Algorithmen und Datenstrukturen
Algorithmen-Schemas

Aufgaben

1. Aufgabe

Rekursiver Divide and Conquer Algorithmus

Die Umkehrung der Buchstabenfolge eines Wortes (z.B. GRAS - > SARG) kann mit einem rekursiven Algorithmus beschrieben werden. Dies geschieht wie folgt: Man nehme den ersten Buchstaben des Wortes, entferne diesen und kehre dann den Rest des Wortes um. Anschliessend wird der entfernte Buchstabe angehängt. Die Basis für einen geeigneten Algorithmus ist deshalb:

1. Entferne den ersten Buchstaben
2. Kehre den Rest des Wortes um
3. Hänge den entfernten Buchstaben an

Der Algorithmus ist offensichtlich rekursiv, denn er beschreibt, wie eine Buchstabenfolge umzukehren ist, indem er beschreibt, wie eine etwas kürzere Buchstabenfolge umzukehren ist. Was noch nicht formuliert wurde, ist die Abbruchbedingung. Diese kann wie folgt lauten:

falls die Buchstabenfolge nur einen Buchstaben enthält dann .....

In einer Pseudosprache formuliert, lautet der Algorithmus:

```
Methode Umkehrung(Buchstabenfolge)
        Falls Buchstabenfolge nur ein Buchstaben enthält
                dann speichere den Buchstaben
                sonst entferne den ersten Buchstaben
                        Umkehrung(Rest der Buchstabenfolge)
                        hänge entfernten Buchstaben an
```

Implementieren Sie diesen Algorithmus in C#.
Hinweis: Das Problem könnte natürlich auch iterativ gelöst werden. Es soll jedoch explizit die Entwicklung eines Divide and Conquer Algorithmus geübt werden.

2. Aufgabe

Fibonacci iterativ

Implementieren Sie die Berechnung der Fibonacci-Zahl iterativ und vergleichen Sie anschliessend die Laufzeiten zwischen der rekursiven und der iterativen Lösung z.B. für Fibonacci(40). Sie können hierfür das Rahmenprogramm verwenden, welches Ihnen zur Verfügung gestellt wurde.

3. Aufgabe

Türme von Hanoi

Implementieren Sie einen Algorithmus für die «Türme von Hanoi». Halten Sie sich dabei an das Schema des Algorithmus, welches in den Slides zum Unterricht ersichtlich ist.

4. Aufgabe **Fibonacci Komplexität**

Bestimmen Sie die Komplexitätsklassen, in dem Sie Ihre Algorithmen mehrfach für verschiedene Problemgrössen durchführen, die Ausführungszeit messen und die entsprechende Kurve aufzeichnen.
Führen Sie dies für folgende Algorithmen durch:

- Fibonacci Iterativ
- Fibonacci Rekursiv
- Türme von Hanoi

5. Aufgabe **Rekursive Komplexität I**

Berechnen Sie die Komplexität der Prozedur procRec1. Wie oft wird do_something() aufgerufen? Wählen Sie für n eine Zweierpotent: $n=2^k$
Überprüfen Sie Ihre Lösung, indem Sie die Prozedur in C# implementieren und einen Zähler einbauen.

```
void procRec1(int n) { O(log n)
   if(n<=1) { O(1)
      return; O(1)
   }

   do_something(n); O(1)
   procRec(n/2); O(log n)
}
```

6. Aufgabe **Rekursive Komplexität II**

Berechnen Sie die Komplexität der Prozedur procRec2. Wie oft wird do_something() aufgerufen? Wählen Sie für n eine Zweierpotent: $n=2^k$
Überprüfen Sie Ihre Lösung, indem Sie die Prozedur in C# implementieren und einen Zähler einbauen.

```
void procRec2(int n, int res) {
   res = do_something(res, n);
   if(n<=1) {
      return res;
   }
   res = procRec2(n/2, res);
   res = procRec2(n/2, res);

   return res;
}
```