Hashtabelle Protokoll

Lorenz Rentenberger, Samuel Hammerschmidt

März 2024

Hashfunktion

Die `hashFunction(self, keyString)` ist eine Methode, die dazu dient, einen Hashwert für einen gegebenen Schlüsselstring zu berechnen. Dabei wird jeder `char` im `keyString` durchlaufen, und der Hashwert wird durch eine Multiplikation mit der Primzahl `79` und der Addition des Unicode-Werts des `char` aktualisiert. Schließlich wird der berechnete Hashwert durch die Größe der Hashtabelle modulo-dividiert, um sicherzustellen, dass er innerhalb des gültigen Indexbereichs liegt.

Kollisionserkennung

In der Methode `addStock(self, stock)` wird die Kollisionserkennung durchgeführt, um sicherzustellen, dass ein neues Element korrekt in die Hashtabelle eingefügt wird. Die Kollisionserkennung überprüft ob der Index bereits von einem anderen Element belegt ist. Wenn eine Kollision festgestellt wird, benützen wir die Quadratische Sondierung um eine alternative Position zu finden. Die Quadratische Sondierung setzt die Wahrscheinlichkeit von Kollisionen während der Sondierung herab. Dabei addieren wir eine Zahl zum Index, welche wir quadratisch erhöhen, um eine freie Stelle in der Hashtabelle zu finden.

Verwaltung der Kursdaten

Die Erfassung von Kursdaten erfolgt durch die Methode `downloadStockData(symbol)`, welche die Daten von der Yahoo Finance API im CSV-Format herunterlädt. Anschließend werden die Daten mit der Methode `importStockData(symbol)` in das System importiert. Die Speicherung der Kursdaten erfolgt mittels der Methode `saveTable(fileName)`, welche die gesamte Hashtable samt Kursdaten in einer strukturierten JSON-Datei speichert. Dadurch wird eine einfache Wiederherstellung der Daten ermöglicht. Umgekehrt erlaubt die Methode `loadTable(fileName)` das Laden der gespeicherten Daten aus einer JSON-Datei zurück in das System. Abschließend ermöglicht die Methode `plotStockData(symbol)` die grafische Darstellung der Kursdaten einer Aktie.

Löschalgorithmus

Die Methode `deleteStock(self, symbol)` dient dazu, ein Aktienelement aus der Hashtable zu entfernen. Zunächst wird der Index des zu löschenden Elements mithilfe der Hashfunktion berechnet. Wenn das Element an diesem Index vorhanden ist, wird `foundStock` auf True gesetzt, andernfalls wird eine Quadratische Sondierung durchgeführt, um das Element zu finden. Sobald das Element gefunden wurde, wird es aus der Hashtable entfernt, indem sein Wert auf `None` gesetzt wird. Falls das Element nicht gefunden wird, wird eine entsprechende Meldung ausgegeben.

Aufwandsabschätzung

Operation	**Best Case**	**Worst Case**	**Average Case**
Suchfunktion (searchStock)	$O(1)$	$O(n)$	$O(1)$
Suchfunktion (normales Array)	$O(1)$ (sortiert)	$O(\log(N))$ (sortiert) $O(n)$ (nicht sortiert)	$O(n)$ (nicht sortiert) $O(\log(n))$ (sortiert)
Suchfunktion (verkettete Liste)	$O(1)$	$O(n)$	$O(n)$
Einfügefunktion (addStock)	$O(1)$	$O(n)$	$O(1)$
Einfügefunktion (normales Array)	$O(1)$	$O(n)$	$O(n)$
Einfügefunktion (verkettete Liste)	$O(1)$	$O(n)$	$O(1)$
Löschfunktion (deleteStock)	$O(1)$	$O(n)$	$O(1)$
Löschfunktion (normales Array)	$O(1)$	$O(n)$	$O(n)$
Löschfunktion (verkettete Liste)	$O(1)$	$O(n)$	$O(n)$

Tabelle 1: Aufwandsabschätzung für verschiedene Operationen