Randomisierte Suchalgorithmen

Fragile Complexity in der Praxis

Julian Lorenz

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

1 Einleitung

Ein fiktives Land möchte seinen besten Boxer für die dieses Jahr anstehenden olympischen Spiele bestimmen. Hierfür soll ein Turnier, analog zu einem Binärbaum, nach K.O.-Runden Prinzip ausgetragen werden. Im Anschluss ist zwar der beste Boxer der Gewinner des Turniers, musste dafür jedoch eine Vielzahl an Kämpfen bestreiten und weist eine entsprechende Erschöpfung oder gar einschränkende Verletzungen auf.
Jeder Kampf des Turniers kann als Vergleich zwischen zwei Elementen aufgefasst werden. Sollte ein bestimmtes, für den Anwender unter Umständen wichtiges, Element wiederholt aus dem Speicher geladen werden, so könnte sich dieser im Laufe der Zeit schneller abnutzen. Insofoern besteht ein Interesse daran, die Abnutzung einzelner Elemente aktiv regulieren zu können.

2 Fragile Complexity

Definition. *Ein vergleichsbasierter Algorithmus $\mathcal{A}$ hat eine fragile complexity von $f(n)$, falls jedes Eingabeelement an maximal $f(n)$ Vergleichen teilnimmt. Insbesondere besitzt ein Element e bezüglich eines Algorithmus $\mathcal{A}$ eine fragile complexity von $f_e(n)$, falls e bei der Ausführung von $\mathcal{A}$ an maximal $f_e(n)$ Vergleichen teilnimmt.*

Die Algorithmen RMinimum und RMedian wurden zuerst im Paper *Fragile Complexity of Comparison-Based Algorithms* vorgestellt und erlauben das Auffinden des Minimum- bezeihungsweise Median-Elements in einer gegebenen Menge. Im Verlaufe einer Bachelorarbeit wurden beide Algorithmen in python implementiert und für relevante Eingabeparameter ausgewertet. Hierbei konnte eine Auswahl der vorgestellten Abschätzungen empirisch bestätigt werden.

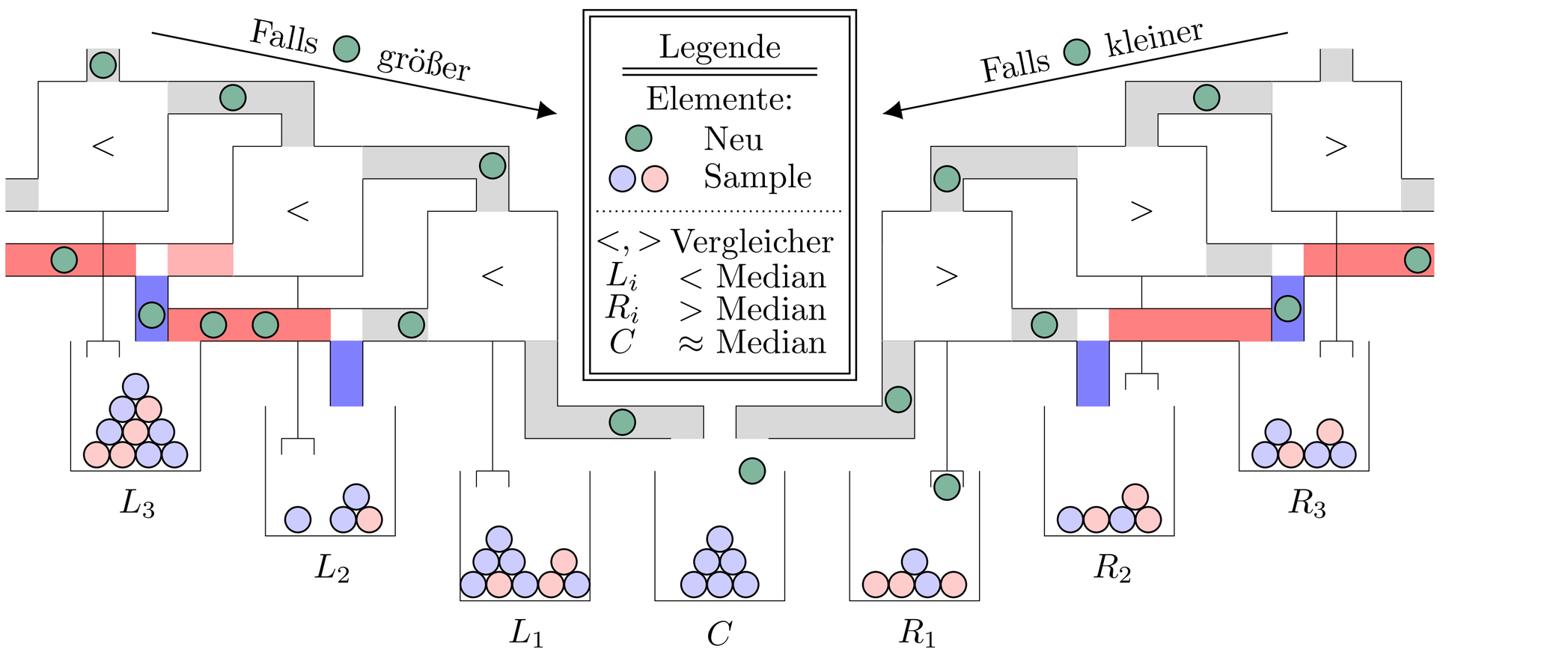

Figure 1: *RMedian*, Schritt 2: Einteilung neuer Elemente durch Vergleiche mit Elementen aus den Buckets L_i, C, R_i

Zunächst wird ein randomisiertes Sample gewählt und sortiert. Hieraus werden nun repräsentative Mengen für Elemente kleiner, größer oder in der Nähe des Medians erwartet. Wie in Abbildung 1 zu sehen, werden nun alle weiteren Elemente nacheinander mit Elementen der einzelnen Buckets verglichen und entsprechend einsortiert.
Die Mengen L_i beinhalten hierbei jene Elemente, die kleiner als der Median vermutet werden und die Mengen R_i analog Elemente größer als der Median. In der Menge C befinden sich abschließend alle Mediankandidaten.

3 RMinimum & RMedian

Beide Algorithmen erhalten als Eingabe eine total geordnete Menge X mit Mächtigkeit n sowie einen sogenannten *Tuningparameter* $k(n)$, mit dessen Hilfe sich ein *Trade-off* zwischen der erwarteten *Fragile Complexity* $f_{min}(n)$ des Minimums beziehungsweise f_{med} des Medians sowie der erwarteten *Fragile Complexity* $f_{rem}(n)$ aller übrigen Elemente regulieren lässt. Eine elementare Idee beider Algorithmen ist es hierbei eine randomisierte Teilmenge der Eingabemenge zu wählen und diese als Sample für weitere Vergleiche zu nutzen. Durch ihren Aufbau nehmen Elemente bei zunehmender Distanz zum Minimum bzw. Median häufiger an Vergleichen teil. Die Analyse der Arbeit beschränkt sich im wesentlichen auf gegebene Abschätzungen der Paare $< \mathbb{E}[f_{min/med}], \mathbb{E}[f_{rem}] >$.

Theorem 1. *Sei $k(n) = \log(n)/\log\log(n)$. Dann benötigt* RMinimum $\mathbb{E}[f_{min}] = \mathcal{O}(\log(n)/\log\log(n))$ *Vergleiche für das Minimum Element und $\mathbb{E}[f_{rem}] = \mathcal{O}(\log(n)/\log\log(n))$ für alle übrigen Elemente.*

Wie in Abbildung 2 zu sehen konnte für alle nicht-median Elemente ein passender *Fit* der Form $F(n) = a \cdot \log(n)/\log\log(n) + b$ gefunden werden.

Figure 2: Fit für f_{min} für RMinimum

Für beide Algorithmen konnte zudem bestätigt werden, dass die Anzahl der insgesamt benötigten Vergleiche linear in der Eingabegröße ist.