Team: 04

Florian Kletz (#2093735), Micha Severin (#2093379)

AI-3 AD — Praktikum #2

Aufgabenaufteilung:

Beide Teammitglieder waren Maßgeblich an allen Abschnitten beteiligt. Im Besoderen, bei der Verifikation und Überprüfung ob Methoden das gewünschte Ergebnis liefern und richtig arbeiten, sowie bei Lösungsansätzen und Ideen bei Fehlern.
Grob gegliedert wurde es wie folgt:

Florian Kletz hat bearbeitet:

- I/O Operationen

Micha Severin hat bearbeitet:

- Mergesort Algorithmus

Quellenangaben:

1. JavaDocs

Begründung für Codeübernahme

1. Es wurde kein Code aus anderen Quellen übernommen.

Bearbeitungszeitraum:

Florian Kletz	Dienstag 16.04.2013 20:00-22:00 Samstag 20.04.2013 14:00-16:00 Mittwoch 24.04.2013 17:00-21:00 Samstag 27.04.2013 09:00-15:00
Micha Severin	Freitag 26.04.2013 11:00-17:00 Samstag 27.04.2013 12:00-18:00 Sonntag 06.04.2013 11:00-16:00
Gemeinsam	Samstag 27.04.2013 13:00-18:00 Sonntag 28.04.2013 12:00-16:00

Aktueller Stand:

- Algorithmen sind implementiert und mit dem Beispiel aus dem Foliensatz getestet.
- Code zur erstellen des Eingabebands ist implementiert.
- Code zum einlesen der Eingabebands ist implementiert.

Probleme:

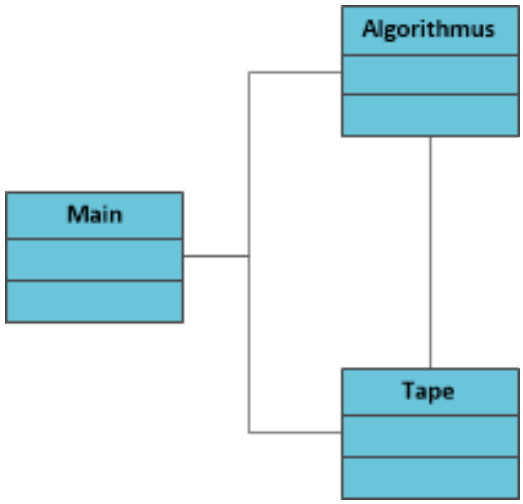
Zu den größten Problemen gehörte, den eigentlichen Mergesort Algorithmus zu implementieren, da dieser nicht über die Grenzen der bereits sortierten Teilfolgen hinaus laufen durfte. Desweiteren erwies sich der Wechsel zwischen den Ausgabebändern als

schwierig, da dies dynamisch berechnet werden musste.

Offene Fragen:

Wir können aus den Folien und auch durch weitere Recherche, nicht eindeutig die Arbeitsweise des Mehr-Wege-Mergesort Algorithmus nachvollziehen.
Unserem Verständnis zufolge, zeigt das Beispiel, das es nur ein Ausgabeband gibt und somit eher das Mehrphasen Mergesort Prinzip darstellt. Somit stell sich uns die Frage, wie der Mehr-Wege-Algorithmus mit verschiedener Anzahl von Bändern arbeiten würde?

Skizze:



Nachtrag:

Für den Mehr-Wege-Mergesort Algorithmus werden Bänder benötigt. Für den ausgeglichenen Zwei-Wege-Mergesort in diesem Fall vier Bänder. Zwei Eingabe- u. zwei Ausgabebänder.
Auf dem ersten Eingabeband muss die unsortierte Folge gespeichert sein. Dies können zum Beispiel Zahlen oder Buchstaben sein. Wir werden nur eine Implementierung mit Integer Zahlen ermöglichen.

Die Bänder werden mit einer Klasse erstellt. Die Länge der Folge kann das Fassungsvermögen des Arbeitsspeicher überschreiten, somit werden die Bänder aus das Dateisystem ausgelagert. Aufgrund des Fassungsvermögens des Arbeitsspeichers ist es nicht möglich eine interne Datenstruktur wie zum Beispiel ein Array zu verwenden.

Die Folge wird durch Zufallszahlen generiert und auf das erste Eingabeband gespeichert.

Danach werden beim initialen Run Folgen mit der Runlänge auf die Ausgabebänder sortiert geschrieben. Diese sortierten Teilfolgen werden immer abwechselt auf die beiden zur Verfügung stehenden Ausgabebänder geschrieben. Um die Teilfolgen zu sortieren würde sich der Bubblesort Algorithmus anbieten. Es wäre aber grundsätzlich möglich verschiedene Sortier-Algorithmen zu verwenden. Die Runlänge ist variabel.

In dem eigentlichen Mergesort-Algorithmus werden dann zwei dieser Teilfolgen zu einer größeren sortierten Teilfolge vereinigt. Deren Länge verdoppelt sich somit. Diese Teilfolgen werden auf den freien Bändern gespeichert. Nachdem die beiden Eingabebänder abgearbeitet sind, findet ein Wechsel zwischen Eingabe und Ausgabebändern statt. Ein- u. Ausgabebänder werden im Wechsel gelesen und beschreiben.Nach jedem Wechsel werden die Bänder zurückgespult und wieder bei dem Index 0 begonnen.

Die gesamte Sortierung ist dann abgeschlossen, wenn die Runlänge größer als die initiale Länge des ersten Eingabebandes ist.