Algorithmen und Datenstrukturen (WS 13/14)

Prof. Dr. Michael Köhler-Bußmeier

Hausaufgaben 3 : Laufzeitanalyse

Übungsaufgabe 3.1: Bei Laufzeitabschätzung von Algorithmen geht man davon aus, dass alle "eingebauten" Konstrukte (wie Addition, Zuweisung usw.) eine konstante Zeit benötigen. Die Konstante selbst kennen wir nicht, sie ist typischerweise auch stark von der Hardware abhängig. Die Laufzeit von Prozeduren hängt dagegen i.a. von den Argumenten ab.

1. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 1 Quersumme von $A[1..n]$

```
1: x = 0
2: for i = 1 to n do
3:     x = x + A[i]
4: end for
5: return x
```

2. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 2 Alg. 1

```
1: for i = 1 to n do
2:     A[i] = i
3: end for
4: for i = 1 to n do
5:     C[i] = 0
6:     for j = n downto 1 do
7:         if A[j] > C[i] then
8:             C[i] = A[j]
9:         end if
10:    end for
11: end for
12: return C
```

3. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 3 Matrixmultiplikation

```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         C[i][j] = 0
4:         for k = 1 to n do
5:             C[i][j] = A[i][k] * B[k][j]
6:         end for
7:     end for
8: end for
9: return C
```

4. Bestimmen Sie Anzahl der Operationen, die der folgenden Algorithmus ausführt:

Algorithm 4 Alg. Beispiel

```
1: for i = 1 to n do
2:     for j = 1 downto i do
3:         x = x + A[i][j]
4:     end for
5: end for
6: return x
```

Übungsaufgabe 3.2:

1. Implementieren Sie folgendes Verfahren zum Potenzieren von x.

Algorithm 5 $exp(x, k)$; berechnet x^k, $k \geq 0$

```
1: r = 1
2: for i = 1 to k do
3:     r = r * x
4: end for
5: return r
```

2. Nutzen Sie folgende Gleichungen aus, um Potenzen schneller zu berechnen:

$$\begin{aligned} x^0 &= 1 \\ x^{2n} &= (x^n)^2 \\ x^{2n+1} &= x \cdot (x^n)^2 \end{aligned}$$

3. Vergleichen Sie die Laufzeit der Algotihmen in Abhängigkeit von k! Machen Sie geeignete Experiemente und stellen Sie die beiden Laufzeiten graphisch dar.

   Für welche Werte von x und k kommen die Implementation an ihre Grenzen?

4. Beide Implementation können auch dann eingesetzt werden, wenn x keine Zahl, sondern eine Matrix ist.

   Erweitern Sie Ihre Matrix-Klasse um eine Potenzierungsmethode. Verwenden Sie die beiden Implementationsvarianten. Vergelichen Sie die Laufzeiten, indem Sie wieder quadratische (geeignet große) Zufallsmatrizen erzeugen und diese Potenzieren. Variieren Sie die Potenz! Betrachten Sie insbesondere, was passiert, wenn k in die Größneordnung der Matrixdimension n kommt.

   Wiederholen Sie das Experiment t-mal und bilden Sie Mittelwerte usw.

   Wo liegen jetzt die Grenzen?

Übungsaufgabe 3.3: Die O-Notation.

1. Zeigen Sie: $15n^2 \in O(n^3)$.
2. Zeigen Sie: $\frac{1}{2}n^3 \notin O(n^2)$.
3. Betrachte $g(n) = 2n^2 + 3$.

   Geben Sie eine Funktion $f_1 : \mathbb{N} \to \mathbb{N}$ an, für die $f_1 \in O(g)$ und $f_1(n) < g(n)$ für alle n ab einem n_0 gilt.

   Geben Sie eine Funktion $f_2 : \mathbb{N} \to \mathbb{N}$ an, für die $f_2 \in O(g)$ und $f_2(n) > g(n)$ für alle n ab einem n_0 gilt.

   (Beide Funktionen f_1 und f_2 liegen also in $O(g)$, aber $f_1(n)$ ist stets kleiner und $f_1(n)$ ist stets größer als $g(n)$.)
4. Wir betrachten Polynome mit natürlichzahligen Koeffizienten, d.h. Funktionen der Form $f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ mit $a_i \in \mathbb{N}$ und wenn $a_k \neq 0$. Das Polynom hat dann den Grad k.

   Zeigen Sie: Für zwei Polynome f und g mit gleichem Grad gilt $f \in \Theta(g)$.
5. Zeigen Sie: Sei $f(n) := \sum_{i=0}^{n} 2^i$. Es gilt $f \in O(2^n)$