Übung 1: Haschtable

Josef Koch, Daniel Krottendorfer

Aufgabenstellung:

Ein Datenstruktur in Form einer Hashtable programmieren die möglichst effizient Aktien ablegt und wiederfindet. Bei Kollisionen soll eine alternierende quadratische Sondierung angewandt werden. Die Haschtable soll maximal 1000 Aktien verwalten die wiederum 30 Kursdaten speichern soll.

Hashfunktion:

Für die Haschfunktion haben wir uns endschieden die einzelnen Buchstaben des Namens der Aktie aufzumultiplizieren und anschließend den Rest bei der Division durch die HTB Länge zu berechnen dann Verwenden wir alternierende quadratische Sondierung falls eine Kollision vorliegt. Die HTB darf niemals zu 100% befüllt sein.

Kollisionserkennung:

Nachdem eine Kollision aufgetreten ist wird die “dependencies” (Abhängigkeiten) variable um eins erhöht, um bei einem worst Case Szenario anzuzeigen, dass da noch etwas kommt.

Löschalgorithmus:

Zuerst wird gesucht ob der gesucht Aktienname existiert, falls ja wird die “dependencies” variable von allen vorigen Aktien verringert. Schlussendlich wird der Wert gelöscht und alle folgenden Werte werden nachgerückt.

Aufwandabschätzung:

Wir haben eine Arraygröße von 1511 gewählt, d.h. bei 1000 Elementen haben wir einen Füllgrad von 66% und im Durchschnitt 3 Zugriffe beim Suchen, Löschen und Hinzufügen.

Bei einem normalen Array hätte man beim Einfügen und Löschen je nach Methode einen Aufwand von 1 bis n (oder n/2).

Beim ganz normalen suchen n/2 (500) und falls das Array geordnet ist und man eine binäre Such anwenden kann dann ist der Aufwand immerhin noch maximal log2(n+1) (10).

Bei einer einfach verketteten liste wäre ein binäre Such nicht ohne weiteres möglich.