Algorithmen und Programmierung 3, WS 2023/24 — Klausur

Abgabe bis Donnerstag, 22. Februar 2024, 16:15 Uhr (120 Minuten)

---

1. Spezifikation und Implementierung eines abstrakten Datentyps, 10 Punkte

   Die folgende Klasse `SoSet` implementiert eine Menge von ganzen Zahlen als Feld (`array`).

```
import java.nio.BufferOverflowException;
public class SoSet {
  private int n, maxGröße;
  private int A[];
  SoSet(int max) {
    maxGröße = max;
    A = new int[maxGröße];
    n = 0;
  }
  public boolean contains(int x) {
    for (int i=0; i<n; i++)
      if (A[i]==x) return true;
    return false;
  }
  public void add(int x) {
    if (this.contains(x)) return;
    if (n == maxGröße) throw
      new BufferOverflowException();
    int i;
    for(i=n-1; i>=0 && A[i]>x; i--)
      A[i+1] = A[i];
    A[i+1] = x;
    n += 1;
  }
  ...
```

PYTHON-Version:

```
class SoSet:

  def __init__(self, maxGröße):
    self.maxGröße = maxGröße
    self.n = 0
    self.A = [0]*self.maxGröße

  def contains(self, x):
    return any(w == x
      for w in self.A[:self.n])

  def add(self, x):
    if self.contains(x): return
    if self.n == self.maxGröße:
      raise MemoryError()
    i = self.n - 1
    while i>=0 and self.A[i]>x:
      self.A[i+1] = self.A[i]
      i -= 1
    self.A[i+1] = x
    self.n += 1

  ...
```

   (a) Wie wird die Menge $\{1, 3, -5\}$ bei einem `SoSet` mit $maxGröße = 10$ dargestellt?

   (b) Definieren Sie einen passenden abstrakten Datentyp, der zusätzlich zur Initialisierung und zu `contains` und `add` auch die Operationen `discard` (Löschen eines Elementes) und `isempty` (Test auf leere Menge) unterstützt.

   Spezifizieren sie alle Operationen und den Konstruktor durch Vorbedingungen und Nachbedingungen.

   (c) Geben Sie die Abstraktionsfunktion und gegebenfalls die Darstellungsinvarianten und Nebenbedingungen dieser Implementierung an. Geben Sie an, was bei Verletzung der Nebenbedingungen passiert. (Die Korrektheit brauchen Sie nicht zu beweisen.)

   (d) Ergänzen Sie die Implementierung der Methode `isempty` in JAVA oder PYTHON. (Die Methode `discard` brauchen Sie nicht zu implementieren.)

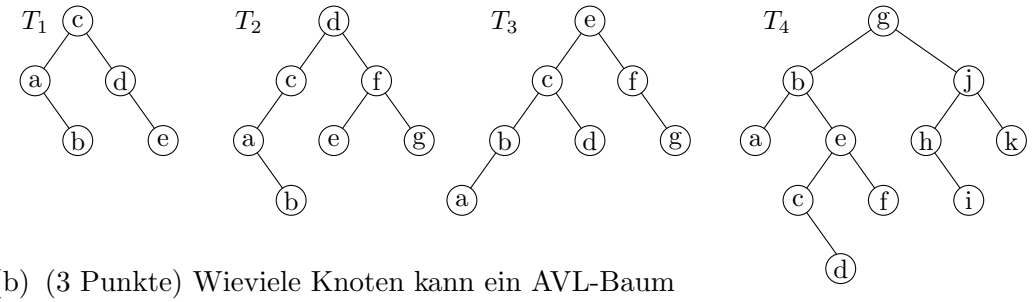
   Es kommt dabei auf die Korrektheit des Algorithmus an. Syntaxfehler werden in Maßen toleriert.

   (e) Bonusfrage, 3 Zusatzpunkte

   An welchen Stellen könnte man die gegebene Implementierung effizienter machen?

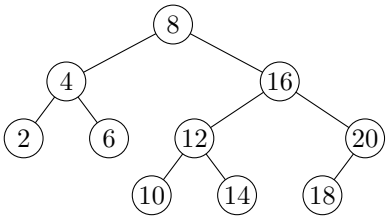
2. AVL-Bäume, 10 Punkte

   (a) (4 Punkte) Welche der folgenden binären Suchbäume sind AVL-Bäume? Zeigen Sie gegebenenfalls, an welcher Stelle die AVL-Bedingung verletzt ist.



   (b) (3 Punkte) Wieviele Knoten kann ein AVL-Baum der Höhe h *höchstens* enthalten?

   (c) (3 Punkte) Fügen Sie den Schüssel 13 in den folgenden AVL-Baum ein. Zeigen und erläutern Sie alle Zwischenschritte.



3. Polynomielle Reduktion, 10 Punkte

   Betrachten Sie zwei folgende Aufgaben:

   KWV (kürzester Weg in einem vollständigen Graphen)
   Eingabe: Ein vollständiger gerichteter Graph F mit positiven ganzzahligen Kantengewichten $c_{ij} > 0$, und zwei Knoten u und v. Gesucht ist die Länge des kürzesten Weges von u nach v.

   ZUSAMMENHANG: *Eingabe:* Ein ungerichteter Graph G.
   Frage: Ist G zusammenhängend?

   Zeigen Sie:

$$\text{ZUSAMMENHANG} <_P \text{KWV}$$

4. Kürzester Weg, 10 Punkte

   Berechnen Sie die Länge der kürzesten Wege vom Knoten 1 zu allen anderen Knoten im Graphen mit der folgenden Gewichtsmatrix $C = (c_{ij})_{i,j=1,2,3,4,5}$ mit dem Algorithmus von Dijkstra.

   Bestimmen Sie auch den kürzesten-Wege-Baum.

$$C = \begin{pmatrix} - & - & 10 & 2 & - \\ - & - & 6 & 7 & 2 \\ 9 & 1 & - & 5 & 2 \\ - & 9 & 5 & - & 8 \\ - & - & - & 3 & - \end{pmatrix}$$

   Geben Sie nach jedem Schritt den Wert der für den Algorithmus wesentlichen Variablen an. (Interne Darstellungen einer Datenstruktur, zum Beispiel einer Prioritätswarteschlange, brauchen Sie nicht anzugeben.)
   Das Betrachten aller ausgehenden Kanten eines Knotens zählt dabei als ein Schritt.