ALP II

SS 2012

Prof. Dr. Margarita Esponda

3. Übungsblatt

Ziel: Auseinandersetzung mit Rekursion vs. Iteration und erste Analyse der Klassenkomplexität von Algorithmen.

1. Aufgabe (12 Punkte)

Wenn wir eine Menge ***M*** mit ***n*** verschiedenen Objekten haben, kann die Anzahl der verschiedenen ***k***-elementigen Teilmengen aus ***M*** mit Hilfe des bekannten Binomialkoeffizienten wie folgt berechnet werden.

Binomialkoeffizient (n, k) = $\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!\,(n-k)!}$ mit $0 \le k \le n$

a) Schreiben Sie eine Python-Funktion mit folgender Signatur

def binom_naiv (n, k),

die mit Hilfe der vorherigen Definition und ohne Rekursion (auch nicht innerhalb der Fakultätsfunktion) für beliebige natürliche Zahlen ***n*** und ***k*** den Binomialkoeffizienten berechnet.

Eine rekursive Definition der gleichen Funktion sieht wie folgt aus:

$$\left(\begin{array}{c} n \\ k \end{array} \right) = 0 \text{ , wenn } k > n$$

$$\left(\begin{array}{c} n \\ 0 \end{array} \right) = \left(\begin{array}{c} n \\ n \end{array} \right) = 1$$

$$\left(\begin{array}{c} n \\ 1 \end{array} \right) = \left(\begin{array}{c} n \\ n-1 \end{array} \right) = n$$

$$\left(\begin{array}{c} n+1 \\ k+1 \end{array} \right) = \left(\begin{array}{c} n \\ k \end{array} \right) + \left(\begin{array}{c} n \\ k+1 \end{array} \right) \text{ für alle } n,k \in \mathrm{N} \text{ mit } 1 < k \le n-1$$

b) Implementieren Sie diese rekursive Definition der Funktion.

Aus der ersten Definition von **a)** kann folgende Gleichung abgeleitet werden:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \begin{cases} 1 & falls \quad k = 0 \\ \frac{n}{1} \cdot \frac{(n-1)}{2} \cdot \frac{(n-2)}{3} \cdots \frac{(n-k+2)}{(k-1)} \cdot \frac{(n-k+1)}{k} & falls \quad k > 0 \end{cases}$$

c) Implementieren Sie eine möglichst effiziente iterative Python-Funktion, die diese Definition des Binomialkoeffizienten verwendet.

d) Schreiben Sie eine Test-Funktion, die die realen Ausführungszeiten aller drei Funktionen mit Hilfe der **clock()**-Funktion des **time-**Moduls vergleicht. Erzeugen Sie mit Ihrer Testfunktion eine Tabelle, in der Sie die Ausführungszeit der drei Funktionen mit verschiedenen Eingabewerten ausgeben.

2. Aufgabe (5 Punkte)

Programmieren Sie eine eigene endrekursive Funktion in Python, die bei Eingabe von zwei positiven Zahlen ***a*** und ***b*** die Potenz a^b berechnet.

Programmieren Sie eine eigene iterative Version ihrer Potenz-Funktion unter Verwendung einer **for**-Schleife.

Analysieren Sie mit Hilfe der O-Notation die Ausführungszeit und den Speicherverbrauch beider Funktionen. Begründen Sie Ihre Antwort.

3. Aufgabe (5 Punkte)

Die Multiplikation von zwei natürlichen Zahlen kann als Reihensumme dargestellt werden und mit einer naiven Implementierung berechnet werden, die rekursiv oder iterativ programmiert werden kann.

Es gibt aber bessere Methoden, um zwei Zahlen zu multiplizieren, die auch bei Hardware-Implementierungen verwendet werden. Eine davon ist die berühmte russische Multiplikation. Der Name kommt von der Tatsache, dass diese Methode unter russischen Bauern üblich war.

Bauern-Multiplikation:

Wenn beispielsweise die Zahlen **a** und **b** multipliziert werden müssen, wird:

1) die Zahl **a** in eine Spalte so lange halbiert bis diese den Wert **1** oder **0** erreicht hat. Gleichzeitig wird die Zahl **b** in einer zweiten Spalte so lange verdoppelt.
2) Die Zeilen, bei denen sich in der ersten Spalte gerade Zahlen befinden, werden gestrichen.

3) Das Ergebnis der Multiplikation ergibt sich aus der Summe der übrig gebliebenen Zahlen der zweiten Spalte.

Beispiel:

45 x 33

45	33
~~22~~	~~66~~
11	132
5	264
~~2~~	~~528~~
1	1056
---	---
	1485

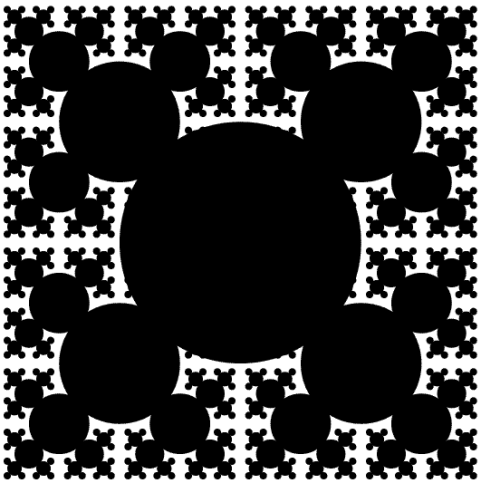
Schreiben Sie eine Python-Funktion, die den Bauern-Algorithmus verwendet. Sie dürfen selbstverständlich keine Multiplikation, Division oder Modulo-Operation dabei benutzen, in Ihrer Funktion sind nur Bit-Operationen, Vergleiche und Summen erlaubt.

Analysieren Sie die Komplexität Ihres Algorithmus mit Hilfe der O-Notation und begründen Sie Ihre Antwort.

4. Aufgabe (7 Punkte)

In der **ps_functions.py.zip**-Datei befindet sich eine einfache Bibliothek von Python-Funktionen. Mit Hilfe dieser Funktionen können einfache graphische Objekte in der PostScript-Sprache erzeugt und in eine Datei geschrieben werden. Eine PostScript-Datei besteht aus einer Reihe von Befehlen aus der PostScript-Sprache, die vom Drucker oder PostScript-Viewer interpretiert werden können.

Schreiben Sie eine rekursive Funktion, die folgendes "Mickey mouse"-Fraktal malt.



Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihre Programme.

4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.

5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.

6) Schreiben Sie getrennte Test-Funktionen für alle 5 Aufgaben für Ihren Tutor.