Algorithmen und Datenstrukturen I

Blatt 5 (Abgabe 1)

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 01.12.2019 16:35

Lesen Sie vor der Bearbeitung `https://ob.cs.hm.edu/exercises.html` **und stellen Sie sicher, dass es *keine* Commits von Ihnen im `master`-Branch gibt.** Nachdem Sie dieses Blatt alleine bearbeiten, reicht es einen `develop`-Branch zu erzeugen und nur in diesen zu committen.

Aufgabe 1 — Template, Linked List, Quicksort

Das Repository für diese Aufgabe bekommen Sie unter `https://classroom.github.com/a/-mTzBURG`.

1. Implementieren Sie die einfach verkettete Liste als Template wie im Repository vorgegeben.
2. Ergänzen Sie die vorhandenen Googletest sinnvoll.
3. Schließen Sie Ihre Arbeiten an diesem Blatt, durch Erzeugen des Pull-Requests, bis zum **25.11.2019, 08:00 Uhr** ab.

Ein Listenelement besteht aus einem **eindeutigen** Schlüssel vom Typ `K` und einem Wert vom Typ `V`. Die Elemente sind einfach verkettet. `head` ist ein Pointer auf das erste Listenelement. Die Methode `search` gibt den Wert zu einem Schlüssel oder `nullptr` zurück. Die Methode `isEmpty` gibt zurück ob die Liste leer ist. Die Methode `popHead` gibt das erste Listenelement als Schlüssel-Wert-Paar zurück und entfernt es aus der Liste.

Die Methode `sort` sortiert die Liste nach dem **Quicksort-Algorithmus** mit Hilfe der übergebenen `lessThan`-Funktion. Dokumentieren Sie im Quellcode dazu den Aufwand in Abhängigkeit der Problemgröße N. Zur Umsetzung des Quicksort-Algorithmus dürfen Sie keine zusätzlichen Listenelemente erzeugen, sondern nur die bereits vorhandenen

neu verlinken. Es ist außerdem nicht erlaubt die Elemente mit Hilfe einer anderen Datenstruktur, wie z.B. einem `vector`, zu sortieren. Die Liste muss *in-situ* sortiert werden. Die Methode `isSorted` gibt `true` zurück, wenn Sie nach der übergebenen Funktion sortiert ist und `false` sonst.

Mit dem Operator `+=` können Listenelemente eingefügt werden. Wenn der einzufügende Schlüssel bereits in der Liste vorhanden ist, wird das vorhandene Listenelement an dieser Position durch das neue ersetzt. Wenn der Schlüssel noch nicht vorhanden ist, wird das neue Element ans Ende der Liste angefügt. Mit dem Operator `-=` entfernen Sie das Element mit dem übergebenen Schlüssel. Wenn es nicht vorhanden ist, bleibt die Liste unverändert.

Tipps/Anmerkungen

1. Wenn Sie in Ihrer Implementierung an irgendeiner Stelle ein neues Element erzeugen wollen, ist das in einem Template nicht straight forward, da `element` abhängig ist von der `list<K,V>` also ein sog. *dependent type*, der erst bei der Template-Spezialisierung spezifiziert wird. Um dies dem Compiler anzuzeigen, dient das Schlüsselwort `typename`. Dahinter müssen Sie den gesamten qualifizierten Namen des dependent types angeben, also z.B.:

   ```
   auto toInsert = new typename list<K, V>::element(t, nullptr);
   ```

2. Zur Übergabe der `lessThan`-Funktion, übergeben Sie entweder einen λ-Ausdruck oder einfach den Funktionsnamen (Beispiel in den bereits vorhandenen Tests). In der zu definierenden Methode können Sie `lessThan` dann einfach nutzen, wie eine ganz normale Funktion, also z.B.

   ```
   bool xIsLessThanY = lessThan(x, y);
   ```