Dr. S. Lewandowski **Einführung in die Informatik I (mech/autip)** WS 2008/09

Übungsblatt 13

Ausgabe: 28.1. Abgabe: Mi., 4.2., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

Von jedem Aufgabenblatt werden maximal 20 Punkte auf den Schein angerechnet.

Aus gegebenen Anlass nochmals der Hinweis: Wählen Sie aussagekräftige Bezeichner (nicht nur a, b, c). Verwenden Sie Umlaute möglichst nicht im Kommentar und auf keinen Fall im Ada-relevanten Quelltext. Wenn der Compiler Warnungen ausgibt (Variable wird nicht benutzt, auf Variable wird zugegriffen, bevor ihr ein Wert zugewiesen wurde, ...), sollten Sie diese soweit möglich behandeln. Und zu guter Letzt: Bei der Angabe von Testfällen sollten Sie auch immer Problemfälle mit raussuchen (divide by zero, range check, ...).

Dieses Aufgabenblatt ist sehr umfangreich – suchen Sie sich selbstständig Aufgaben im Umfang von 20 Punkte zur Bearbeitung aus.

---

1. (2+3 Punkte, mittel) **Listen:** Schreiben Sie zwei **rekursive** Prozeduren zur Verarbeitung von einfach verketteten Listen (als Element-Typ können Sie `integer` verwenden):

   - `Put_revers(anker : Liste)` – gibt die Liste in umgekehrter Reihenfolge aus. Die Liste selbst soll dabei unverändert bleiben.
   - `Revers_List(anker : Liste)` – diese Prozedur soll die Richtung der Verzeigerung innerhalb der Liste umkehren. Nach dem Aufruf soll der Anker also auf das vormals letzte Element zeigen, dieses auf das vorletzte, usw. Von welcher Parameterart sollte `anker` sein? Was passiert, wenn Sie `Revers_List` zwei Mal aufrufen?

   Geben Sie jeweils den Aufwand (als Kommentarzeilen eingefügt) in O-Notation an.

2. (3+3 Punkte, mittel) **Binärbäume und Suchbäume:** Wir betrachten hier Bäume mit `integer`-Werten als Knoteninhalt.

   - Schreiben Sie eine Funktion, die in einem Binärbaum den Wert des Knotens mit größtem Inhalt bestimmt und diesen zurückgibt (ist der Baum leer, so soll der Wert 0 zurückgegeben werden).
   - Schreiben Sie eine effiziente Funktion, die in einem Suchbaum den Wert des Knotens mit größtem Inhalt bestimmt und diesen zurückgibt (ist der Suchbaum leer, so soll der Wert 0 zurückgegeben werden).

   Geben Sie jeweils den Aufwand (als Kommentarzeilen eingefügt) in O-Notation an.

3. (4 Punkte, leicht–mittel) **Datenstrukturen für Graphen:** Graphen liegen manchmal nicht in dem gewünschten Datenformat vor. Schreiben Sie eine Funktion, die aus einer Adjazenzmatrix die Adjazenzlisten-Darstellung des Graphen erzeugt. (Um Testfälle zu erzeugen, können Sie die Matrix zufällig mit 0 und 1 füllen – um Schlingen zu vermeiden, sollten dabei die Elemente in der Diagonalen 0 sein.)

   Geben Sie den Aufwand in O-Notation an.

4. (5+4+3 Punkte, mittel–schwer) **Graphdurchläufe:** In der Vorlesung haben wir die rekursive Fassung der Tiefensuche kennengelernt. Dort wurde auch skizziert, wie man dieses iterativ formulieren kann, indem man den Keller der noch zu bearbeitenden Knoten selbst verwaltet.

(a) Schreiben Sie eine iterative Version der Prozedur `Tiefensuche`.

(b) Ersetzen Sie den Keller in der iterativen Version durch eine Schlange (Sie erhalten so eine Breitensuche). Fügen Sie dabei eine zusätzliche Zahl als Inhalt zu den Knoten hinzu. Diese Zahl soll für den Startknoten (mit dem die Breitensuche gestartet wird) mit 0 initialisiert werden. Wenn die ausgehenden Kanten (u, v) eines Knotens u betrachtet werden, so soll der Wert des jeweils adjazenten Knotens v auf einen eins höheren Wert als der des Knotens u gesetzt werden.

(c) Bei der Breitensuche geben die Nummern, die den Knoten zugewiesen wurden, jeweils die Länge des kürzesten Wegs zu diesen Knoten an. Begründen Sie, warum dies so ist.

5. (3+1+5 Punkte) **Transitive Hülle:** Die Transitive Hülle G_{tH} eines Graphen G erweitert die Kantenmenge so, dass es zwischen zwei Knoten u und v genau dann eine Kante (u, v) gibt, wenn es im Graphen G einen Weg von u nach v gibt. Folgender Algorithmus ändert die Adjazenzmatrix `A` eines Graphen so ab, dass diese am Ende der Transitiven Hülle des Graphen entspricht.

```
for i in 1..n loop A(i,i):=1 end loop; -- der leere Weg verbindet i mit i
for k in 1..n loop
  for i in 1..n loop
    for j in 1..n loop
      if A(i,k)=1 and A(k,j)=1 then -- es existiert ein Weg über den Knoten k
        A(i,j):=1;
      end if;
    end loop;
  end loop;
end loop;
```

(a) (mittel, 3 Punkte) Begründen Sie, warum der Algorithmus die Transitive Hülle eines Graphen korrekt berechnet (Hinweis: nach dem ersten Durchlauf durch die `k`-Schleife sind alle Kanten `(i,j)` im durch die Adjazenzmatrix $\mathtt{A}_{\mathtt{ij}}$ dargestellten Graphen vorhanden (d.h. `A(i,j)=1`), die entweder bereits im Ausgangsgraphen vorhanden sind oder für die es einen Weg gibt, der nur über den Zwischenknoten `1` führt – setzen Sie den Beweis induktiv fort).

(b) (leicht, 1 Punkte) Geben Sie in O-Notation an, welchen Aufwand der Algorithmus in Abhängigkeit von der Knotenanzahl `n` hat.

(c) (schwer, 5 Punkte) In der Regel liegt der Graph als Adjazenzliste vor, sodass die Abfrage, ob `A(i,j)=1`, nicht in einem Schritt erfolgen kann. Will man direkt auf der Adjazenzliste arbeiten, so muss man zunächst die Knotenliste durchsuchen, um den Knoten `i` zu finden und in dessen Kantenliste überprüfen, ob es dort eine Kante zum Knoten `j` gibt.

Beschreiben Sie, wie die Knoten- und Kantenlisten angeordnet sein sollten, um unnötige Arbeit zu ersparen.

Schätzen Sie ab, wie groß der Aufwand zur Berechnung der Transitiven Hülle ist, wenn man direkt auf der Adjazenzliste arbeitet.

Wie groß wäre der Aufwand, wenn man zunächst die Adjazenzliste in eine Adjazenzmatrix umwandelt, dann mit der Matrix die Transitive Hülle berechnet und danach die Matrix wieder in eine Adjazenzliste zurückwandelt?

Geben Sie Ihre Abschätzungen jeweils in O-Notation an und begründen Sie Ihre Antworten.
