Script generated by TTT

Title: Täubig: GAD (31.05.2012)

Date: Thu May 31 12:39:15 CEST 2012

Duration: 36:23 min

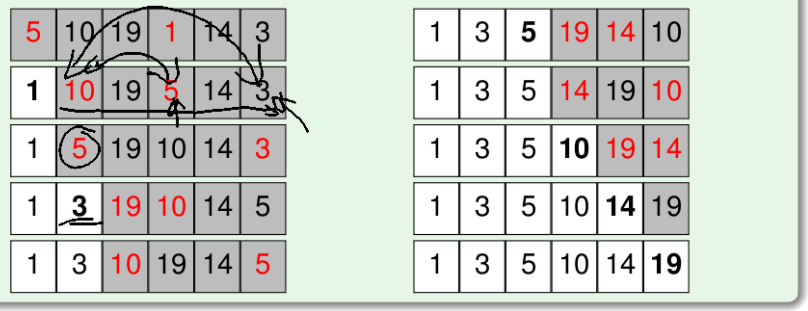
Pages: 27

SelectionSort

Sortieren durch Auswählen

Wähle das kleinste Element aus der (verbleibenden) Eingabesequenz und verschiebe es an das Ende der Ausgabesequenz

Beispiel



SelectionSort

Sortieren durch Auswählen

```
selectionSort(Element[ ] a, int n) {
  for (int i = 0; i < n; i++)
    // verschiebe min{a[i],...,a[n − 1]} nach a[i]
    for (int j = i + 1; j < n; j++)
      if (a[i] > a[j])
        swap(a[i], a[j]);
}
```

Zeitaufwand:

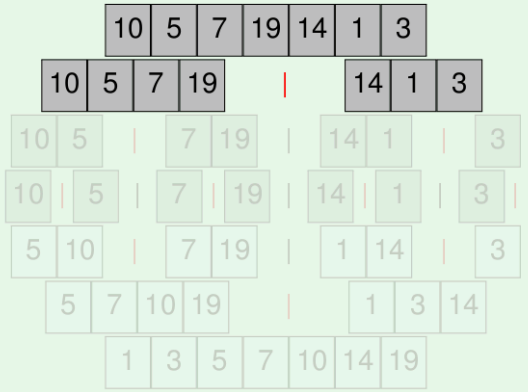
- Minimumsuche in Feld der Größe i: $\Theta(i)$
- Gesamtzeit: $\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$
- Vorteil: einfach zu implementieren
- Nachteil: quadratische Laufzeit

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel

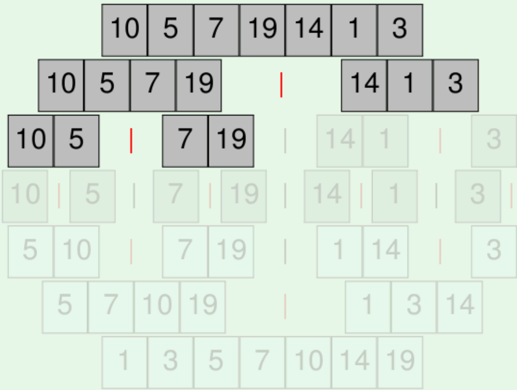


MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



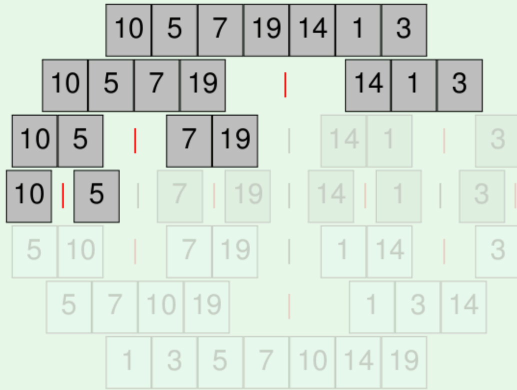
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



H. Täubig (TUM) GAD SS'12 230 / 636

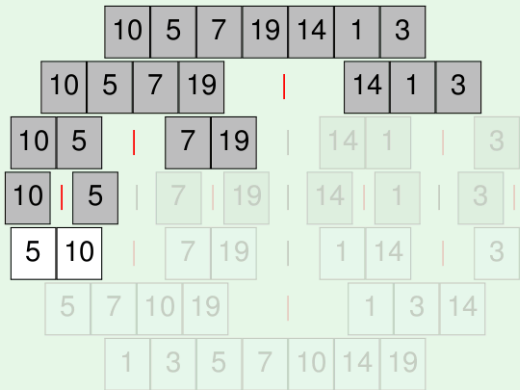
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



H. Täubig (TUM) GAD SS'12 230 / 636

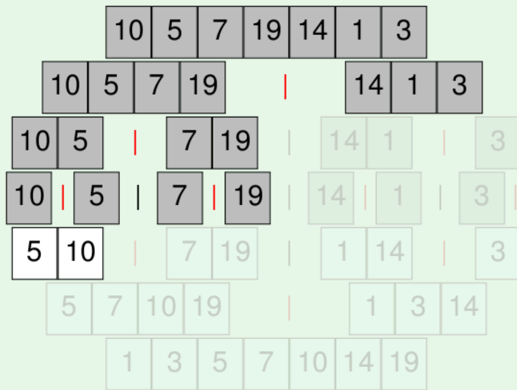
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



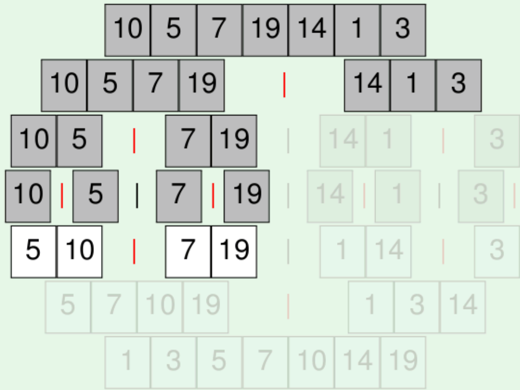
H. Täubig (TUM) GAD SS'12 230 / 636

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



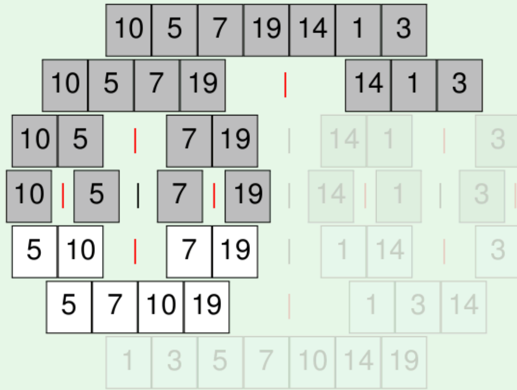
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



H. Täubig (TUM) GAD SS'12 230 / 636

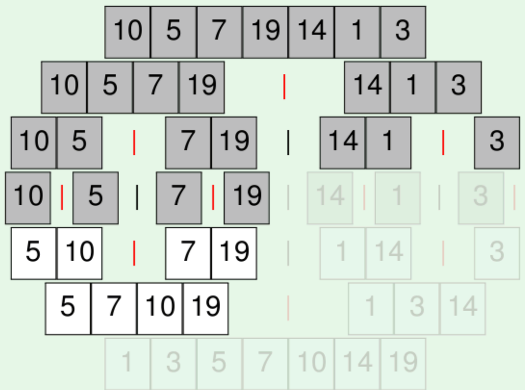
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



H. Täubig (TUM) GAD SS'12 230 / 636

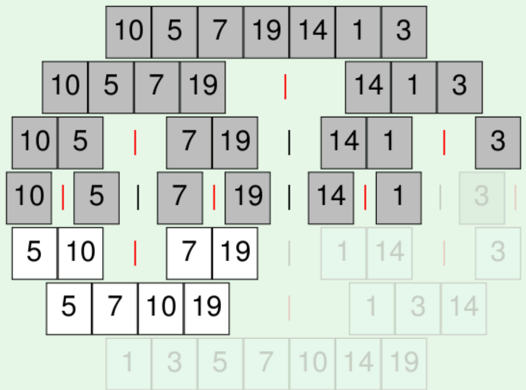
Sortieren MergeSort

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



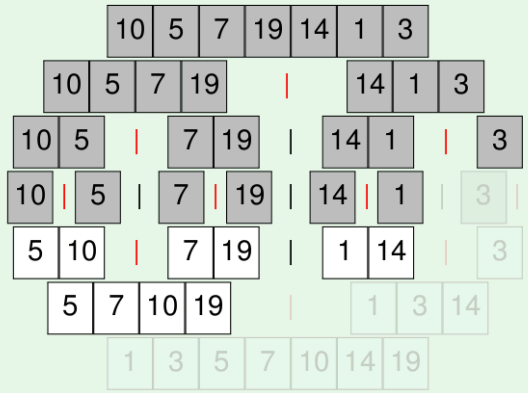
H. Täubig (TUM) GAD SS'12 230 / 636

MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel

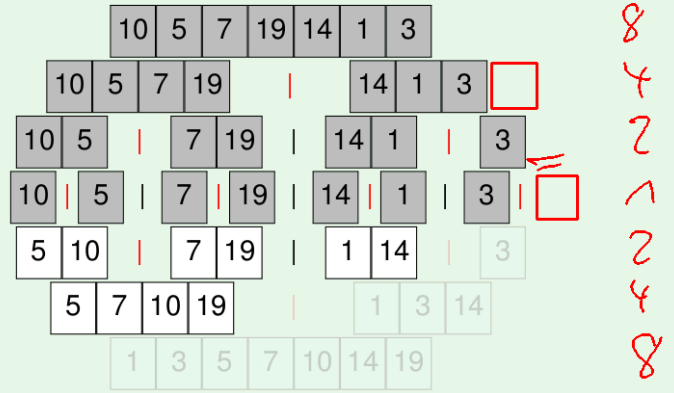


MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel

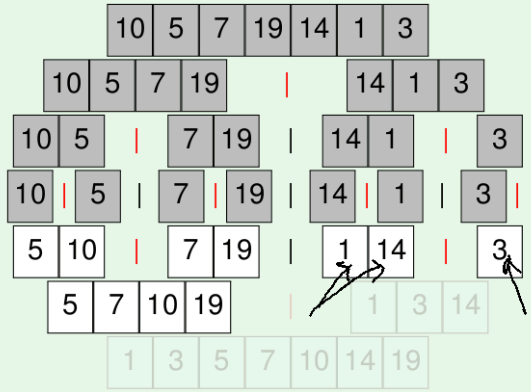


MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel

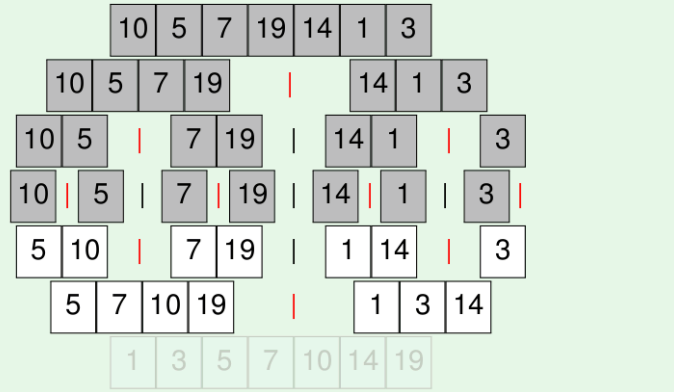


MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel

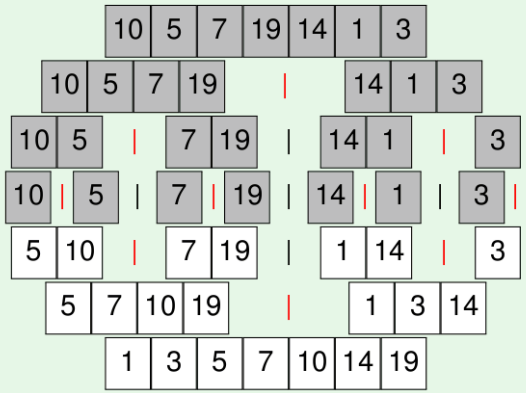


MergeSort

Sortieren durch Verschmelzen

Zerlege die Eingabesequenz rekursiv in Teile, die separat sortiert und dann zur Ausgabesequenz verschmolzen werden

Beispiel



MergeSort

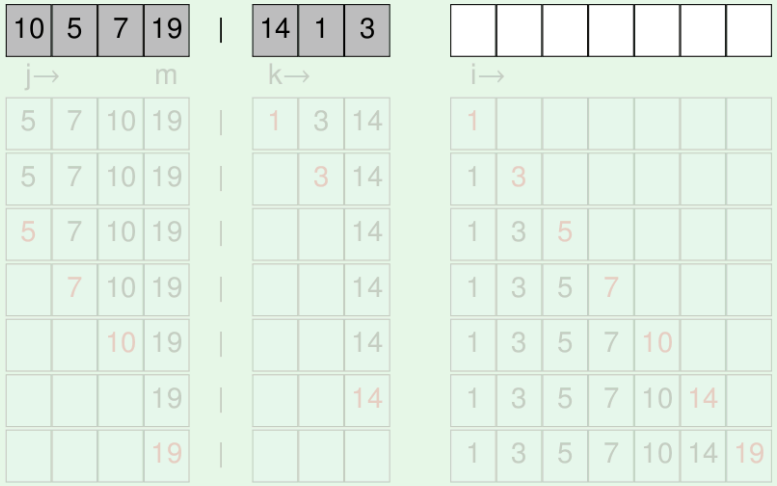
Sortieren durch Verschmelzen

```
mergeSort(Element[] a, int l, int r) {
  if (l == r) return;              // nur ein Element
  m = ⌊(r + l)/2⌋;                 // Mitte
  mergeSort(a, l, m);              // linken Teil sortieren
  mergeSort(a, m + 1, r);          // rechten Teil sortieren
  j = l;   k = m + 1;              // verschmelzen
  for i = 0 to r − l do
    if (j > m) { b[i] = a[k];  k++; }          // linker Teil leer
    else
      if (k > r) { b[i] = a[j];  j++; }        // rechter Teil leer
      else
        if (a[j] ≤ a[k]) { b[i] = a[j];  j++; }
        else { b[i] = a[k];  k++; }
  for i = 0 to r − l do   a[l + i] = b[i];     // zurückkopieren
}
```


MergeSort

Sortieren durch Verschmelzen

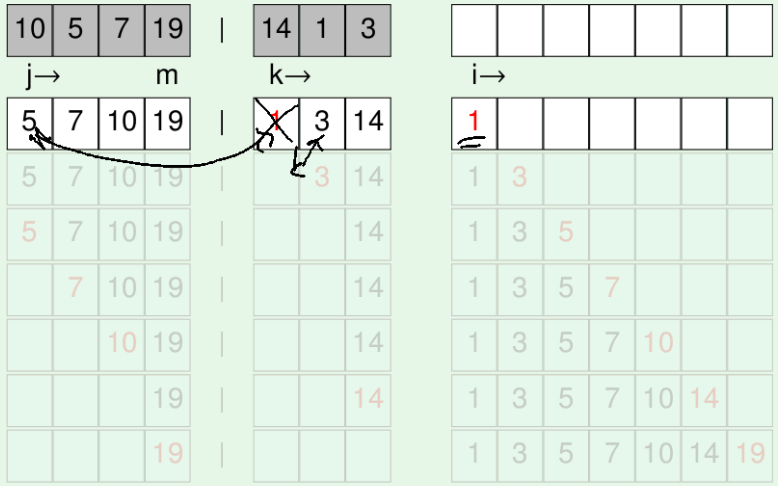
Beispiel (Verschmelzen)



MergeSort

Sortieren durch Verschmelzen

Beispiel (Verschmelzen)



MergeSort

Sortieren durch Verschmelzen

Beispiel (Verschmelzen)

10	5	7	19	|	14	1	3							
j→			m		k→			i→						
5	7	10	19	|	1	3	14	1						
5	7	10	19	|		3	14	1	3					
5	7	10	19	|			14	1	3	5				
	7	10	19	|			14	1	3	5	7			
		10	19	|			14	1	3	5	7	10		
			19	|			14	1	3	5	7	10	14	
			19	|				1	3	5	7	10	14	19

MergeSort

Sortieren durch Verschmelzen

Beispiel (Verschmelzen)

10	5	7	19	|	14	1	3							
j→			m		k→			i→						
5	7	10	19	|	1	3	14	1						
5	7	10	19	|		3	14	1	3					
5	7	10	19	|			14	1	3	5				
	7	10	19	|			14	1	3	5	7			
		10	19	|			14	1	3	5	7	10		
			19	|			14	1	3	5	7	10	14	
			19	|				1	3	5	7	10	14	19

MergeSort

Sortieren durch Verschmelzen

Beispiel (Verschmelzen)

10	5	7	19	|	14	1	3							
j→			m		k→			i→						
5	7	10	19	|	1	3	14	1						
5	7	10	19	|		3	14	1	3					
5	7	10	19	|			14	1	3	5				
	7	10	19	|			14	1	3	5	7			
		10	19	|			14	1	3	5	7	10		
			19	|			14	1	3	5	7	10	14	
			19	|				1	3	5	7	10	14	19

MergeSort

Sortieren durch Verschmelzen

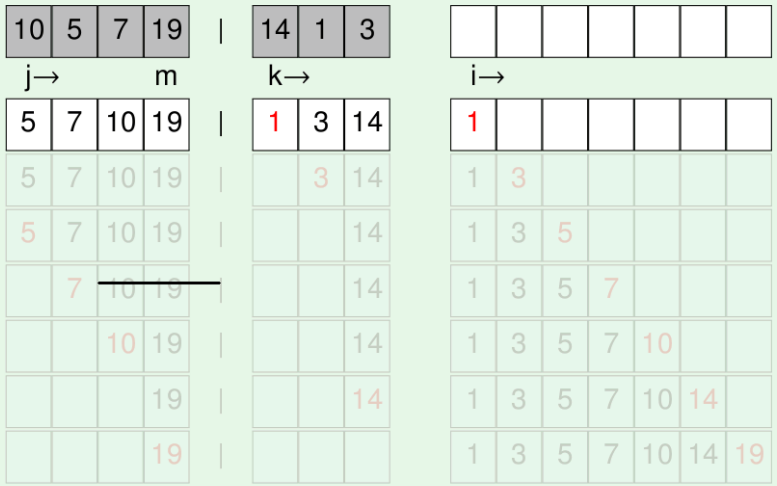
Beispiel (Verschmelzen)

10	5	7	19	|	14	1	3							
j→			m		k→			i→						
5	7	10	19	|	1	3	14	1						
5	7	10	19	|		3	14	1	3					
5	7	10	19	|			14	1	3	5				
	7	10	19	|			14	1	3	5	7			
		10	19	|			14	1	3	5	7	10		
			19	|			14	1	3	5	7	10	14	
			19	|				1	3	5	7	10	14	19

MergeSort

Sortieren durch Verschmelzen

Beispiel (Verschmelzen)



Analyse rekursiver Funktionen

Divide-and-Conquer-Algorithmen

- nichtrekursive Unterprogrammaufrufe sind einfach zu analysieren (separate Analyse des Funktionsaufrufs und Einsetzen)
- rekursive Aufrufstrukturen liefern Rekursionsgleichungen

$\Rightarrow$ Funktionswert wird in Abhängigkeit von Funktionswerten für kleinere Argumente bestimmt
gesucht: nichtrekursive / geschlossene Form

Anwendung: Divide-and-Conquer-Algorithmen

- gegeben: Problem der Größe $n = b^k$ $(k \in \mathbb{N}_0)$
- falls $k \geq 1$:
  - zerlege das Problem in d Teilprobleme der Größe n/b
  - löse die Teilprobleme (d rekursive Aufrufe)
  - setze aus den Teillösungen die Lösung zusammen
- falls $k = 0$ bzw. $n = 1$: investiere Aufwand a zur Lösung

Analyse rekursiver Funktionen

Divide-and-Conquer-Algorithmen

- nichtrekursive Unterprogrammaufrufe sind einfach zu analysieren (separate Analyse des Funktionsaufrufs und Einsetzen)
- rekursive Aufrufstrukturen liefern Rekursionsgleichungen

$\Rightarrow$ Funktionswert wird in Abhängigkeit von Funktionswerten für kleinere Argumente bestimmt
gesucht: nichtrekursive / geschlossene Form

Anwendung: Divide-and-Conquer-Algorithmen

- gegeben: Problem der Größe $n = b^k$ $(k \in \mathbb{N}_0)$
- falls $k \geq 1$:
  - zerlege das Problem in d Teilprobleme der Größe n/b
  - löse die Teilprobleme (d rekursive Aufrufe)
  - setze aus den Teillösungen die Lösung zusammen
- falls $k = 0$ bzw. $n = 1$: investiere Aufwand a zur Lösung

Analyse rekursiver Funktionen

Divide-and-Conquer-Algorithmen

- Betrachte den Aufwand für jede Rekursionstiefe
- Anfang: Problemgröße n
- Level für Rekursionstiefe i: d^i Teilprobleme der Größe n/b^i

$\Rightarrow$ Gesamtaufwand auf Rekursionslevel i:

$$d^i c \frac{n}{b^i} = cn\left(\frac{d}{b}\right)^i \qquad \text{(geometrische Reihe)}$$

- $d < b$ Aufwand sinkt mit wachsender Rekursionstiefe; *erstes* Level entspricht konstantem Anteil des Gesamtaufwands
- $d = b$ Gesamtaufwand für jedes Level gleich groß; maximale Rekursionstiefe logarithmisch, Gesamtaufwand $\Theta(n \log n)$
- $d > b$ Aufwand steigt mit wachsender Rekursionstiefe; *letztes* Level entspricht konstantem Anteil des Gesamtaufwands