Hochschule Braunschweig/Wolfenbüttel
Institut für Angewandte Informatik
Prof. Dr. F. Seutter

Algorithmen und Datenstrukturen, Übungsaufgaben

1. Ein Ausdruck der Form

$$pol(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

heißt für $a_n \neq 0$ Polynom vom Grad n. Der Wert des Polynoms an der Stelle $x \in \mathbb{R}$ kann nach den Algorithmen 1.1 *Polynomwert* und 1.2 *Horner-Schema* berechnet werden.

Algorithmus 1.1 (Polynomwert)
Eingabe: $A = (a_0, a_1, a_2, \ldots, a_n), x \in \mathbb{R}$
Ausgabe: $pol = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$

```
POLYNOMWERT(A, x)
1  pol ← A[0]
2  for i ← 1 to n
3      do pol ← pol + A[i] · x^i
4  return pol
```

Algorithmus 1.2 (Horner-Schema)
Eingabe: $A = (a_0, a_1, a_2, \ldots, a_n), x \in \mathbb{R}$
Ausgabe: $pol = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$

```
HORNER-SCHEMA(A, x)
1  pol ← A[n]
2  for i ← 1 to n
3      do pol ← pol · x + A[n − i]
4  return pol
```

   (a) Verifizieren Sie den Algorithmus *Polynomwert*. Spezifizieren Sie dazu seine Schleifeninvariante und beweisen Sie diese durch vollständige Induktion.

   (b) Führen Sie eine Komplexitätsanalyse für den Algorithmus *Polynomwert* bzgl. seiner Laufzeit durch. Betrachten Sie dazu insbesondere die Anzahl der Multiplikationen:

      i. Wie viele Multiplikationen werden im schlechtesten Fall in Abhängigkeit von der Eingabegröße ausgeführt? Beachten Sie dabei, dass die Berechnung der Potenz x^i durch $\underbrace{x \cdot x \cdot \ldots \cdot x}_{i\text{-mal}}$ erfolgt.

      ii. Wie viele Multiplikationen werden im besten Fall ausgeführt?

      iii. Drücken Sie die Komplexität des schlechtesten Falls auch in asymptotischer Notation Θ aus. Beweisen Sie Ihre Lösung.

   (c) Führen Sie eine Komplexitätsanalyse für den Algorithmus *Horner-Schema* bzgl. seiner Laufzeit durch. Betrachten Sie dazu insbesondere die Anzahl der Multiplikationen:

      i. Wie viele Multiplikationen werden im schlechtesten Fall in Abhängigkeit von der Eingabegröße ausgeführt?

      ii. Wie viele Multiplikationen werden im besten Fall ausgeführt?

      iii. Drücken Sie die Komplexität des schlechtesten Falls auch in asymptotischer Notation Θ aus. Beweisen Sie Ihre Lösung.

   (d) Diskutieren Sie kurz die Ergebnisse der Komplexitätsanalyse der beiden Algorithmen.

2. Türme von Hanoi

   Das Spiel besteht aus drei Stäben A, B und C, auf die n gelochte Scheiben gelegt werden, alle verschieden groß. Zu Beginn liegen alle Scheiben auf Stab A, der Größe nach geordnet, mit der größten Scheibe unten und der kleinsten oben. Ziel des Spiels ist es, den kompletten Scheiben-Stapel von A nach C zu versetzen.

   Bei jedem Zug darf die oberste Scheibe eines beliebigen Stabes auf einen der beiden anderen Stäbe gelegt werden, vorausgesetzt, dort liegt nicht schon eine kleinere Scheibe. Folglich sind zu jedem Zeitpunkt des Spieles die Scheiben auf jedem Feld der Größe nach geordnet.

   Algorithmus 2.1 (Türme von Hanoi)
   Eingabe: $n \in \mathbb{N}$ Anzahl der Scheiben, x, y, z Verweise auf die drei Stäbe
   Ausgabe: keine

   ```
   TuermeHanoi(n, x, y, z)
   1  if n = 1
   2     then ZieheScheibe(x, z)
   3     else TuermeHanoi(n − 1, x, z, y)
   4          ZieheScheibe(x, z)
   5          TuermeHanoi(n − 1, y, x, z)
   ```

   Der Algorithmus besteht im Wesentlichen aus einer Prozedur TuermeHanoi, die vier Parameter besitzt. Mit n ist die Anzahl der zu verschiebenden Scheiben bezeichnet, mit x der Stab von dem verschoben werden soll, mit y der Stab, der als Zwischenziel dient und mit z der Stab, auf den die Scheiben verschoben werden sollen. Zur Lösung des eigentlichen Problems wird TuermeHanoi mit der Scheibenanzahl n, $x = A, y = B$ und $z = C$ aufgerufen. Die Prozedur ZieheScheibe(x, z) erzeuge hier die Ausgabe „Ziehe Scheibe von x nach z.“

   (a) Der Algorithmus werde mit TuermeHanoi$(3, A, B, C)$ aufgerufen. Erstellen Sie ein Protokoll über den Ablauf des Algorithmus. Dazu sind die jeweils aktuellen Parameter der Aufrufe von TuermeHanoi und die von ZieheScheibe erzeugten Ausgaben in der Folge ihrer Ausführung anzugeben.

   (b) Bestimmen Sie die Zeitkomplexität des Algorithmus:

      i. Bestimmen Sie die Eingabegröße.
      ii. Bestimmen Sie die grundlegende Anweisung.
      iii. Bestimmen Sie die Rekursionsgleichung zur Berechnung der Anzahl der Ausführungsschritte der grundlegenden Anweisung.
      iv. Lösen Sie die Rekursionsgleichung.
      v. Spezifizieren Sie die Zeit-Komplexität in asymptotischer Notation (ohne Beweis).

3. Der Algorithmus *Binäres Suchen* ist in dem Skript als ein *iteratives* Verfahren implementiert.

   (a) Modifizieren Sie diesen Algorithmus zu einem *rekursiven* Verfahren.

   (b) Überprüfen Sie Ihr Verfahren für $A = (2, 4, 7, 11, 12, 14, 15, 19)$ und den zu suchenden Werten $x = 11$ und $x = 13$.

   (c) Bestimmen Sie die Komplexität Ihres Verfahrens im besten und im schlechtesten Fall in asymptotischer Notation.

4. Protokollieren Sie den Ablauf des Algorithmus *Quick-Sort* für die folgenden Zahlenfelder:

   (a) $A = (7, 4, 2, 3, 8, 5)$

   (b) $A = (2, 3, 4, 5, 7, 8)$

   Geben Sie dazu insbesondere die vorgenommenen Vertauschungen in ihrer Reihenfolge an. Kommentieren Sie die Laufzeiten für beide Aufgabenteile.

5. (a) Die abstrakten Datentypen
   i. Stapel,
   ii. Warteschlange

   sind basierend auf der statischen Datenstruktur eines Feldes der Länge n zu implementieren. Spezifizieren Sie in Pseudocode Prozeduren zum Einfügen, Lesen und Auslesen (Lesen und Löschen) von Elementen. Zum Abfangen von Über- und Unterläufen sind entsprechende Prozeduren hinzuzufügen.

   (b) Implementieren Sie den abstrakten Datentyp einer Prioritäts-Warteschlange auf Basis einer einfach verketteten Liste. Neue Elemente werden entsprechend der aufsteigenden Sortierung eingefügt – bei Gleichheit wird dahinter eingefügt – und ausgelesen wird das jeweils erste Element. Spezifizieren Sie in Pseudocode Prozeduren zum Einfügen, Lesen und Auslesen (Lesen und Löschen) von Elementen.

   (c) Bestimmen Sie die Zeitkomplexitäten der Prozeduren (ohne Beweis).

6. (a) Gegeben sei ein Feld A_1 der Länge 15. A_1 sei von $A_1[1]$ beginnend mit den Werten $(23, 18, 21, 15, 16, 11, 8, 9)$ belegt, und ist damit ein binärer Max-Heap mit $heapsize[A_1] = 8$.
   i. Stellen Sie A_1 als Baum dar.
   ii. Fügen Sie ein neues Element mit dem Wert 20 in A_1 ein, so dass A_1 ein binärer Max-Heap bleibt. Welche Vertauschungen sind vorzunehmen?
   iii. Das Element mit dem Wert 23 werde auf den Wert 10 geändert. Die Max-Heap-Eigenschaft ist für A_1 wieder herzustellen. Welche Vertauschungen sind vorzunehmen?
   iv. Geben Sie die Wertefolge der Elemente von A_1 in der Darstellung als Feld an.

   (b) Gegeben sei ein Feld A_2 der Länge 15. A_2 sei von $A_2[1]$ beginnend mit den Werten $(8, 9, 11, 15, 16, 18, 21, 23)$ belegt.
   i. Stellen Sie A_2 als Baum dar.
   ii. Überführen Sie A_2 in einen binären Max-Heap. Welche Vertauschungen sind vorzunehmen?
   iii. Geben Sie die Wertefolge der Elemente von A_2 in der Darstellung als Feld an.

   (c) Zeigen Sie durch vollständige Induktion: In einem n-elementigen Heap gibt es höchstens $\lceil n/2^{h+1} \rceil$ Knoten, von denen Unterbäume genau der Höhe h ausgehen, $0 \leq h \leq \lfloor \mathrm{ld}\, n \rfloor$.

7. Ein Verein habe 756 Mitglieder, und auch in der Zukunft werde der Verein nicht mehr als 1000 Mitglieder haben. Eine Mitglieds-Nummer gebe es nicht.

   Für diesen Verein ist eine Mitglieder-Datei anzulegen. Die Mitglieder-Datei ist als eine Hashtabelle mit offener Adressierung und einem initialen Belegungsfaktor von 1/2 zu implementieren. Dabei sind die ersten fünf Buchstaben des Nachnamens für das Hashing zu verwenden. Ist der Nachname kürzer, dann ist er durch Anhängen von *Blanks* entsprechend zu verlängern.

   Das zugrunde liegende Alphabet bestehe dabei aus 26 Buchstaben und dem *Blank*, eine Unterscheidung zwischen Groß- und Kleinbuchstaben werde nicht gemacht.

   (a) Spezifizieren Sie eine Hashfunktion.

   (b) Bestimmen Sie den aktuellen Belegungsfaktor.

   (c) Welche mittlere Länge hätten die Kollisionsketten, wenn zu Hashing mit geschlossener Adressierung übergegangen werden würde?