Übungsblatt 4

Abgabe via Moodle.
Deadline Fr. 3ter Juni

Aufgabe 1 (*Hashing,* $4+1+2+1$ *Punkte*)

Gegeben sei eine Hashtabelle mit 10 Buckets. Dabei sei die Hashfunktion h definiert als die Funktion, welche eine Zahl auf die Einerstelle abbildet, zum Beispiel gilt $h(13) = 3$.

1. Verwenden Sie einerseits Hashing mit verketteten Listen, andererseits Hashing mit linearer Suche (zyklisches Array), um folgende Operationen durchzuführen:

   Einfügen von 19, 52, 25, 29, 62, 12, 88, 10, 53

   Geben Sie jeweils die Tabellen nach dem Einfügen von 62 und nach dem Einfügen von 53 an.

2. Geben Sie die Anzahl der beim Einfügen betrachteten Hashtabellenplätze für beide Verfahren an!

3. Nehmen Sie an, dass nach jedem Datum mit gleicher Wahrscheinlichkeit gesucht wird. Welche Kosten sind für das jeweilige Verfahren für eine erfolgreiche Suche zu erwarten?

4. Wie groß ist der Speicherverbrauch, wenn sowohl Zeiger als auch Element ein Maschinenwort benötigen, und mit einem Nullzeiger, das Ende einer Liste markiert werden kann?

Aufgabe 2 (*Rechnungssystem,* $6+2$ *Punkte*)

Nehmen Sie an Sie haben eine große Datei die aus Tripeln der Form (*transaction, price, customerID*) besteht.

1. Entwickeln Sie einen Algorithmus, der den Gesamtzahlbetrag pro Kunden in erwarteter linearer Laufzeit berechnet.

2. Begründen Sie die erreichte Laufzeit!

Aufgabe 3 (*Entwurf einer Datenstruktur,* 8 *Punkte*)

Gegeben sei ein Datentyp D mit Elementen der Form (*Schlüssel*, *Nutzdaten*). Die Größen von Schlüssel und Nutzdaten seien konstant. Die maximale Anzahl n der verschiedenen Schlüssel, die zu einem bestimmten Zeitpunkt in einem etwaigen System existieren, sei von vornherein bekannt. Die maximale Anzahl m der verschiedenen *Tupel* sei jedoch *nicht* von vornherein bekannt. Es können zudem auch Tupel mit dem gleichen Schlüssel vorkommen (im Extremfall können sogar alle vorhandenen Tupel den gleichen Schlüssel haben). Entwerfen Sie nun eine Datenstruktur für den Typ D, die folgendes leistet:

- Es existiert eine Operation *Insert*(x_s, x_d), die ein Tupel (x_s, x_d) aus Schlüssel und Nutzdaten in die Datenstruktur einfügt. Ein Tupel wird dabei auch dann eingefügt, wenn in der Datenstruktur bereits ein Tupel mit dem selben Schlüssel existiert.
- Es existiert eine Operation *Remove*(x_s), die für einen Schlüssel x_s ein Tupel mit passendem Schlüssel zurückgibt und aus der Datenstruktur entfernt. Falls kein solches Tupel in der Datenstruktur existiert, wird NIL zurückgegeben.
- Beide Operationen benötigen erwartet $\mathrm{O}(1)$ Zeit.
- Die Datenstruktur hat einen Speicherbedarf von höchstens $\mathrm{O}(\max\{n, m\})$.

Aufgabe P4 (*Modified Fibonacci, optional*)

Für die praktischen Übungen verwenden wir die Plattform `www.hackerrank.com`. Hier müssen Sie sich registrieren um an den Übungen teilzunehmen. Unter dem Link

`https://www.hackerrank.com/ads-i-praktische-uebung`

finden die praktischen Übungen in der Form eines Programmierwettbewerbs statt.

In der vierten Challenge geht es um eine modifizierte Fibonacci Folge.

Die Vorschrift zur Berechnung der Folge ist

$$t_{i+2} = (t_i)^2 - t_{i+1} \tag{1}$$

Die Anfangswerte t_0 und t_1 die für die Berechnung notwendig sind werden als Eingabe bereitgestellt.

Zusätzlich ist die Anzahl der zu berechnenden Folgenglieder n gegeben. Das heißt, die Rückgabe der zu vervollständigenden Funktion **newFibonacci** soll gerade t_n sein.

Um die Effizienz ihres Algorithmus zu testen gibt es wie letzte Woche einen großen Test (Case 3) auf Hackerrank. Wenn Ihr Algorithmus effizient genug ist, wird der Test in der vorgegebenen Zeit von 2 Sekunden meistens durchlaufen. Leider funktioniert es nicht immer dass dieser Test durchläuft, selbst bei der optimalen Lösung. Am besten submitten Sie ihren Algorithmus mehrmals.

Falls dieser Test trotzdem immer aufgrund von Zeitmangel fehlschlägt, gibt es eventuell die Möglichkeit Ihren Algorithmus noch zu verbessern.

Eine genauere Beschreibung, sowie ein Beispiel finden Sie auf HackerRank.