Implementation des ADT Set

Finn Jannsen, Philipp Schwarz

1. April 2019

Inhaltsverzeichnis

1 Einführung

Diese Dokumentation beschreibt drei Implementations-Varianten des Abstrakten Datentyps Set. Dieser Datentyp soll eine Menge darstellen. Als Vorgabe zur Art der Implementation wurde 1. ein Array von Elementen, 2. ein Array von Containern und 3. eine verkettete Liste von Containern angeführt. In Abschnitt 2 wird darauf eingegangen, wie die verschiedenen Varianten realisiert wurden. Anschließend wird in Abschnitt 3.1 geprüft, ob die vorgegebenen Operationen funktionieren und in Abschnitt 3.2 die Performance der Varianten verglichen.

2 Implementation

2.1 Set als Array

Implementation beschreiben

2.2 Set als Array von Containern

Implementation beschreiben

2.3 Set als einfach verkettete Liste von Containern

Implementation beschreiben

3 Verifizieren und Testen

3.1 Verifizieren der Funktionalität

Da die drei Varianten sich nur intern im Aufbau unterscheiden und dies nach außen hin gekapselt ist, verifizieren wir ihre Funktion anhand einer Menge an gleichen Tests. Die in Tabelle 1 enthaltenen Tests soll eine ausreichende Verifizierung der Operationen auf den Sets ermöglichen.

Die in der Tabelle 1 aufgeführten Tests wurden mit verschiedenen Parametern und Quantitäten in verschiedenen Sequenzen in JUnit4 getestet. Alle drei Implementations-Varianten erfüllen die oben genannten Tests mit pre- und post-condition.

3.2 Aufwandsanalyse

T1	$add : SET \times ELEMENT \rightarrow SET \times POS$
Beschreibung	Hinzufügen eines Elements $e \in ELEM$ in die Menge $s \in SET$
pre-condition	-
post-condition	`s.find(e.key).isValid = TRUE`

T2	$delete : SET \times POS \rightarrow SET$
Beschreibung	Entfernen eines Elements $e \in ELEM$ in Position $p \in POS$ aus der Menge $s \in SET$
pre-condition	`p.getSet() == s && s.retrieve(p) == e && p.isValid = TRUE`
post-condition	`p.isValid = FALSE`

T3	$delete : SET \times KEY \rightarrow SET$
Beschreibung	Entfernen eines Elements $e \in ELEM$ in Position $p \in POS$ mit Schlüssel $k \in KEY$ aus der Menge $s \in SET$
pre-condition	`s.find(k).getSet() == s && s.retrieve(s.find(k)) == e && s.find(k).isValid = TRUE`
post-condition	`p.isValid = FALSE`

T4	$find : SET \times KEY \rightarrow POS$
Beschreibung	Suchen der Position $p \in POS$ eines Elements $e \in ELEM$ mit Schlüssel $k \in KEY$ aus der Menge $s \in SET$
pre-condition	-
post-condition	Wenn $k \in s$: `s.find(k).isValid = TRUE`. Wenn $k \notin s$: `s.find(k).isValid = FALSE`

T5	$retrieve : SET \times POS \rightarrow ELEM$
Beschreibung	Zugriff auf Element $e \in ELEM$ in Position $p \in POS$ aus der Menge $s \in SET$
pre-condition	`p.getSet() == s && p.isValid = TRUE`
post-condition	`s.retrieve(p) == e`

T6	$size : SET \rightarrow INTEGER$
Beschreibung	Mächtigkeit der Menge $s \in SET$
pre-condition	-
post-condition	`s.size() == |s|`

T7	$unify : SET \times SET \rightarrow SET$
Beschreibung	Vereinigung zweier Mengen $s_1, s_2 \in SET$
pre-condition	-
post-condition	`s.unify(s1, s2) ==` $s_1 \cup s_2$

Tabelle 1: Verifikationstests

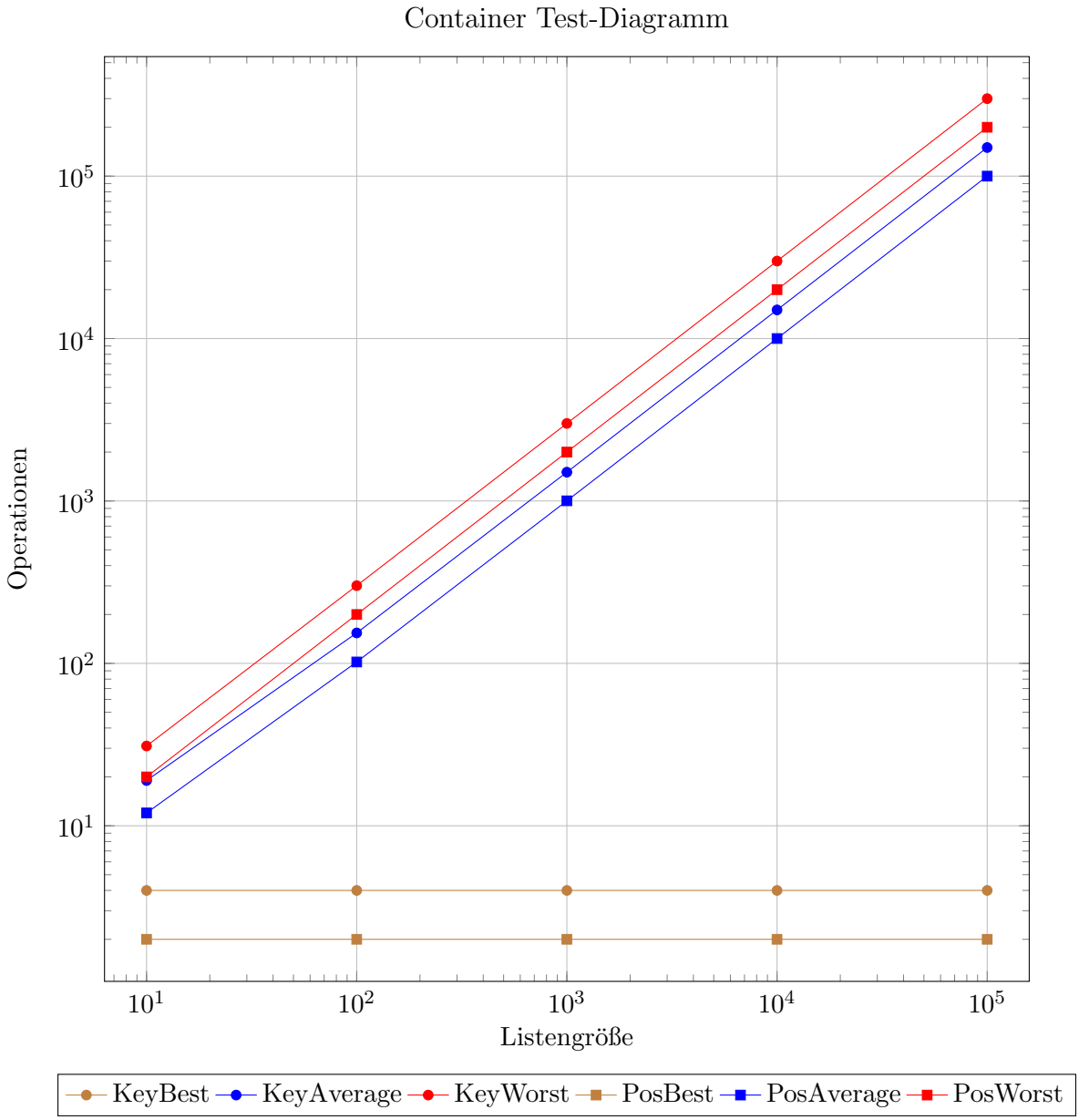

Abbildung 3.1: delete(Pos) und delete(Key) in Container-Liste