ADP Aufgabe 1 Ergebnisse, Abgabe 1

Team 1
Hugo Protsch, Justin Hoffmann

November 16, 2020

1 Formales

1.1 Aufgabenaufteilung

Der Code wurde zusammen entwickelt.

1.2 Quellenangaben

Es wurden lediglich Vorlesungsmaterialien verwendet.

1.3 Bearbeitungszeitraum

Für die Bearbeitung und Überarbeitung des Entwurfs haben wir in etwa 10 bis 12 Stunden benötigt. Für die Entwicklung des Quellcodes und die Laufzeitanalyse haben wir in etwa 15 bis 20 Stunden benötigt.

1.4 Aktueller Stand

Der Quellcode ist funktionsfähig und wurde auf Laufzeit überprüft.

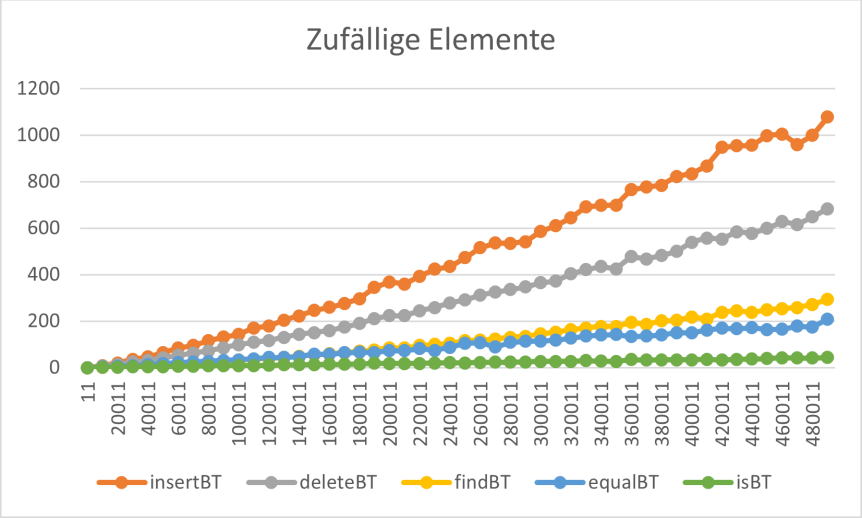
1.5 Änderungen des Entwurfes

– nicht zutreffend –

2 Laufzeitmessung

Im Folgenden werden Vermutungen zur Laufzeitkomplexität aufgestellt. Diese sind ohne Regression jedoch sehr spekulativ. Beispielsweise kann zwischen eine logarithmischen und einer linearen Kurve in der Darstellung schwer unterschieden werden. Dies ist im Folgenden zu bedenken.

2.1 Zufällig



2.1.1 Einstellungen

Die Folgenden Einstellungen wurden bei der Messung verwendet:

- 0 Startelemente
- 1000 Elemente Schrittgröße
- 20 Schritte
- Mitteln über 5 Messungen

2.1.2 Ergebnisse

InsertBT Bei InsertBT ist die längste Laufzeit zu erkennen, da wir den Baum zum Anhängen immer bis zum Ende durchlaufen müssen. Außerdem ist bei einer höheren Elementanzahl ein Anstieg der Steigung zu erkennen. Dies könnte auf eine höhere Laufzeit als linear hindeuten. Erwartet war eine logarithmische Laufzeit. Demnach liegen wir unter den Erwartungen.

DeleteBT DeleteBT hat bei zufälligen Elementen eine dezent kürzere Laufzeit als InsertBT, da wir zum Entfernen eines Elementes nicht zwangsläufig bis zum Ende Laufen müssen. Falls jedoch im Knoten des zu löschenden Elements sowohl ein linker, als auch einen rechter Teilbaum existiert, muss

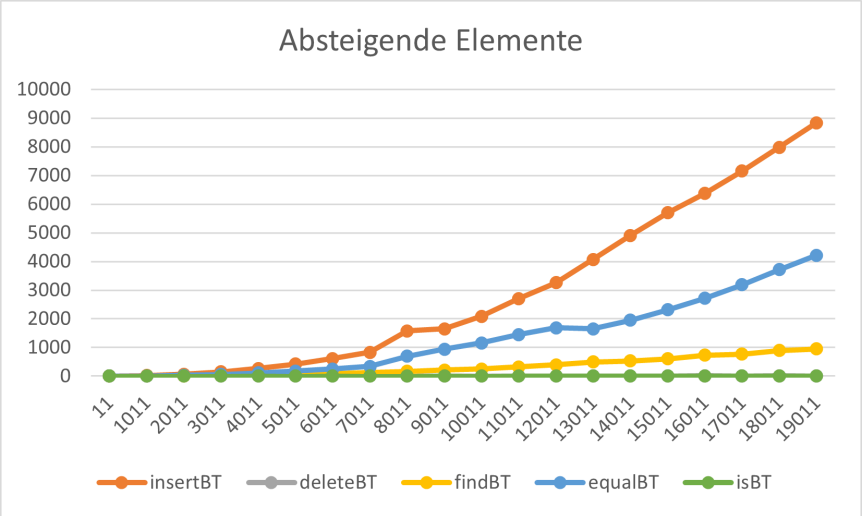
der Baum bis zum größten Element des linken Teilbaums traversiert werden, um das Ersatzelement zu finden. Hier lässt sich ein linearer Anstieg erkennen. Dies entspricht demnach dem erwarteten Worst-Case mit einer linearen Komplexität.

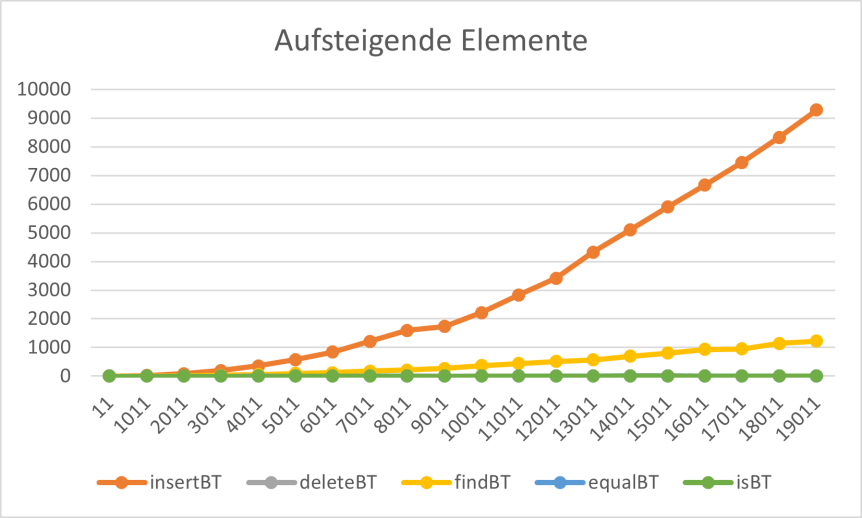
FindBT FindBT hat eine deutlich kürzere Laufzeit als DeleteBT und InsertBT, da der Baum in jedem Fall nur bis zum gesuchten Element traversiert wird. Hier haben wir also eine deutlich flachere Kurve. Das könnte auf eine logarithmische Laufzeit hindeuten, wie im Entwurf erwartet.

EqualBT Auch EqualBT hat eine sehr flache Kurve. Logarithmische Laufzeit. Schneller als erwartet.

IsBT IsBT hat mit Abstand die kürzeste Laufzeit von allen Funktionen. Erwartet war eine lineare Laufzeit, wir liegen also deutlich über unseren Erwartungen.

2.2 Aufsteigend und Absteigend





2.2.1 Einstellungen

Die Folgenden Einstellungen wurden bei der Messung verwendet:

- 0 Startelemente
- 10000 Elemente Schrittgröße
- 50 Schritte
- Mitteln über 10 Messungen

2.2.2 Ergebnisse

InsertBT Bei InsertBT vermuten wir sowohl mit absteigenden Elementen, als auch mit aufsteigenden Elementen eine quadratische Laufzeit, da ein starker Anstieg der Steigung bei einer höheren Elementanzahl sichtbar ist.

FindBT FindBT besitzt wie erwartet eine sehr flache Kurve. Unterschiede zum Average-Case sollte es hier nicht geben. In beiden Fällen wird der Baum bis zum gesuchten Element traversiert.

DeleteBT Die Laufzeit von DeleteBT hat bei den Extremfällen eine deutlich flachere Kurve, als bei zufällig angeordneten Elementen. Wir gehen davon aus, dass dies nicht die tatsächliche Laufzeit wiederspiegelt. Erwartet

war die gleiche Laufzeit wie bei FindBT, da in einem Baum mit aufsteigenden bzw. absteigenden Element stets nur bis zum zu löschenden Element traversiert werden muss. Dies könnte sowohl an nicht gefundenen Fehlern der Implementation, als auch an Fehlern in der Messung selbst liegen.

IsBT, EqualBT IsBT und EqualBT sollten bei beiden Grenzfällen die gleiche Laufzeit besitzen wie bei einer zufälligen Anordnung, da alle Knoten durchlaufen werden müssen. Dies lässt sich nicht in den Ergebnissen erkennen, da das gleiche Problem auftritt wie bei DeleteBT.