1 Mergesort

Erstellen Sie eine Klasse `de.hska.iwi.ads.solution.sorting.Mergesort`, welche die Klasse `de.hska.iwi.ads.sorting.AbstractMergesort` als Basisklasse besitzt. Der initiale Aufruf, bei der ein ausreichend großer Zusatzspeicher erzeugt wird, ist dort bereits enthalten.

Implementieren Sie rekursives Mergesort, so wie es in der Vorlesung behandelt wurde.

Testen Sie Ihre Implementierung ausgiebig mit JUnit, um alle Fehler zu finden. Überprüfen Sie insbesondere, ob Ihre Implementierung ein stabiler Sortieralgorithmus ist.

2 Umdrehen von Teilfolgen in einem Feld

Es gibt in der Standard-Java-Bibliothek keine Implementierung, um Teile eines Feldes umzudrehen.

Erstellen Sie eine Klasse `de.hska.iwi.ads.solution.sorting.ReverseArray`, welche das Interface `de.hska.iwi.ads.sorting.Reverse` implementiert.

Die dort definierte Methode `void reverse(E [] a, int from, int to)` soll die Reihenfolge alle Feldelemente im Bereich $a[from..to]$ umdrehen. Ihre Implementierung darf nur konstant viel Zusatzspeicher verwenden.

Falls die Werte von $from$ und to außerhalb der in der Javadoc angegebenen Bereiche liegen, ist nicht weiter definiert, was die Methode machen soll.

Testen Sie Ihre Implementierung mit JUnit.

3 Mergesort. Rechte Teilfolge umdrehen

Erstellen Sie eine Klasse `de.hska.iwi.ads.solution.sorting.ReverseMergesort`.

Implementieren Sie wieder rekursives Mergesort aber mit folgender Variante für das Verschmelzen von linker ($a[links..middle]$) und rechter ($a[middle + 1..rechts]$) bereits sortierter Teilfolge: Innerhalb des Verschmelzen wird erst die Reihenfolge der rechten Teilfolge umgekehrt. Dadurch ist die rechte Teilfolge absteigend sortiert. Beim Verschmelzen muss die rechte Teilfolge von rechts nach links durchwandert werden. Die beiden Sonderfälle, die in der vorhergehenden Variante auftreten, fallen jetzt weg.

Auch diese Implementierung soll `de.hska.iwi.ads.sorting.AbstractMergesort` als Subtyp besitzen. Um redundante Codeteile zum vorhangehendem `Mergesort` zu vermeiden, können Sie für `Mergesort` und `ReverseMergesort` eine eigenen gemeinsame abstrakte Basisklasse, die `AbstractMergesort` als Basisklasse besitzt, programmieren.

Verwenden zum Umkehren von Feldwerten Ihre Implementierung `ReverseArray`.

Testen Sie Ihre Implementierung ausgiebig mit JUnit, um alle Fehler zu finden.

Ist dieser Sortiertalgorithmus noch stabil?

Der Zeitaufwand Ihrer Implementierung soll im schlimmsten Fall $\Theta(n \log_2 n)$ sein. Prüfen Sie dies mit einen Feld, welches mindestens 100 000 Werte enthält. Die Werte können alle 0 sein. Bei quadratischen Zeitaufwand müsste dies schon einige Sekunden dauern.

Pseudocode

Im folgenden der Pseudocode von Mergesort aus der Vorlesung. Der Verweis auf das Feld `b` wird hier allerdings nicht übergeben, da er schon als Objektattribut existiert.

```
1  mergesort(a, links, rechts)
2  if links < rechts
3    m = (links + rechts) / 2
4    mergesort(a, b, links, m)
5    mergesort(a, b, m + 1, rechts)
6    verschmelzen(a, b, links, m, rechts)
```

```
1  verschmelzen(a, links, m, rechts)
2  l ← links
3  r ← m + 1
4  for i ← links to rechts
5    if (r > right
6          or (l ≤ m and a[l] ≤ a[r]) )
7      b[i] ← a[l]
8      l ← l + 1
9    else
10     b[i] ← a[r]
11     r ← r + 1
12 for i ← links to rechts
13   a[i] ← b[i]
```