Technische Universität München
Fakultät für Informatik, I-14
Prof. Harald Räcke
Ruslan Zabrodin

SoSe 2021
Übungsblatt 2
23. April 2021

Algorithmen und Datenstrukturen

Aufgabe 1 Rekursion

a) Folgendes Produkt definiert eine Zahlenfolge:

$$C_n = \prod_{k=1}^{n} \frac{4k-2}{k+1}$$

(i) Stellen Sie C_n in einer Rekursionsformel dar, d.h. führen Sie C_n auf den Wert C_{n-1} zurück!

(ii) Schreiben Sie einen *rekursiven* Algorithmus in Pseudocode zur Berechnung von C_n

b) Gegeben sei folgende Rekursionsgleichung zur Annäherung von π:

$$f_n = \begin{cases} \frac{4}{2n+1} + f_{n-1} & n \text{ gerade} \\ \frac{-4}{2n+1} + f_{n-1} & n \text{ ungerade} \end{cases}$$

wobei $f_0 = 4$.

Schreiben Sie ein C/C++ Programm, das nach Eingabe des Index n die Annäherung f_n für π *rekursiv* berechnet und ausgibt.

```
float f(int n) {
    \\ ...
}
```

Aufgabe 2 Binomische Koeffizienten - Memoization

Im vorherigen Aufgabenblatt haben Sie bereits die folgende Rekursionsvorschrift für Binomialkoeffizienten kennengelernt:

$$\binom{n}{0} = \binom{n}{n} = 1 \qquad \text{für } n \geq 0$$

$$\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1} \qquad \text{für } n > 0 \text{ und } 0 \leq k < n$$

Schreiben Sie ein rekursives C/C++ Programm, das den Binomialkoeffizienten für gegebene n und k berechnet! Benutzen Sie die *Memoizationtechnik*, wie sie in der Vorlesung für die Fibonacci Zahlen gezeigt wurde.

Aufgabe 3 Linked List - Erweiterte Operationen

In der Vorlesung wurde eine vollständige Implementierung von Linked List in C/C++ präsentiert. Basierend auf diesem Code, sollen Sie die Klasse Linked List um bestimmte Operationen erweitern. Orientieren Sie sich dabei an die in der Vorlesung bereits präsentierten Operationen. Implementieren Sie folgende Funktionen mit einer rekursiven Hilfsfunktion:

a) Implementieren Sie die Funktion

int max();

und die dazugehörige rekursive Hilfsfunktion

int max(ListNode∗ &ptr);

Die Funktion soll das maximale Datenelement, das in der Linked List gespeichert ist, ausgeben.

b) Implementieren Sie die Funktion

int sum(**int** i, **int** j);

und die dazugehörige rekursive Hilfsfunktion

int sum(**int** i, **int** j, ListNode∗ &ptr);

Die Funktion soll die Summe der Datenelemente von dem i-ten Listenelement bis zum j-ten Listenelement ausgeben. Dabei soll $i \leq j$ angenommen werden.

c) Implementieren Sie die Funktion

void sorted_insert(ListNode∗ node);

und die dazugehörige rekursive Hilfsfunktion

void sorted_insert(ListNode∗ node, ListNode∗ &ptr);

Nehmen Sie an, die Liste ist bereits aufsteigend sortiert. Die Funktion soll ein neues ListNode Element (node) an den richtigen Platz in die sortierte Liste einfügen, sodass die Liste nach dem Einfügen weiterhin aufsteigend sortiert bleibt.

d) Implementieren Sie die Funktion

void insertion_sort();

und die dazugehörige rekursive Hilfsfunktion

void insertion_sort(ListNode∗ &ptr);

Später in der Vorlesung werden Sie ausgefeiltere Sortieralgorithmen kennenlernen, aber ein einfacher Algorithmus ist bereits jetzt möglich. Insertionsort sortiert eine Liste aufsteigend nach folgendem Schema:

```
function INSERTION_SORT(ptr)
    if die Liste mit ptr als Kopf ist nicht leer then
        Sortiere die Unterliste mit ptr.next als Kopf rekursiv
        Füge das Element auf das ptr zeigt in die sortierte Unterliste ein
    end if
end function
```

Implementieren Sie diesen Algorithmus in-place, d.h. ohne temporären Speicher / Nodes. Benutzen Sie dabei die Hilfsfunktion aus Teilaufgabe c) sorted_insert.

Aufgabe 4 Endrekursion

In dieser Aufgabe sei % der Modulo-Operator und / der Operator für ganzzahlige Division. D. h. $12345/10 = 1234$ und $12345\%10 = 5$.

Gegeben ist die folgende Funktion $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

$$g(x,y) = \begin{cases} x^y & x < 10 \\ (x\%10)^y + g(x/10, y+1) & x \geq 10 \end{cases}$$

- Implementieren Sie die Funktion *g endrekursiv* als eine C/C++ Methode

  int g_tailrec(**int** x, **int** y) ;

  Sie benötigen dazu eine Hilfsfunktion. Überlegen Sie, welche und wie viele Parameter diese Hilfsfunktion benötigt, und implementieren Sie die Hilfsfunktion.

- Implementieren Sie die Funktion *g iterativ* als eine C/C++ Methode

  int g_iter(**int** x, **int** y);

  indem Sie die Endrekursion entfernen, analog der Vorgehensweise wie in der Vorlesung für die Funktion *ggT* gezeigt.