Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2024
Musterlösung Übungsblatt 1

Abgabe: Dienstag, 23. April, 2024, 10:00 Uhr

Aufgabe 1: Anmeldung *(5 Punkte)*

Melden Sie sich beim Kurssystem Daphne (Ist noch nicht eingerichtet, wir geben über Zulip Bescheid. Ihr könnt aber schonmal mit Aufgabe 2 und 3 anfangen.) an. Den Link dazu finden Sie auch auf der Kurs-Website. Achten Sie darauf, dass Ihre Daten korrekt sind, insbesondere dass Sie unter der angegebenen E-Mail Adresse auch erreichbar sind. Führen Sie ein `checkout` auf Ihr SVN-Repository durch.[1]

Aufgabe 2: Mehrteiliger Mergesort *(5 Punkte)*

Implementieren Sie den in der Vorlesung erklärten Algorithmus *Mergesort* mit *zusätzlichem Parameter* k. Der Parameter k entscheidet in wie viele Teile wir das Array im Divide Step aufteilen. Zum Beispiel ist für $k = 2$ dieser parametrisierte Mergesort identisch mit dem in der Vorlesung vorgestellten. Für $k = 100$ wird jedesmal in 100 gleich große Teile getrennt, dementsprechend müssen also der Divide-und der Combine-Step beide angepasst werden. Verwenden Sie dazu die auf der Webseite verlinkte Design-Vorlage `MergeSort.py`. Schreiben Sie Unit Tests [2]. Die Unit Tests sollten grundsätzlich mindestens ein nicht-triviales Beispiel überprüfen. Wenn es kritische Grenzfälle gibt, die sich leicht nachprüfen lassen (z.B. Verhalten einer Methode bei leerem Eingabefeld), sollen Sie dies tun.

Musterlösung

Siehe `MergeSort.py` auf der Website.

Aufgabe 3: Zeitmessungen *(5 Punkte)*

Messen Sie die Laufzeit Ihrer *Mergesort*-Implementierung mit verschiedenen Werten für den Parameter k. Wir wollen 4 verschiedene Werte für k testen, $k = 2$, $k = 3$, $k = \lceil \log_2 n \rceil$ und $k = \lceil n/4 \rceil$, wobei hier n die Länge des zu sortierenden Arrays ist.[3] Stellen Sie für jede der vier Varianten die Entwicklung der Laufzeit über n graphisch dar, indem Sie Arrays der Länge 100 bis 10000 (zum Beispiel inkrementell in 100er Schritten) zufällig generieren und die Zeit messen.[4] Diskutieren Sie Ihre Ergebnisse kurz in Ihren `erfahrungen.txt` (siehe Aufgabe 4).

[1] Ihr SVN-Repository wird bei der ersten Anmeldung bei Daphne automatisch angelegt. Die URL ist: `https://daphne.informatik.uni-freiburg.de/ss2024/AlgoDat/svn/ihr-rz-account-name`

[2] Nutzen Sie dafür `doctest`'s wie sie auch schon in der Vorlage zu finden sind. Mit `python -m doctest MergeSort.py` können Sie ihre Tests ausführen.

[3] Code der zufällige Arrays generiert lässt sich bereits in der Vorlage finden.

[4] Die Unterschiede der Laufzeiten der Algorithmen werden am deutlichsten wenn diese gemeinsam in einem einzigen Schaubild aufgetragen werden mit n auf der x-Achse und der Laufzeit der entsprechenden Variante auf der y-Achse. Für ein klareres Bild können Sie sowohl eine *lineare* als auch eine *logarithmische* y-Skala ausprobieren.

Musterlösung

Abbildung 1 stellt die Laufzeiten jeweils mit unterschiedlicher y-Skala graphisch dar. Wir haben in unseren Graphiken auch noch den interessanten Fall $k = \sqrt{n}$ und machen folgende Beobachtungen:
Für $k = 2, k = 3$ und $k = \log n$ lässt sich quasi kein Unterschied erkennen und für $k = \sqrt{n}$ und $k = \frac{n}{4}$ sind die Ergebnisse schlechter. Auf der Linearen Skala sieht man nur eine leichte Verschlechterung für $k = \sqrt{n}$ und eine deutliche verschlechterung für $k = \frac{n}{4}$. Die Log Skala zeigt dann das bei $k = \sqrt{n}$ bereits eine deutliche Verschlechterung vorliegt, diese aber in der Linearen Skala noch nicht so auffält. Die Lineare Skala wird durch den sehr schlechten Fall $k = \frac{n}{4}$ verzehrt.
Nächste Woche werden wir sehen wie sich diese Unterschiede erklären.

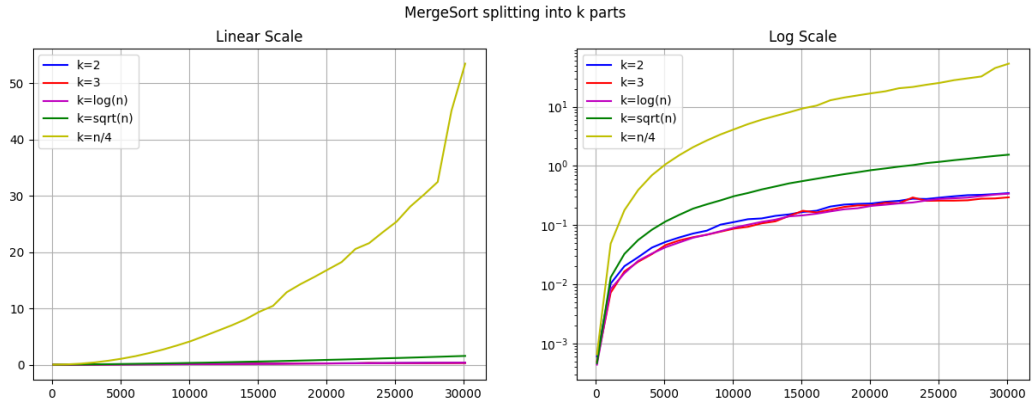

Abbildung 1: Die Laufzeitvon Mergesort (in Sekunden) auf Zufällig generierten Arrays der Größe 1000 bis 30000 für verschiedene Einstellungen des Parameters k.

Aufgabe 4: Abgabe *(5 Punkte)*

Committen Sie Ihren Code (inkl. Tests) und die Schaubilder in das SVN, in einen eigenen Unterordner `uebungsblatt-01`. Gehen Sie dabei so vor, wie in der Vorlesung vorgeführt. Stellen Sie sicher, dass auf Jenkins alles (inkl. Style Check und Unit Tests) fehlerfrei durchläuft.
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?
Was passiert in dem Fall $k = \frac{n}{4}$, wie Verhält sich der Algorithmus? Ist es ähnlich zu einem anderen Sortieralgorithmus den wir kennengelernt haben?