Mergesort

Vorbemerkungen

Mergesort und Quicksort sind essentielle Bestandteile der heutigen informationstechnologischen Infrastruktur - ohne diese beiden Algorithmen sähe die Welt anders aus, da praktisch alle IT Systeme ständig Daten sortieren müssen.

- Mergesort kommt beispielsweise zum Einsatz bei Java-Methoden zum Sortieren von Objekte, sortieren in Perl, stabile Sortierverfahren bei CPP und Python, JavaScript Implementation von Firefox. Mergesort war einer der ersten in den 1940er Jahren auf dem EDVAC ausgeführten Algorithmen.
- Quicksort ist die Basis beim Sortieren von primitiven Typen in Java, qsort bei C, Visual C, Python, Matlab und JavaScript in Chrome basierten Browsern.

Grundsätzliche Funktionsweise

Mergesort ist eine Divide-and-Conquer Algorithmus, der prinzipiell folgendermaßen funktioniert:

- **Teile** das Array in zwei Hälften (Divide).
- Sortiere (**rekursiv**) die beiden Hälften (Sort).
- **Füge** die beiden Hälften wieder zu einen sortierten Array **zusammen** (Merge).

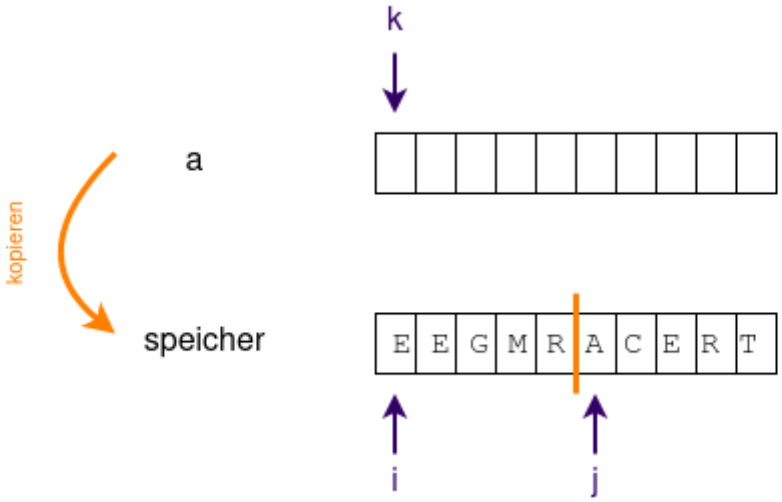
Zentrale Teilaufgabe: Merge

Das Zusammenfügen der sortierten Teilarrays ist ein zentraler Bestandteil des Mergesort-Verfahrens, darum schauen wir uns das hier an einem Beispiel nochmal etwas genauer an, wie man diesen Vorgang verstehen und implementieren kann.

Gegeben sind also zwei sortierte Teilarrays, die zu einem sortierten Array zusammengefügt werden sollen:

Dazu gehen wir wie folgt vor:

- Zunächst kopieren wir das gesamte Array in ein "Speicherarray".
- Anschließend verwenden wir drei Indizes:
  - i: Beginnt beim ersten Element des ersten Teilarray des Speicherarrays
  - j: Beginnt beim ersten Element des zweiten Teilarrays im Speicherarray
  - k: Beginnt beim ersten Element des Arrays, das zusammengesetzt werden soll



i und j laufen jetzt durch die Teilarrays, die zugehörigen Elemente werden verglichen. Das kleinere der beiden Elemente wird ins ursprüngliche Array übernommen und der jeweilige Index inkrementiert. Gelangt ein Index ans Ende, kann der Rest des unvollständig abgearbeiteten Teilarrays übernommen werden.

(A1)

Implementiere im Bluej-Projekt https://codeberg.org/qg-info-unterricht/algs4-sort-bluej in der Klasse `StandaloneMerge` das Zusammenführen der beiden Teilarrays a und b. Die Hinweise sind gestufte Hilfen, die du bei Problemen nacheinander ansehen kannst.

Hinweis 1:

Da die Eingabe bereits in zwei Teilarrays erfolgt, entfällt der Kopiervorgang aus dem Beispiel - du kannst direkt aus den Arrays a und b in das Array merged einfügen.

Hinweis 2:

```
// Das sind die Indizes für die beiden Teilarrays
// Beide beginnen bei 0
int i=0;
int j=0;

// k ist läuft von 0 bis merged-length-1
// bei jedem Durchlauf muss ein neues Element eingefügt werden -
// nur welches?
// Es müssen 4 Fälle unterschieden werden. Denke an die Länge der Arrays a
und b!
for(int k=0; k<merged.length; k++) {

// Fall 1:

// Fall 2:

// Fall 3:

// Fall 4:

}
```

Hinweis 3:

```
// Das sind die Indizes für die beiden Teilarrays
// Beide beginnen bei 0
int i=0;
int j=0;

// k ist läuft von 0 bis merged-length-1
// bei jedem Durchlauf muss ein neues Element eingefügt werden -
// nur welches?
// Es müssen 4 Fälle unterschieden werden. Denke an die Länge der Arrays a
und b!
for(int k=0; k<merged.length; k++) {


// Fall 1 und 2 müssen überprüfen, ob sich die
// Indizes von a und b im erlaubten Bereich befinden.
// Was bedeutet das, wenn der INdex außerhalb des erlaubten Bereichs ist?
// Was muss dann geschehen?

  // Fall 1: i ist außerhalb der Länge von a
  if(i >= a.length) {

  // Fall 2:  j ist außerhalb der Länge von b
  } else if(j >= b.length) {
```

```
  // Fall 3:
  } else if() {

  // Fall 4:
  } else {

  }
}
```

Hinweis 4:

```
int i=0;
int j=0;

for(int k=0; k<merged.length; k++) {
  if(i >= a.length) {
    // Das Array a ist abgearbeitet, jetzt können die Elemente
    // von b ohne weiteres nach merged übertragen werden, weil a und b
    // ja sortiert sind! Der Index von b muss jeweils erhöht werden.
    merged[k] = b[j];
    j++;
  } else if(j >= b.length) {
    // Das Array b ist abgearbeitet, jetzt können die Elemente
    // von b ohne weiteres nach merged übertragen werden, weil a und b
    // ja sortiert sind!
    merged[k] = a[i];
    i++;
  } else if(less(b[j], a[i])) {
    // Was fehlt hier?
  } else {
    // Was fehlt hier?
  }
}
```

Lösungsvorschlag

Wenn du auf den Lösungsvorschlag zurückgreifen musstest: Kommentiere den Lösungsvorschlag ausführlich, um dir klar zu machen, was hier geschieht!

```
int i=0;
int j=0;

for(int k=0; k<merged.length; k++) {
  if(i >= a.length) {
     merged[k] = b[j++];
```

```
  } else if(j >= b.length) {
     merged[k] = a[i++];
  } else if(less(b[j], a[i])) {
      merged[k] = b[j++];
  } else {
      merged[k] = a[i++];
  }
}
```

(A2)

Gib den Aufwand des Merge in O-Notation an und begründe deine Aussage.

Mergesort: Versuch 1

(A3) Naiver Mergesort

Implementiere unter Verwendung der oben implementierten *merge*-Methode einen naiven Mergesort-Algorithmus in der Methode `public String[] msort(String[] a)`. Eine Vorlage findest du im Bluej-Projekt https://codeberg.org/qg-info-unterricht/algs4-sort-bluej in der Klasse `MergeSortNaiv`. Eine Methode 'public String[] merge(String[] s, String[] t)'' ist entsprechend bereits implementiert.

Orientiere dich an folgendem Pseudocode:

```
msort: String[] a
  wenn a.laenge ist 1:
    return a
  ende wenn
  mitte = a.lange/2
  ta1=teilarray von a von 0 bis mitte
  ta2=teilarray von mitte+1 bis a.laenge-1
  return merge(msort(ta1), msort(ta2))
```

- Teste den Algorithmus
- Bewerte die Implementation in Hinsicht auf Ressourcenverbrauch und Effizienz

Arrayerzeugung ist teuer

Prinzipiell funktioniert unser "naiver Mergesort", jedoch ist die häufig auftretende Operation der Arrayerzeugung und das hineinkopieren der Array-Elemente beim Teilen des Arrays teuer und sollte vermieden werden.

Merge reloaded

Wir befassen uns zunächst nochmal mit dem Merge-Vorgang und implementieren mit unseren Kenntnissen vom ersten Versuch die Methode

```
public void merge(String[] a, String[] speicher, int min, int mitte, int max)
```

in der Vorlage *StandaloneMerge*. Um diese Methode aufrufen zu können, müssen wir die beiden Arrays a und b im Konstruktor zusammenfügen und außerdem ein Speicherarray derselben Länge erzeugen, das man dem Methodenaufruf mitgeben kann.