**Aufwandsabschätzung:**

**Hashtabelle:**

Einfügen: Im Durchschnitt O(1), im schlimmsten Fall O(n)

Suchen: Im Durchschnitt O(1), im schlimmsten Fall O(n)

Löschen: Im Durchschnitt O(1), im schlimmsten Fall O(n)

Der Füllgrad der Hashtabelle hat einen großen Einfluss auf die Laufzeit. Wenn die Tabelle zu voll ist, kann die Suche oder das Einfügen aufgrund von Kollisionen (mehrere Schlüssel, die auf den gleichen Hashwert abgebildet werden) langsamer werden. Wir haben einen Füllgrad von 50% da wir eine Größe von 2000 haben aber nur 1000 Aktien maximal speichern.

Für eine Hashtabelle mit einer geeigneten Größe und einer guten Hashfunktion kann das Einfügen, Suchen und Löschen in O(1) durchgeführt werden. Wenn die Tabelle jedoch zu voll wird, kann die Leistung aufgrund von Kollisionen leiden. Zur Lösung von Kollisionen haben wir die quadratische Sondierung eingebaut.

**Normales Array:**

Einfügen: O(n)

Suchen: O(n)

Löschen: O(n)

Für ein normales Array muss der gesamte Speicherplatz umsortiert werden, wenn ein Element eingefügt oder gelöscht wird. Das Suchen ist zwar effizient möglich (O(1)), aber nur, wenn der Index des gesuchten Elements bekannt ist.

Für 1000 Aktien würde die Anzahl der notwendigen Operationen für das Einfügen, Suchen und Löschen in einem normalen Array im schlimmsten Fall O(n) betragen.

**Einfach verkettete Liste:**

Einfügen: O(1)

Suchen: O(n)

Löschen: O(n)

Das Einfügen in eine einfach verkettete Liste ist einfach, da nur ein neues Element hinzugefügt werden muss und ein paar Verbindungen aktualisiert werden müssen. Die Suche und das Löschen können jedoch langsam sein, da man sich von Anfang an durch die Liste arbeiten muss.

Für 1000 Aktien würde die Anzahl der notwendigen Operationen für das Einfügen in eine einfach verkettete Liste im Durchschnitt O(1) betragen. Die Anzahl der notwendigen Operationen für die Suche und das Löschen würde jedoch im schlimmsten Fall O(n) betragen.

Insgesamt lässt sich sagen, dass die Hashtabelle im Durchschnitt am effizientesten ist, gefolgt von der einfach verketteten Liste und dem normalen Array im schlimmsten Fall. Die Laufzeitkomplexität wird in der O-Notation ausgedrückt, wobei O(1) für konstante Laufzeit, O(n) für lineare Laufzeit und O(n^2) für quadratische Laufzeit steht.

**Wie ist der Aufwand allgemein nach O-Notation definiert?**

Die O-Notation wird verwendet, um die Komplexität von Algorithmen zu beschreiben. Sie gibt an, wie schnell die Laufzeit des Algorithmus im Verhältnis zur Größe der Eingabe wächst. Dabei wird nur der Einfluss des größten Faktors betrachtet.

**Beschreibung Datenstruktur:**

**Hashfunktion**: Hierfür haben wir den ASCII-Wert jedes Buchstabens des Namens bzw. des Kürzels einer Aktie genommen und mit 7^(n-1) gerechnet.

**Kollisionserkennung**: Es wird geschaut, ob das Feld schon einen Wert hat. Wenn das der Fall ist wird mit der quadratischen Sondierung das nächste freie Feld gesucht.

**Verwaltung der Kursdaten**: In der Hashtabelle wird für jede Aktie der Name der dazugehörige csv-Datei gespeichert und dadurch ist es möglich auf die Kursdaten zuzugreifen.

**Löschalgorithmus**: Es wird nach dem Namen der zu löschenden Aktie gesucht und als Platzhalter bei der WKN wird „delete“ gespeichert werden, damit das Programm weißt, dass die Aktie gelöscht ist, aber das Hashen der anderen Aktien immer noch funktioniert.