Technische Universität Berlin **WiSe 2023/24**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Dorothea Kiefer-Hoeft
Prof. Dr. Max Klimm
Dr. Frank Lutz, Svenja M. Griesbach, Martin Knaack

7. Programmieraufgabe Computerorientierte Mathematik I

Abgabe: 22.12.2023 über den Comajudge bis 17 Uhr

Bitte beachten Sie: Die Herausgabe oder der Austausch von Code (auch von Teilen) zu den Programmieraufgaben führt für *alle* Beteiligten zum *sofortigen Scheinverlust*. Die Programmieraufgaben müssen von allen Teilnehmenden alleine bearbeitet werden. Auch Programme aus dem Internet dürfen nicht einfach kopiert werden.

1 Problembeschreibung

Für $n \in \mathbb{N}$ sei $(a_1, \ldots, a_n)$ eine endliche, monoton wachsende Folge reeler Zahlen. In dieser Programmieraufgabe betrachten wir die folgenden zwei Probleme:

1. Finde den ersten Eintrag in der Folge der größer als ein gegebener Wert ist.
2. Finde eine Teilfolge, die alle Elemente bis auf Wiederholungen enthält.

Beide Probleme können intuitiv in linearer Laufzeit $O(n)$ gelöst werden, indem man über alle Elemente iteriert.

In dieser Aufgabe wollen wir Programme mit teilweise besseren Laufzeiten programmieren. In der Vorlesung haben Sie bereits die binäre Suche kennengelernt. Die binäre Suche ermöglicht es, einen Eintrag in einer geordneten Liste der Länge n mit Laufzeit $O(\log n)$ zu finden. Nutzen Sie die binäre Suche um für das erste Problem einen Algorithmus mit Laufzeit $O(\log n)$ zu schreiben. Schreiben Sie anschließend einen Algorithmus, der das zweite Problem mit Laufzeit $O(k \log n)$ löst, wobei k der Anzahl der Elemente in der Folge ohne Wiederholungen entspricht.

2 Aufgabenstellung und Anforderungen

1. Schreiben Sie eine Funktion

   `find_first_larger(L, e)`,

   die den Index des ersten Elementes in der aufsteigend sortierten Liste `L` findet, das einen größeren Wert hat als das Element `e`. Sollte ein solches Element nicht existieren, dann soll die Funktion Position hinter dem letzten Element ausgeben werden. Die Laufzeit des Programms soll in $O(\log n)$ liegen.

2. Schreiben Sie eine Funktion

   `unique(L)`,

   die für eine aufsteigend sortierte Liste `L` eine neue Liste ausgibt, die alle Elemente von `L` bis auf Wiederholungen enthält. Die ausgegebene Liste soll weiterhin aufsteigend sortiert sein. Die Laufzeit des Programms soll in $O(k \log n)$ liegen, wobei k die Anzahl der Elemente in `L` ohne Wiederholungen sei.

2.1 Beispielaufrufe

```
>>> L = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> i = find_first_larger(L, 0)
>>> print(i)
0
>>> i = find_first_larger(L, 10)
>>> print(i)
5
>>> i = find_first_larger(L, 20)
>>> print(i)
10

>>> L = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> K = unique(L)
>>> print(K)
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
>>> L = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
>>> K = unique(L)
>>> print(K)
[1]
>>> L = [1, 1, 1, 2, 2, 2, 3, 3, 4, 5, 6, 6]
>>> K = unique(L)
>>> print(K)
[1, 2, 3, 4, 5, 6]
```

3 Tipps und Anmerkungen

- Zu Ihrer Erinnerung: Sie müssen eine der Programmieraufgaben PA07, PA08 oder PA09 bei einem Tutor oder einer Tutorin in den Rechnerbetreuungen vorstellen. Die vorläufige Deadline ist der 26.01.23. Der Code den Sie vorstellen sollte kommentiert sein.