Sarah Köhler
Dora Szücs
15 Jun 2014

Blatt 8

Aufgabe 2.2. Hashfunktionen beurteilen

Funktionsweise

Die gegebene Hash-Funktion berechnet aus vier Attributen des Objekts den Hash-Wert. Jedes Objekt bildet ein Fahrzeug ab, das durch die Anzahl der Reifen und die PS definiert wird sowie die Eigenschaften auf Schienen fahren zu können bzw. zu fliegen.

Der Hash-Wert, d.h. der Index, wird dabei durch Multiplikation bzw. Addition der Werte und niedriger gerader Ganzzahlen gebildet.

Kollisionen

Anhand zweier Beispiele lässt sich bereits erahnen, dass das Verfahren nicht optimal funktioniert. Kollidierende Instanzen wären nach dem geschilderten Algorithmus etwa die folgenden Beispiele:

Vehicle	wheelsWithTires	canDriveOnTracks	canFly	horsePower	Hashwert int	Hashwert byte
Heißluftballoon	0	FALSE	TRUE	0	2	2
F1 Rennwagen	4	FALSE	FALSE	750	6004	116
PKW	4	FALSE	FALSE	110	884	116
Fahrrad	2	FALSE	FALSE	0	2	2
Tretroller	2	FALSE	FALSE	0	2	2
Quad	4	FALSE	FALSE	110	884	116
Tretboot	0	FALSE	FALSE	0	0	0
Floß	0	FALSE	FALSE	0	0	0
Kanu	0	FALSE	FALSE	0	0	0

Ergebnis sind diverse Kollisionen: Beispielsweise kollidieren alle Fahrräder, bzw. ummotorisierten Zweiräder. Für eine Menge von Fahrrädern wäre die Hash-Funktion beispielsweise völlig ungeeignet. Ähnlich verhält es sich für die große Menge motorisierter Kraftfahrzeuge mit vier Rädern, wie verschiedene PKW, LKW, Traktoren etc, die bei gleicher PS-Zahl jeweils den gleichen Hash-Wert aufweisen.

Bewertung

Besonders die Bedingung, dass eine möglichst gleichmäßige Verteilung der Indizes auf die Schlüssel erfolgen sollte, ist verletzt. So werden, wenn man von realistischen Daten ausgeht, hauptsächlich gerade Zahlen als Ergebnisse auftreten, während ungerade Ergebnisse sehr selten sein sollten: Der PS-Wert kann zwar ebenso gerade wie ungerade sein, durch Multiplikation mit 8 erhält man im ersten Schritt aber auf jeden Fall eine gerade Zahl als Ergebnis. Dies ändert sich auch in den beiden folgenden Schritten nicht, egal welche Werte für die Eigenschaften canFly und canDriveOnTracks gesetzt sind, da jeweils entweder der Wert unverändert verbleibt oder mit einer geraden Zahl (4 bzw. 2) addiert wird. Nur der letzte Schritt könnte das Ergebnis auch zu einer ungeraden Zahl machen, allerdings würde das in der Praxis wohl fast nie vorkommen. Denn zuletzt wird das bisherige Ergebnis zu der Anzahl der Reifen addiert, welche für fast alle Fahrzeugtypen gerade (bzw. null) ist. Das heißt nur für sehr spezielle Datenmengen, wo Fahrzeuge mit einer geraden und ungeraden Anzahl von Reifen in etwa im Verhältnis 1:1 enthalten sind, wäre mit einer Gleichverteilung von geraden und ungeraden Hash-Werten zu rechnen. Für eine realistisch erwartbare Menge, etwa nur Autos oder Lokomotiven oder eine Mischung ganz verschiedener Fahrzeugtypen ist immer damit zu rechnen, dass überwiegend die geraden Indizes (oder auch ungeraden, etwa bei einer Menge von Trikes) belegt werden, während die anderen frei bleiben.

Das heißt besonders die ersten drei Schritte (d.h. die Zeilen 10, 12 und 15) sorgen dafür, dass die Hash-Werte zu gleichförmig bleiben. Der letzte Schritt wäre theoretisch besser geeignet, jedoch ist hier in der Praxis mit einer geringen Variationsbreite zu rechnen: Viele Fahrzeugtypen haben keine Reifen und diejenigen mit Reifen haben meist eine Anzahl im niedrigen Bereich, die zudem meist ein gerader Wert ist.

Die PS-Zahl ist als Ausgangswert etwas besser, jedoch ist auch hier mit gewissen Häufungen zu rechnen, etwa im Bereich um 100. Auch die Multiplikation mit 8 erzeugt große Lücken in der Hashtabelle: Da die PS-Angabe ein ganzzahliger Wert sein muss,

können viele Zahlen gar nicht erreicht werden, beispielsweise alle Werte zwischen 1 und 7. Andere nur schwer, wenn auch theoretisch auf Grund der anderen drei Schritte. Trotzdem ist mit Häufungen im Bereich der Vielfachen von 8 zu rechen, denn die beiden bedingten Ausdrücke werden praktisch wohl nur selten aufgerufen, da nur wenige Fahrzeuge flugfähig sind oder auf Schienen fahren können.

Auch der Cast von int zu byte ganz am Ende kann das Ergebnis verschlechtern, wenn etwa nur sehr große Werte für PS in der Datenmenge vorkommen, etwa weil nur Flugzeuge aufgelistet werden. Dort kann es passieren, dass der informative Teil der Zahl durch den Cast abgeschnitten wird und die Nullen am Ende übrig bleiben (bzw. eine 2 im Fall einer Menge von Flugzeugen).

Verbesserungsmöglichkeit

Verbessern könnte man den Algorithmus, indem statt der Multiplikation mit 8 der erste Schritt den Teilungsrest der Ps-Werte und einer Primzahl berechnet. Dabei sollte der genaue Wert der Primzahl von den zu erwartenden Eingabedaten (Bandbreite der PS-Werte) und der gewünschten Größe der Hashtabelle abhängen. Die Modulo-Operation sollte die Werte besser verteilen. Allerdings ist die Funktion dann immer noch ungeeignet um Fahrzeuge des gleichen Typs, etwa mehrere Fahrräder, auf eine Hashtabelle zu verteilen.

Deswegen sollte ein anderer (zusätzlicher) Parameter herangezogen werden, etwa ein Wert, der sich aus den Namen berechnet.