Algorithmen und Datenstrukturen INF3/ICS3

Wintersemester 2023/24

Prof. Dr. Georg Schied

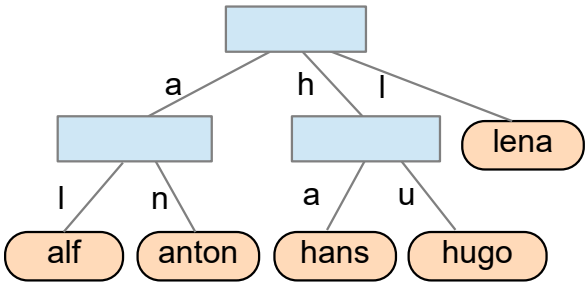
Aufgabenblatt 10

Abgabetermin: Mo. 8. Januar 2024, 23:59 Uhr

Zum Bestehen müssen 10 von 20 Punkten erreicht werden.

Aufgabe 10.1

Gegeben sei folgender Trie für Zeichenketten über dem Alphabet {a, …., z}:



Fügen Sie folgende Zeichenketten in den Trie ein:

anna, leon, alfred, hermann

Aufgabe 10.2 - Scheinaufgabe (5 Punkte)

Gegeben ist das Alphabet Σ = {a,b,c}. Fügen Sie folgende Zeichenketten nacheinander in einen zu Beginn leeren **Trie** ein:

ccaa, bac, abba, caba, aab, abbcc, abb

Aufgabe 10.3 - Scheinaufgabe (4 P)

In einem Dokumentenmanagementsystem müssen zu Dokumenten-IDs die zugehörigen Dateinamen gespeichert werden. Dazu soll eine **Hashtabelle** der Größe **m = 10** mit **Verkettung der Überläufer** eingesetzt werden. Es wird eine Hashfunktion nach **Division-Rest-Methode** verwendet**.**
Führen Sie nacheinander folgende Operationen durch, ausgehend von einer leeren Tabelle.

1. put(63, "A.doc")
2. put(47, "B.doc")
3. put(23, "C.doc")
4. put(44, "D.doc")
5. put(53, "E.doc")

6. remove(23)
7. put(63, "F.doc")
8. put(33, "G.doc")

Kommentieren Sie kurz das Vorgehen und geben Sie den Zustand der Hashtabelle nach diesen Operationen an. Achten Sie darauf, dass klar zu sehen ist, was gespeichert wird.

Aufgabe 10.4

Führen Sie die gleichen Operationen wie in Aufgabe 10.3 mit einer Hashtabelle der Größe m=10 durch, die **offene Adressierung** und **lineare Sondierung** zur Kollisionsauflösung verwendet.

Aufgabe 10.5 - Scheinaufgabe (6 P)

In einer **Hashtabelle** der Größe **m = 10** mit **offener Adressierung** soll der Punktestand von Spielern eines Computerspiels gespeichert werden. Es wird **quadratische Sondierung** mit der **Hashfunktion**

$$h(str, i) = (str.\text{length}() + i^2) \bmod m$$

verwendet. Dabei ist *str* ein String und *i* der Sondierungsschritt (gezählt ab 0).

a) Ausgehend von einer leeren Tabelle werden zunächst folgende Operationen nacheinander ausgeführt:

1. put("Ben", 80)
3. put("Lara", 71)
2. put("Sofia", 92)
4. put("Max", 53)
5. put("Lara", 84)
6. put("Anna", 55)

Welcher Zustand ergibt sich für die Hashtabelle nach diesen Operationen? Achten Sie darauf, dass klar erkennbar ist, welche Daten in der Tabelle gespeichert werden (d.h. welche Spalten die Tabelle hat).

b) Es werden anschließend folgende Operationen ausgeführt:

7. remove("Sofia")
8. remove("Max")
9. put("Barbara", 26)
10. put("Anna", 67)

Geben Sie den neuen Zustand der Hashtabelle nach diesen Operationen an.

c) Welche Sondierungsfolge (Adressen der Hashtabelle) ergibt sich, wenn nun

(1) der Punktestand von `Anna` gesucht wird?

(2) der Punktestand von `Mia` gesucht wird?

Aufgabe 10.6

a) In einer Hashtabelle mit **offener Adressierung** und **linearer Sondierung** steigt der Belegungsfaktor von **70% auf 95%**. Wie wirkt sich das auf die Laufzeit für die Suche im erfolgreichen Fall bzw. im erfolglosen Fall aus?

b) Wie verhält sich analog die Laufzeit bei einer Hashtabelle mit **Verkettung der Überläufer**, wenn der Belegungsfaktor von 70% auf 95% steigt?

c) Welchen Vorteil bietet **quadratische** Sondierung im Vergleich zu **linearer** Sondierung?

d) Welchen Vorteil bietet **doppeltes Hashing** im Vergleich zu **quadratischer Sondierung**?

Aufgabe 10.7 - Scheinaufgabe (5 P)

Eine Methode

```
public static Set<String> commonNames(String[] names1,
                                      String[] names2)
```

soll für zwei Arrays, die Namen enthalten, effizient die Menge aller Namen berechnet, die in beiden Arrays vorkommen.

a) Implementieren Sie in Klasse `Aufg10_7_Common` zwei Varianten dieser Methode, einmal unter Verwendung von `HashSet` und einmal unter Verwendung von `TreeSet` als Mengenimplementierung.

b) Welche Laufzeitkomplexität erwarten Sie für Ihre Implementierungen, unter der Annahme, dass beide Arrays die gleiche Länge *n* haben und etwas die Hälfte der Namen gemeinsam sind?

c) Messen Sie jeweils mit der vorgegebenen Messmethode in `Aufg10_7_Common` die Laufzeit beider Implementierung (mit Option `-Xint`).

- Ist das erwartete Laufzeitverhalten zu erkennen?
- Sind Unterschiede zwischen beiden Varianten erkennbar?