Lehrstuhl für Algorith.
und Datenstrukturen
Prof. Dr. Hannah Bast
Axel Lehmann

Algorithmen und Datenstrukturen
(Informatik II) SS 2017

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Übungsblatt 1

Abgabe bis Dienstag, den 2. Mai um 12:00 Uhr

Aufgabe 1 (5 Punkte)

Melden Sie sich bei unserem Kurssystem Daphne an. Den Link dazu finden Sie auf der Wikiseite der Lehrveranstaltung. Achten Sie darauf, dass Ihre Daten korrekt sind, insbesondere dass Sie unter der angegebenen E-Mail Adresse auch erreichbar sind.

Aufgabe 2 (5 Punkte)

Implementieren Sie das in der Vorlesung erklärte (nicht-rekursive!) *MergeSort*. Beachten Sie dazu die auf dem Wiki verlinkte (programmiersprachen-unabhängige) Design-Vorlage *MergeSort.TIP*.

Beachten Sie unbedingt auch die grundsätzliche Bemerkungen zu den Implementierungen auf Seite 2 dieses Übungsblattes (gültig für *alle* Übungsblätter dieser Lehrveranstaltung).

Aufgabe 3 (5 Punkte)

Messen Sie die Laufzeit von Ihrem *MergeSort* aus Aufgabe 2 und auch noch einmal von dem *MinSort* aus der Vorlesung. Betrachten Sie verschiedene Eingabegrößen n wie folgt:

1. Wählen Sie die Eingabegrößen so, dass sich erstens ein aussagekräftiges Schaubild ergibt und zweitens das Sortieren mit der von Ihnen gewählten Programmiersprache nicht zu kurz und nicht zu lange dauert (bei unter 1 Mikrosekunde pro Sortiervorgang kann es Probleme mit der Messgenauigkeit geben, und insgesamt sollte ihr Experimente nicht länger als 1 Minute dauern, sonst treiben Sie Ihren Tutor in den Wahnsinn).

2. Wählen Sie Ihre Eingaben alle so, dass die Elemente dort gerade verkehrt herum sortiert sind.

3. Diskutieren Sie Ihre Ergebnisse kurz in der Datei *erfahrungen.txt* (siehe Aufgabe 4).

Aufgabe 4 (5 Punkte)

Committen Sie Ihren Code (samt Unit Tests) und die Schaubilder in das SVN, in einen eigenen Unterordner *blatt-01*. Committen Sie in diesem Unterordner außerdem eine Textdatei *erfahrungen.txt*. Beschreiben Sie dort in ein paar Sätzen Ihre Erfahrungen mit diesem Übungsblatt und den Vorlesungen dazu. Insbesondere: Wie lange haben Sie ungefähr gebraucht? An welchen Stellen gab es Probleme und wieviel Zeit hat Sie das gekostet?

[in freudiger Erwartung wenden]

Die 10 Gebote (gültig für alle Übungsblätter)

1. Als Programmiersprache stehen Ihnen grundsätzlich Python, Java und C++ zur freien Auswahl. Sie müssen sich auch nicht festlegen, sondern können sich für jedes Übungsblatt neu entscheiden.

2. Die Designvorlage (.TIP Datei, falls vorhanden) ist nicht verbindlich. Sie beruht aber auf viel Erfahrung, von daher sollten Sie sich dreimal überlegen, wenn Sie davon abweichen.

3. Schreiben Sie Ihren Code modular: wenn ein Teil des Codes für sich alleine umfangreich bzw. komplex genug ist oder mehrfach benötigt wird, gehört er in eine eigene Funktion.

4. Dokumentieren Sie Ihren Code: jedes Stück Code, dessen Funktionsweise sich nicht unmittelbar duch Lesen des Codes ergibt, sollte erklärt werden. Die Erklärung sollte kurz und aussagekräftig sein. Manchmal ist ein Beispiel nützlicher (und kürzer) als eine abstrakte Erklärung.

5. Schreiben Sie für jede nicht-triviale Funktion einen Unit Test. Jeder Unit Test sollte mindestens ein nicht-triviales Beispiel überprüfen. Wenn es kritische Grenzfälle gibt, die sich durch wenig Aufwand leicht nachprüfen lassen (z.B. Verhalten einer Methode bei leerem Eingabefeld), sollten Sie das ebenfalls tun.

6. Laden Sie Ihren Code vollständig in unser SVN hoch, inklusive *Makefile* (für Python und C++) bzw. *build.xml* (Java). Andere Dateien (insbesondere *.pyc* oder *.o* oder *.class*) dürfen nicht mit hochgeladen werden, sonst droht ewige Verdammnis oder Schlimmeres.

7. Die finale Version von Ihrem Code muss fehlerfrei auf Jenkins durchlaufen. Zwischenversionen (die Sie zum Beispiel rein zu Sicherungszwecken in unser SVN hochgeladen haben) müssen nicht auf Jenkins durchlaufen.

8. Code, der auf Jenkins nicht zumindest kompiliert („make compile“ bzw. „ant compile“) oder der keine Tests hat, wird nicht korrigiert, weil das unzumutbar viel Arbeit für die Tutoren wäre.

9. Wenn Sie ein Problem bei der Implementierung haben, suchen Sie ein paar Minuten (auf Google oder auch auf dem Forum, da wurden sehr viele Fragen schon mal gestellt) nach dem Fehler. Wenn Sie nicht fündig werden, fragen Sie gerne auf dem Forum, da wird Ihnen in der Regel schnell geholfen. Eine Anleitung für das richtige Fragen auf dem Forum findet sich auf dem Wiki (Stichwort: konkret fragen mit Fehlermeldung + Zeilennummer + relevantem Code, und nicht nur „Mein Code geht nicht“, sonst kann Ihnen keiner helfen).

10. Sie können gerne zusammen über die Übungsblätter nachdenken, aber der Code bzw. die Lösungen müssen zu **100%** selber geschrieben werden. Auch das teilweise Übernehmen gilt als Täuschungsversuch, mit den entsprechenden Konsequenzen. Wir müssen das so deutlich sagen, weil es in der Vergangenheit immer wieder vorgekommen ist. Man lernt nichts, wenn man Code bzw. Lösungen von anderen übernimmt und es ist auch einfach unfair gegenüber dem Großteil der Teilnehmer, die sich ehrlich Mühe geben.