Theofanis Gkioles Blatsoukas Matrikelnummer: s0583909

Algorithmen und Datenstrukturen – Programmieraufgabe II

HashTabelle

Projetbescreibung

Der bereitgestellte Code ist eine Java-Implementierung einer Hashtabelle. Er enthält eine HashTable-Klasse, die die Hash-Tabellen-Datenstruktur darstellt, und eine ValueEntry-Klasse, die in der Hash-Tabelle gespeicherte Schlüssel-Wert-Paare darstellt. Der Code enthält auch eine Main-Klasse namens App, die die Verwendung der Hash-Tabelle demonstriert.

Die Klasse HashTable verfügt über die folgenden Funktionen:

1. Einfügen von Schlüssel-Wert-Paaren unter Verwendung der Kollisionsauflösungstechniken Linear Probing und Quadratic Probing (double hashing).
2. Entfernen eines Schlüssel-Wert-Paares aus der Hash-Tabelle.
3. Abruf des mit einem bestimmten Schlüssel verbundenen Wertes.
4. Berechnung des Lastfaktors oder der Belegung der Hash-Tabelle.
5. Initialisierung der Hash-Tabelle mit einer geeigneten Größe auf der Grundlage der Anzahl der Einträge und Primzahlberechnungen.

Die Klasse ValueEntry ist eine Datenstruktur, die ein Schlüssel-Werte-Paar enthält.

Die Klasse App aus dem hashtabelle-Paket dient als Haupteinstiegspunkt des Programms. Sie liest Einträge aus der names.csv Datei, initialisiert eine Hashtabelle und bietet ein Konsolenmenü zum Suchen und Entfernen von Namen aus der Hashtabelle. Sie ermöglicht auch die Abfrage der Belegung der Hashtabelle.

Code Dokumentation

HashTable-Klasse:

HASH_TABLE_SIZE: Stellt die Größe der Hashtabelle dar.
Größe: Speichert die Anzahl der derzeit in der Hash-Tabelle gespeicherten Schlüssel-Wert-Paare.
table: Ein Array von ValueEntry-Objekten, die die Hash-Tabelle darstellen.
totalprimeSize: Speichert die Größe der Primzahl, die für die quadratische Sondierung verwendet wird.

HashTable(int ts): Konstruktor zum Initialisieren der Hashtabelle mit der angegebenen Größe.
getPrime(): Berechnet und liefert eine Primzahl, die zur Größe der Hash-Tabelle passt.
getSize(): Gibt die Anzahl der Schlüssel-Wert-Paare in der Hash-Tabelle zurück.
isEmpty(): Prüft, ob die Hash-Tabelle leer ist.

getkey(String key): Liefert den Wert, der dem angegebenen Schlüssel zugeordnet ist.
insert(String key, int value): Fügt ein Schlüssel-Werte-Paar in die Hash-Tabelle ein.
remove(String key): Entfernt das Schlüssel-Wert-Paar mit dem angegebenen Schlüssel aus der Hash-Tabelle.
myhash1(String y): Erzeugt einen Hash-Wert für die angegebene Zeichenkette durch lineares Probing.
myhash2(String y): Erzeugt einen Hash-Wert für die angegebene Zeichenkette unter Verwendung von quadratischem Probing.

ValueEntry-Klasse:

Schlüssel: Stellt den Schlüssel des Schlüssel-Wert-Paares dar.
Wert: Stellt den mit dem Schlüssel verbundenen Wert dar.
ValueEntry(String key, int value): Konstruktor zur Erstellung eines neuen ValueEntry-Objekts mit dem angegebenen Schlüssel und Wert.

App-Klasse:

main(String[] args): Der Haupteinstiegspunkt des Programms.
countEntries(String filename): Zählt die Anzahl der Einträge in der angegebenen Datei.
calculateSuitableSize(int numEntries): Berechnet anhand der Anzahl der Einträge eine geeignete Größe der Hashtabelle.
isPrime(int n): Prüft, ob die angegebene Zahl eine Primzahl ist.
readEntriesFromFile(HashTable hashTable, String filename): Liest Einträge aus einer Datei und fügt sie in die Hash-Tabelle ein.

Qellen

i. Thomas Ottmann, Peter Widmayer: "Algorithmen und Datenstrukturen". Springer Vieweg Berlin, Heidelberg. ISBN: 978-3-662-55650-4

ii. GeeksforGeeks: "Double Hashing". Verfügbar unter: https://www.geeksforgeeks.org/double-hashing/

iii. GeeksforGeeks: "Java Implement Double Hashing". Verfügbar unter: https://www.geeksforgeeks.org/java-program-to-implement-hash-tables-with-double-hashing/

iv. W3chools: "Java HashMap". Verfügbar unter: https://www.w3schools.com/java/java_hashmap.asp

v. Spring.io: "Gradle Build Tool Guide". Verfügbar unter: https://spring.io/guides/gs/gradle/