

Implementation des ADT Set

Finn Jannsen, Philipp Schwarz

1. April 2019

Inhaltsverzeichnis

1	Einführung	1
2	Implementation	2
2.1	Set als Array	2
2.2	Set als Array von Containern	2
2.3	Set als einfach verkettete Liste von Containern	2
3	Verifikation	2
3.1	Set als Array	2
3.2	Set als Array von Containern	2
3.3	Set als einfach verkettete Liste von Containern	2
4	Aufwandsanalyse	2
4.1	Set als Array	2
4.2	Set als Array von Containern	2
4.3	Set als einfach verkettete Liste von Containern	2

1 Einführung

Diese Dokumentation beschreibt drei Implementations-Varianten des Abstrakten Datentyps Set. Dieser Datentyp soll, sofern möglich, 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 prüfen wir in Abschnitt 3, ob die vorgegebenen Operationen funktionieren und vergleichen in Abschnitt 4 die Performance der Varianten.

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 Verifikation

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 folgende Liste an Tests soll eine ausreichende Verifizierung der Operationen auf den Sets ermöglichen:

3.1 Set als Array

Verifizieren

3.2 Set als Array von Containern

Verifizieren

3.3 Set als einfach verkettete Liste von Containern

Verifizieren

4 Aufwandsanalyse

4.1 Set als Array

Verifizieren

4.2 Set als Array von Containern

Verifizieren

4.3 Set als einfach verkettete Liste von Containern

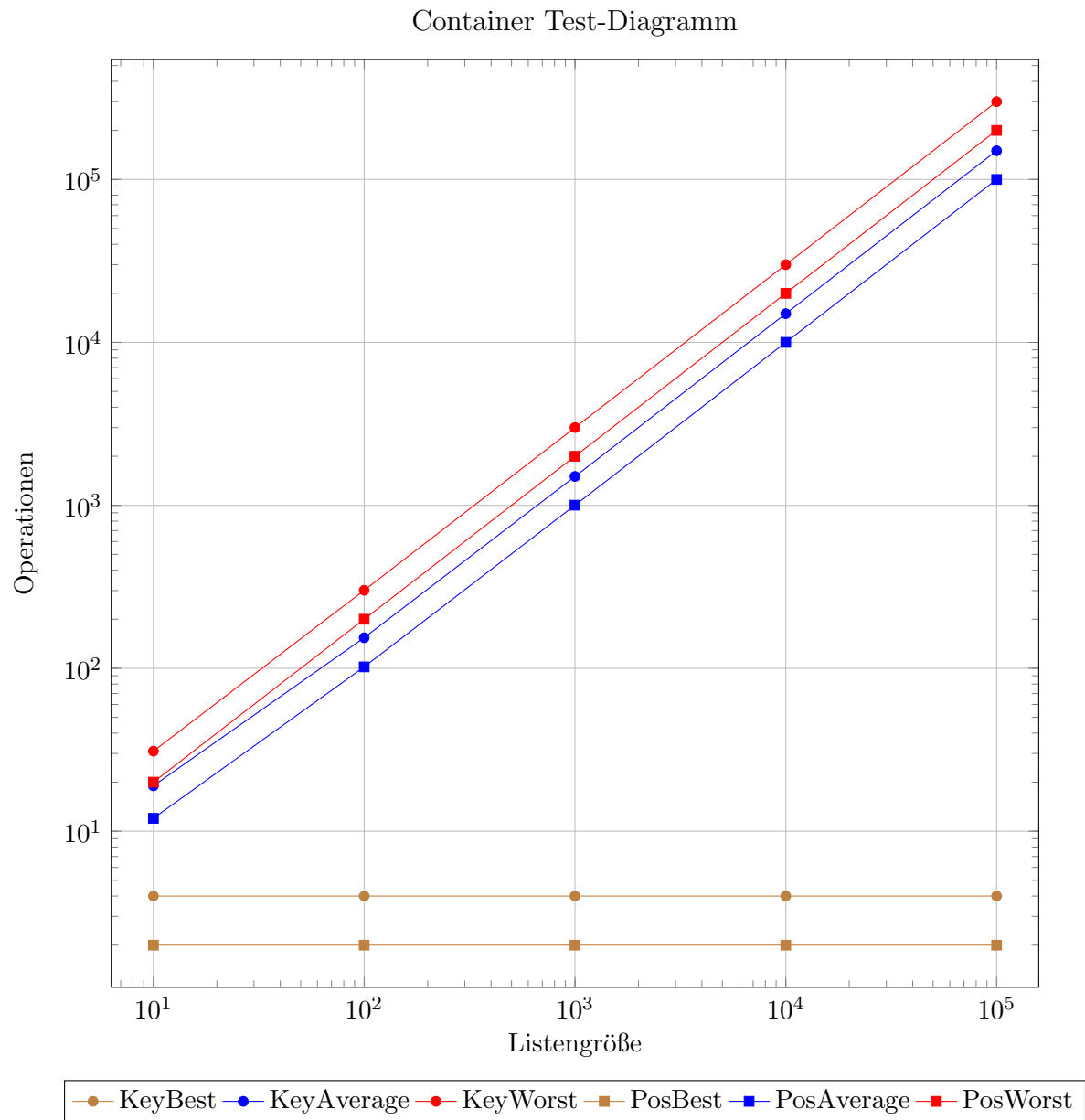


Abbildung 4.1: delete(Pos) und delete(Key) in Container-Liste

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	$s.unify(s1, s2) == s_1 \cup s_2$