Algorithmen und Datenstrukturen

Dr. Martin Moritz Kleine

Aufgabenblatt 1 : (Experimentelle) Laufzeitaufwandsbestimmung

Implementierung von einfachen Algorithmen in JavaScript. Wählen Sie geeignete Testdaten, so dass die Tests und Experimente nicht zu lange dauern, aber dennoch Aussagekraft besitzen. Die Ausführungszeit der Tests und Experimente für alle Aufgaben des Blattes gesamt soll im Sekundenbereich liegen (bezogen auf die Ladezeit des Lösungs-HTML-Dokuments auf einem modernen PC mit einem aktuellen Chrome).

Übungsaufgabe 1.1: Minimale Anzahl von Schritten. Gegeben seien ein Startpunkt $x \in \mathbb{N}$ und ein Zielpunkt $y \in \mathbb{N}$ auf einer Geraden, sowie ein Schrittmaß $d \in \mathbb{N}$. Gesucht ist die Anzahl der Schritte $steps(x, y, d) \in \mathbb{N}$, die benötigt werden, um einen Punkt zu erreichen, dessen Wert größer als y ist.

Beispiel: Bei der Eingabe $x = 15$, $y = 80$, $d = 30$ sind drei Schritte nötig.

1. Welchen Zeit- und welchen Platzaufwand hat eine naive Implementierung?
2. Implementieren Sie die Funktion $steps(x, y, d)$, welche die gesuchte Anzahl der Schritte mit konstantem Zeit- und Platzaufwand ($O(1)$) berechnet.

Übungsaufgabe 1.2: Die Fibonacci-Zahlen. Gegeben sei die folgende Funktion:

$$fib(n) = \begin{cases} 1, & n \leq 1 \\ fib(n-1) + fib(n-2) & n > 1 \end{cases}$$

1. Implementieren Sie fib direkt (also naiv).
2. Bestimmen Sie experimentell den Zeitaufwand von fib und stellen Sie das Ergebnis graphisch dar. Geben Sie die gemessene Laufzeitklasse möglichst präzise in der O-Notation an.
3. Die Fibonacci Zahlen können auch schneller berechnet werden. Schreiben Sie jetzt zwei verbesserte Funktionen $fib2$ und $fib3$, die rekursiv mit linearem Zeitaufwand ablaufen ($O(n)$).

   $fib2$ soll *memoization* verwenden.

   $fib3$ soll ohne *memoization* auskommen.
4. Bestimmen Sie experimentell den jeweiligen Zeitaufwand von $fib2$ und $fib3$ und stellen Sie die Ergebnisse graphisch dar.
5. Erläutern Sie, ob und wie sich Ihre Implementierung $fib2$ und $fib3$ im Platzaufwand unterscheiden.
6. Implementieren Sie $fib4$ als iterative Version von $fib3$.

Alle Versionen $fib1$, ... $fib4$ müssen durch Tests abgedeckt sein.

Übungsaufgabe 1.3: Verkettete und Sequentielle Listen. Unsere Listen sollen folgende Methoden unterstützen:

- Ein Element vorne anfügen. $cons(x, list)$
- Vorderstes Element zurückgeben. $head(list)$
- Restliste zurückgeben. $tail(list)$
- Länge einer Liste bestimmen. $length(list)$
- Ist die Liste leer? $isempty(list)$
- Ein Element nach n Elementen einfügen. $insert(x, n, list)$

  Dabei bedeutet nach $n = 0$ Elementen einfügen, das man das Element am Anfang einfügt. (Klären Sie, was passieren soll, wenn n größer als die Anzahl der Listenelemente ist!)

1. Implementieren Sie eine einfach verkettete Liste.
2. Messung des Zeitaufwandes: Wir nehmen plausiblerweise an, dass der Zeitaufwand einer Listenoperation in dieser Implementation proportional zur Anzahl der Dereferenzierungen ist. Fügen Sie Messung des Zeitaufwandes Hilfsvariablen ein, die mitzählen, wie oft Referenzen "verfolgt" wurden. Implementieren Sie Methoden, um diese Zähler abzufragen, zurücksetzen usw.
3. Ein ersten Experiment: Fügen Sie nacheinander die Werte i von 1 bis n in eine leere Liste ein – jeweils am Anfang der Liste. Bestimmen Sie den Zeitaufwand mit obigen Zählern. Ist das gemessene Ergebnis der zu erwartende Wert?
4. Wie zuvor, nur dass Sie am Ende einfügen.
5. Noch ein Experiment: Fügen Sie nacheinander die Werte i von 1 bis n in eine leere Liste ein, wobei das Element i an eine *zufällige* Position zwischen 0 und $i-1$ eingefügt werden soll.

   Führen Sie das Experiment t-mal durch und bestimmen Sie den Mittwelwert für den Zeitaufwand.

   Tragen Sie den mittleren Zeitaufwand als Funktion gegen die Anzahl t der Versuche auf.

   Vergleichen Sie die Ergebnisse mit den beiden vorherigen Experimenten!
6. Implementieren Sie eine sequentielle (Array-basierte) Liste.
7. Erweitern Sie Ihre Implementierung der sequentiellen Liste mit geeigneten Zählern zur Bestimmung dest Zeitaufwands der angebotenen Operationen.
8. Führen Sie die mit der verketteten Liste durchgeführten Experimente mit der sequentiellen Liste durch.
9. Vergleichen Sie die beiden Implementierungsvarianten der Liste anhand der Ergebnisse der oben geforderten Experimente.