Algorithmische Mathematik

Wintersemester 2013
Prof. Dr. Marc Alexander Schweitzer und
Dr. Einar Smith
Patrick Diehl und Daniel Wissel

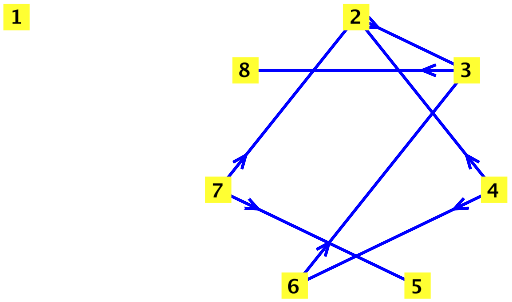
universitätbonn

Übungsblatt 4.

Abgabe am **18.11.2013**.

Aufgabe 1. (Topologische Sortierung)

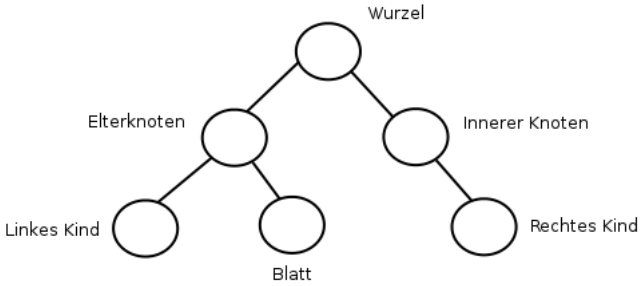
Geben Sie die topologische Sortierung für diesen Graphen an.



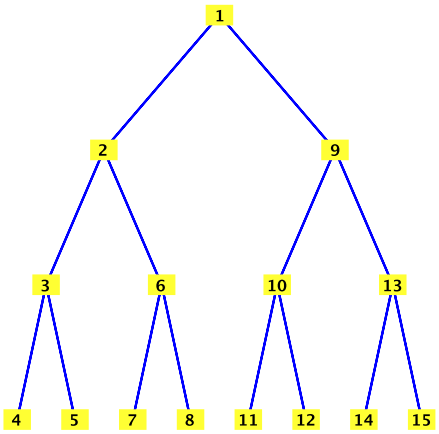
(2 Punkte)

Aufgabe 2. (Binärbaum)

In der Vorlesung ist in der Definition 2.8 der Baum als Spezialfall eines Graphen definiert. Ein weiterer Spezialfall eines Baums ist der Binärbaum. Die Abbildung zeigt die Begriffsdefinitionen eines Binärbaums. In einem Binärbaum darf ein Knoten maxi-



mal zwei Kinderknoten haben. Die untersten Knoten werden als Blätter bezeichnet. Alle Knoten links unterhalb der Wurzel werden als linker Teilbaum bezeichnet. Die Abbildung zeigt einen vollständigen Binärbaum der Höhe $h = 4$.



a. Zeigen Sie, dass ein vollständiger Binärbaum genau 2^{h-1} Blätter hat.

b. In welchem Fall artet der Binärbaum zu einer Liste aus?

(4 + 1 = 5 Punkte)

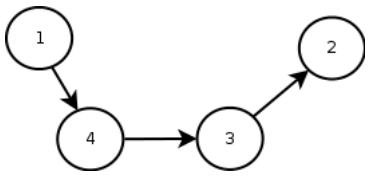
Aufgabe 3. (Visualisierung der Tiefensuche)

In der Tiefensuche wird das Prinzip des Backtrackings verwendet. Der Algorithmus in Pseudocode wendet die Tiefensuche auf einen Graphen an und färbt die Knoten des Backtrackingvorgangs mit schwarzer Farbe.

```
1  Tiefensuche(G)
       for each v of G {
3          farbe[v] = 'weiss';
           vorgaenger[v] = 0;
5      }
   for each u of G
7      if farbe[u] == 'weiss'
   besuche(u);
9
   besuche(u)
11    farbe[u] = 'grau';
      for each v of Adjazenzliste[u] {
13        if farbe[v] == 'weiss' {
             vorgaenger[v] = u;
15           besuche(v);
          }
17    }
      farbe[u] = 'schwarz';
```

a. Wenden Sie den Algorithmus ausgehend von Knoten 1 auf den Graphen an. Zeichnen Sie in jedem Schritt den Graphen und aktualisieren Sie die Farbe und den Vorgänger des Knoten. Schreiben Sie den aktuellen Vorgänger und die Farbe jeweils neben den Knoten.

b. Erklären Sie, was man bei der Tiefensuche unter Backtracking versteht.



c. Geben Sie den zusätzlichen Speicherverbrauch des Algorithmus an, wenn Sie nicht wissen, in welcher Form der Graph gespeichert ist.

d. Geben Sie die Komplexität des Algorithmus an, wenn der Graph als Adjazenzliste bzw. als Adjazenzmatrix gespeichert ist.

$(4 + 2 + 2 + 2 = 10$ Punkte)

Aufgabe 4. (Programmierung auf dem Papier (Prüfungsvorbereitung))

Geben Sie ein `struct` an, das einen Knoten im Binärbaum repräsentiert. Hierzu müssen Sie den Inhalt des Knoten (`int`) und die beiden Kinder speichern. Achten Sie auf eine korrekte C99-Syntax.

(3 Punkte)

Programmieraufgabe 1. (Addition mit Listen)

Gegeben sei folgendes Beispielprogramm:

```
#include <limits.h>
#include <stdio.h>

int main(void)
{
  unsigned int i = UINT_MAX;
  unsigned int j = UINT_MAX;

  printf("%u␣\n" , i + j);
}
```

Dieses Programm (Test.c) finden Sie in den Materialien zum dritten Übungsblatt auf der Homepage. Kompilieren Sie dieses Programm und schauen Sie sich die Ausgabe an. Was bemerken Sie?

In manchen Fällen wollen Sie auch ganze Zahlen, die über die maximalen Werte der Standarddatentypen gehen, verwenden. Hierfür eignen sich dynamische Datenstrukturen, wie einfach verkettete Listen. In diesen Listen können sehr große Ganzzahlen in folgender Form gespeichert werden:

$$127_{10} = 1 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 \Rightarrow \text{ Liste } \boxed{7} \rightarrow \boxed{2} \rightarrow \boxed{1}$$

Mit dieser Darstellung der Ganzzahl können nun beliebig große Zahlen addiert werden:

$$\begin{array}{l} \boxed{7} \rightarrow \boxed{2} \rightarrow \boxed{1} \quad + \\ \boxed{7} \rightarrow \boxed{0} \rightarrow \boxed{0} \end{array}$$

Nun kann man, wie gewohnt aus Grundschulzeiten, von hinten die Zahlen in der Liste addieren und den Übertrag zur nächsten Stelle weiter reichen.

$$\begin{array}{r} \text{Erste Stelle: } 7 + 7 = 14 \text{ Übertrag: } 1 \\ \text{Zweite Stelle: } 2 + 0 + 1 = 3 \text{ Übertrag: } 0 \\ \text{Dritte Stelle: } 1 + 0 + 0 = 1 \text{ Übertrag: } 0 \end{array}$$

In der resultierenden Liste steht nun das Ergebnis $134 = \boxed{4} \rightarrow \boxed{3} \rightarrow \boxed{1}$.

a. In den Materialien zum Übungsblatt finden Sie die Datei Main_Template.c. In dieser Datei finden Sie jede Menge Kommentare mit `//`**ToDo**. Bearbeiten Sie diese Arbeitsaufträge.

b. Addieren folgenden Zahlen mit ihrem neuen Datentyp:

   (a) $65535 + 65535$

   (b) $18446744073709551615 + 9223372036854775807$

Hinweise:

a. Die aktuelle Ziffer bekommen Sie mittels *(ziffer1 + ziffer2 + ?) % 10.*

b. Den Übertrag bekommen Sie mittels *(int)((ziffer1 + ziffer2 + ?) / 10).*

c. Für jedes der ersten 6 `//`**ToDo** bekommen Sie jeweils 3 Punkte. Auch wenn Sie nicht alle `//`**ToDo** bearbeiten können, bekommen Sie die entsprechenden Teilpunkte angerechnet.

(18 + 2 = 20 Punkte)

Abgabe am 25.11.2013 zwischen Vorlesung A und Vorlesung B

Programmieraufgabe 2. (Implementierung von Mengen)

In dieser Aufgabe sollen Sie eine Liste, die eine mathematische Menge M repräsentiert, implementieren. In der Menge M sollen nur Elemente $x \in \mathbb{N}$ gespeichert werden können. Folgende Operationen sollen auf der Menge M ausführbar sein:

a. Einfügen: `void insert(struct node** s, int v)`.

b. Löschen: `void delete(struct node** s, int v)`.

c. Mächtigkeit einer Menge: `int cardinality(struct node* s)`.

d. Element in der Menge: `int contains(struct node* s, int v)`.

e. Schnittmenge $M_1 \cap M_2 := \{x|(x \in M_1) \wedge (x \in M_2)\}$
`void intersection(struct node* s1, struct node* s2, struct node** result)`

f. Differenzmenge $M_1 \setminus M_2 := \{x|(x \in M_1) \wedge (x \notin M_2)\}$
`void difference(struct node* s1, struct node* s2, struct node** result)`

g. Schreiben Sie für alle Operationen sinnvolle Testfälle in Ihr Hauptprogramm. Anhand dieser Testfälle muss ersichtlich sein, dass Ihre Operationen auf der Liste funktionieren.

Hinweise: Ihr `struct` zur Speicherung des Knoten muss `struct node` benannt werden.

((6*3)+4=22 Punkte)

Abgabe am 25.11.2013 zwischen Vorlesung A und Vorlesung B