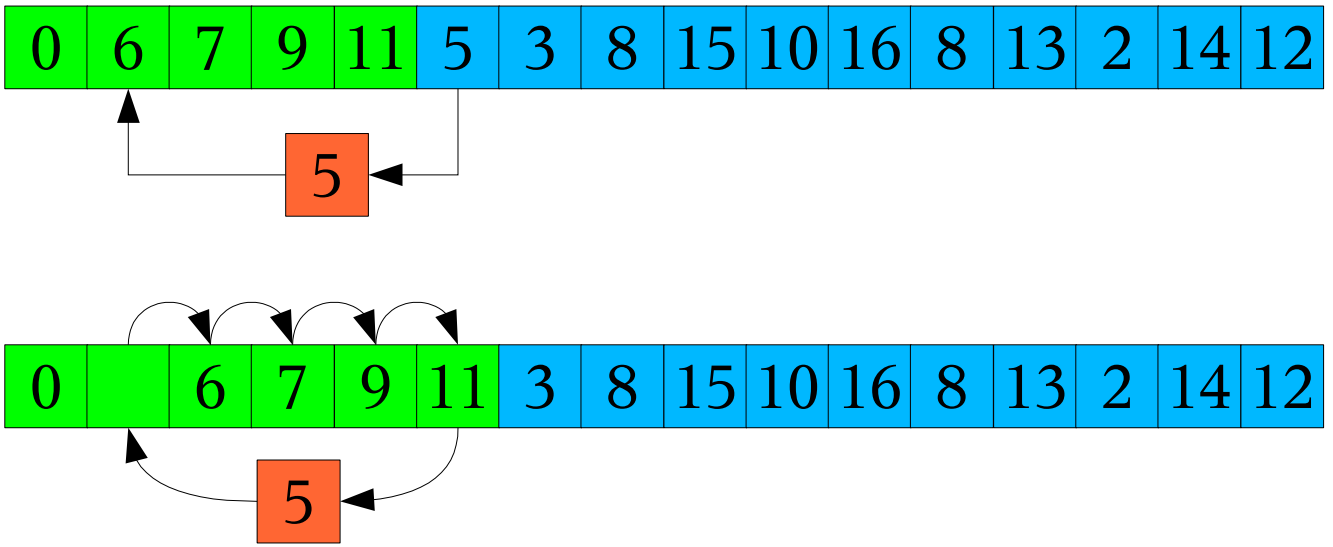
Direkte Sortierverfahren

- Sortieren durch Einfügen
  - Suche zu jedem Element den richtigen Platz
  - Verschiebe erstes unsortiertes Element nach links, bis das linke Element kleiner oder gleich diesem
    i. Sortierte Menge [1..1] ist sortiert
    ii. Füge ein Element zur Sortierten Menge hinzu
    iii. Vertausche dieses mit links benachbartem, solange bis linkes Element kleiner oder gleich diesem.



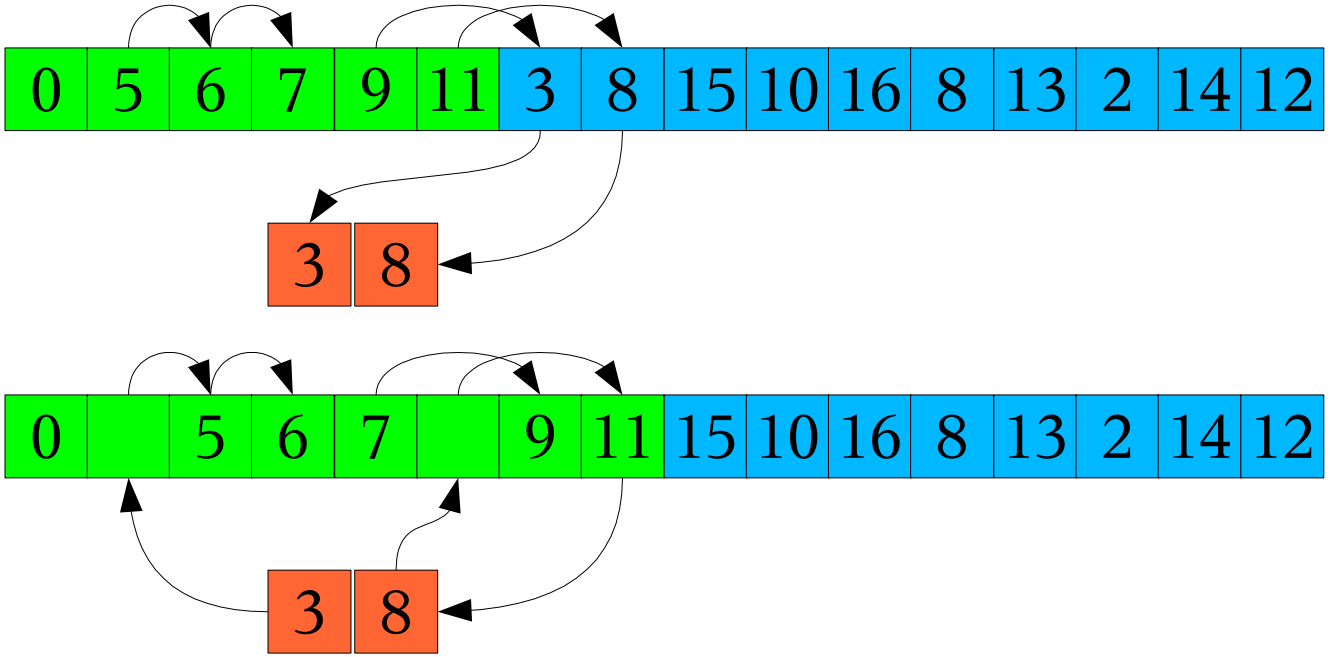
Direkte Sortierverfahren

- Sortieren durch Einfügen
  - Verbesserungsmöglichkeiten
    - Statt Vertauschen, verschiebe, bis kleineres gefunden



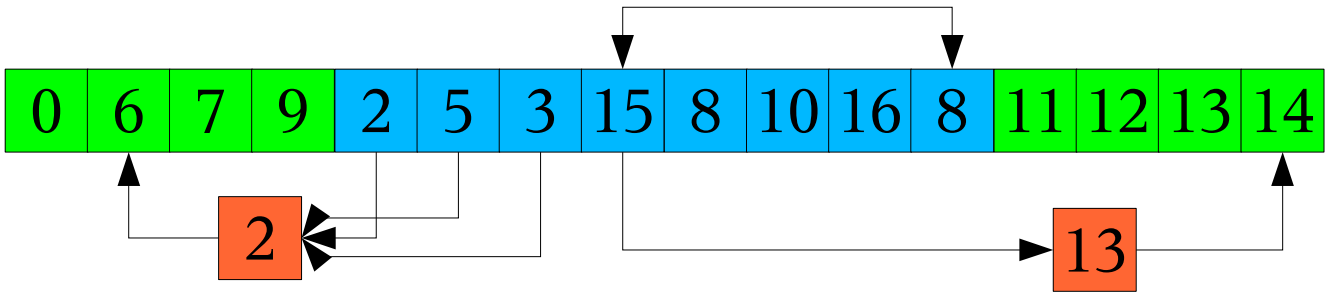
Direkte Sortierverfahren

- Sortieren durch Einfügen
  - Verbesserungsmöglichkeiten
    - Verschiebe gleichzeitig 2 (oder mehrere) Element



Direkte Sortierverfahren

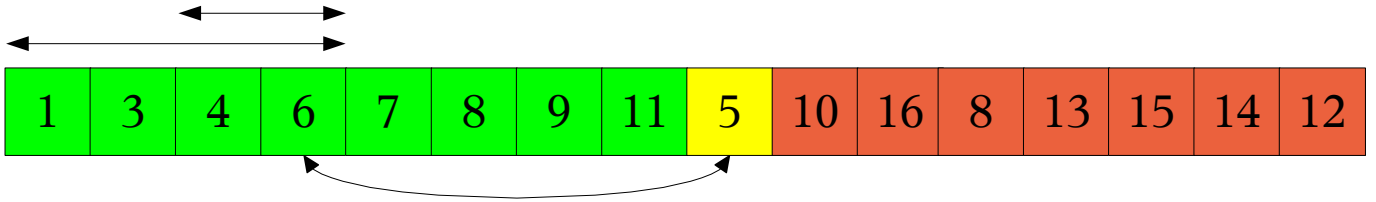
- Sortieren durch Einfügen
  - Verbesserungsmöglichkeiten
    - Bringe kleine Element nach vorne, große nach hinten
      - Pivot-Element (10)



      - Quadratische Laufzeit: $O(N^2) = 2 \times O(2 \cdot \left(\frac{N}{2}\right)^2)$

Direkte Sortierverfahren

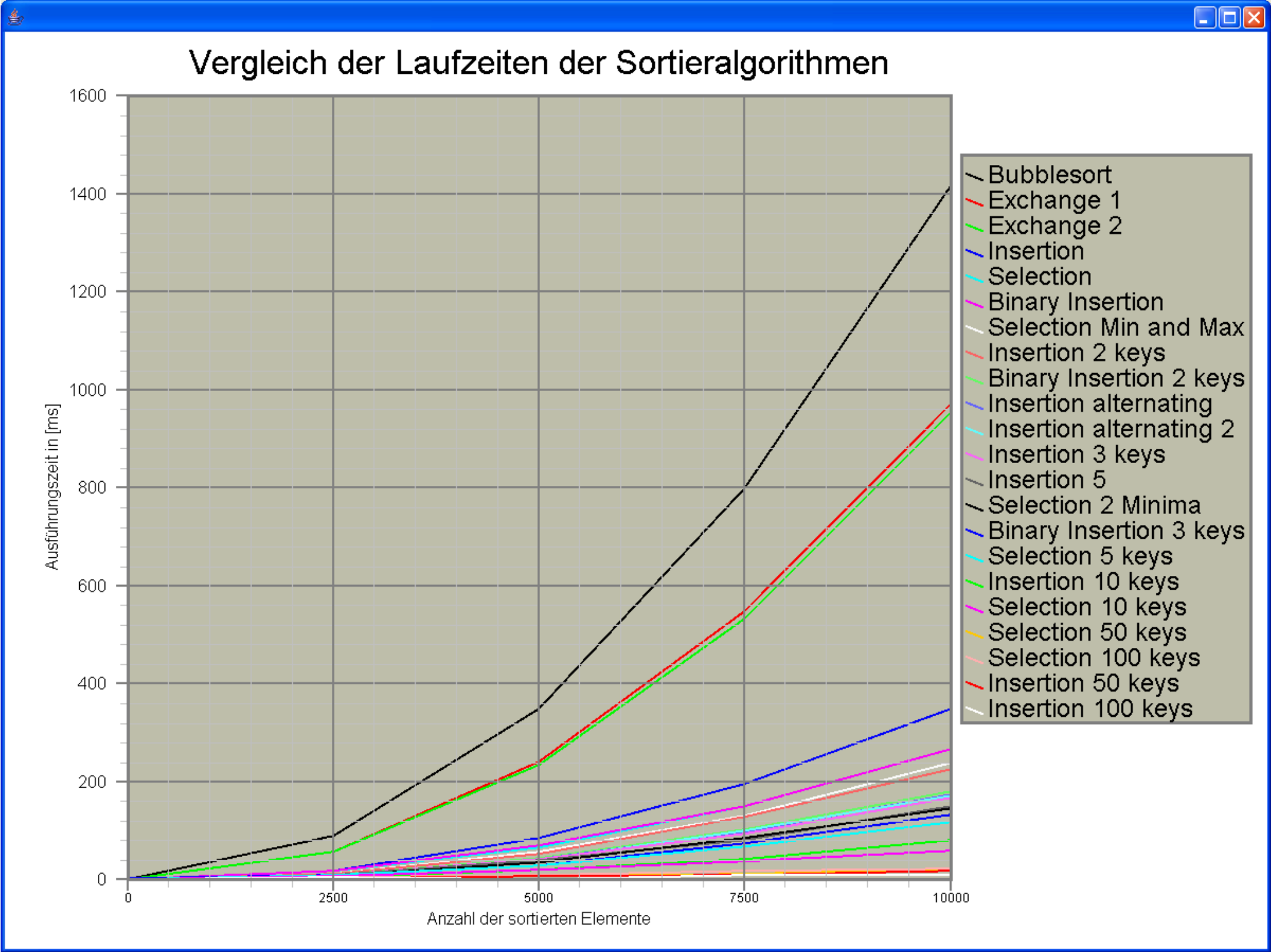
- Sortieren durch Einfügen
  - Verbesserungsmöglichkeiten
    - Kombiniere diese Verfahren
    - Suche Einfügestelle durch Binärteilung

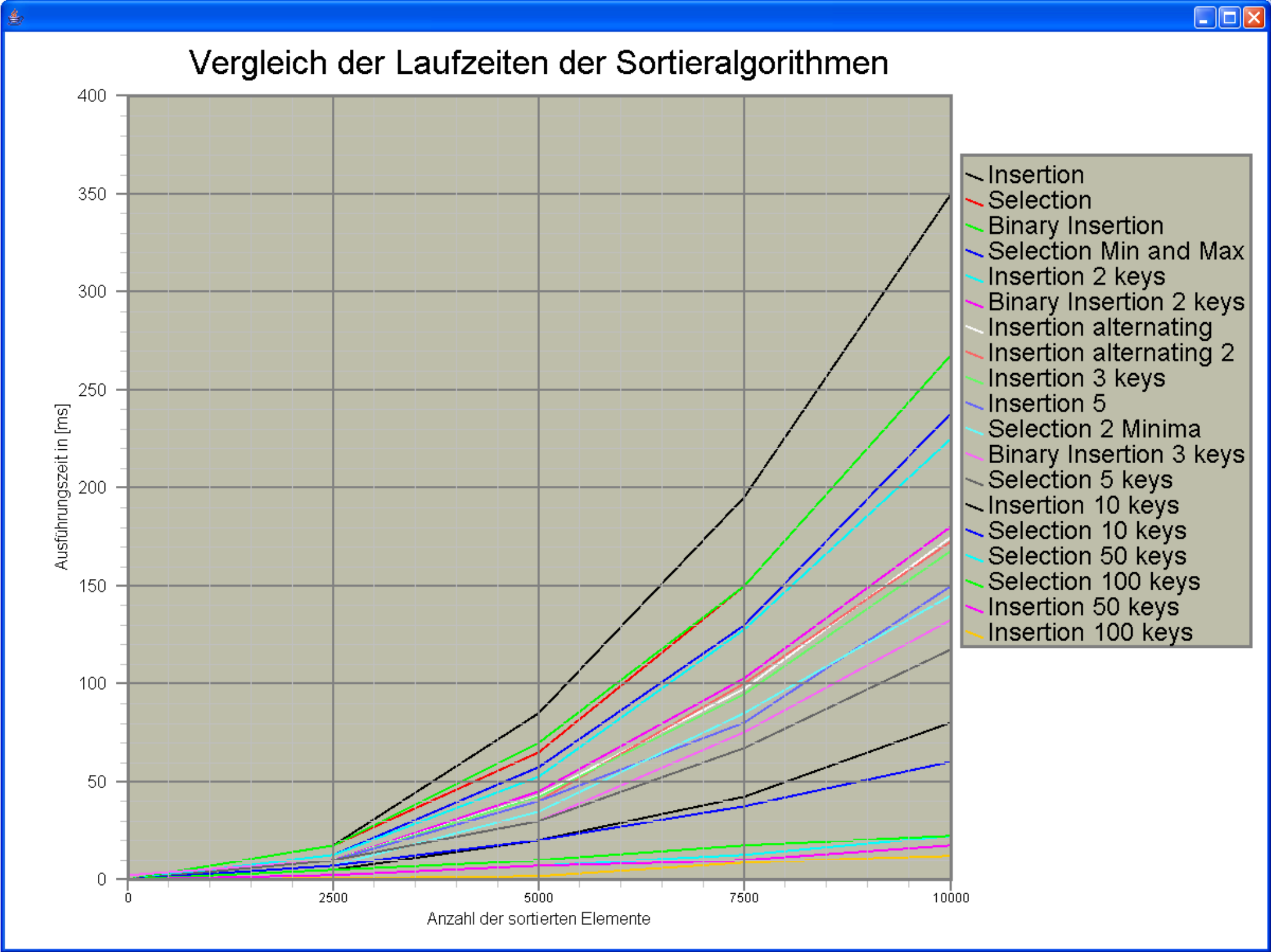


    - Verringert Zahl der Vergleiche

$$\log_2 2+\log_2 3+\ldots+\log_2(N-1)=\log_2(N-1)!<\log_2((N-1)^{N-1})=(N-1)\cdot\log_2(N-1)$$

    - Zahl der Schiebeoperationen bleibt unverändert


Vergleich der Laufzeiten der Sortieralgorithmen
Ausführungszeit in [ms]
Anzahl der sortierten Elemente
0
200
400
600
800
1000
1200
1400
1600
0
2500
5000
7500
10000
Bubblesort
Exchange 1
Exchange 2
Insertion
Selection
Binary Insertion
Selection Min and Max
Insertion 2 keys
Binary Insertion 2 keys
Insertion alternating
Insertion alternating 2
Insertion 3 keys
Insertion 5
Selection 2 Minima
Binary Insertion 3 keys
Selection 5 keys
Insertion 10 keys
Selection 10 keys
Selection 50 keys
Selection 100 keys
Insertion 50 keys
Insertion 100 keys


Vergleich der Laufzeiten der Sortieralgorithmen
Ausführungszeit in [ms]
400
350
300
250
200
150
100
50
0
0
2500
5000
7500
10000
Anzahl der sortierten Elemente
Insertion
Selection
Binary Insertion
Selection Min and Max
Insertion 2 keys
Binary Insertion 2 keys
Insertion alternating
Insertion alternating 2
Insertion 3 keys
Insertion 5
Selection 2 Minima
Binary Insertion 3 keys
Selection 5 keys
Insertion 10 keys
Selection 10 keys
Selection 50 keys
Selection 100 keys
Insertion 50 keys
Insertion 100 keys

Direkte Sortierverfahren

- Direktes Sortieren
  - Quadratischer Aufwand häufig zu groß
  - Bessere Verfahren
    - Shellsort
    - Heapsort
    - Quicksort
    - Mergesort
    - Mischen (externes Verfahren)
  - Demnächst in diesem Theater!