ALP 3 - Übung 1 - Tutor: Till Zoppke

Florian Deinert Holger Kiesewalter

10. November 2003

Aufgabe 1

$(b)\log_2 n \quad (f)\sqrt{n} \quad (d)n \quad (g)n(\log_2 n)^2 \quad (h)n^2 \quad (a)n^3 \quad (c)1{,}8^n \quad (e)3^n$

	n=20	n=1000
$\log_2 n$	20^{1000}	10^{3000}
$\sqrt{n}$	$2 \cdot 10^7$	10^9
n	20000	10^6
$n(\log_2 n)^2$	2839	300009
n^2	632	31622
n^3	200	10000
$1,8^n$	31	1011
3^n	26	1006

Die Ergebnisse mussten gegebenenfalls gerundet werden, da es keine halben Probleme gibt. Genauer mussten sie abgerundet werden, da nur diese Problemzahl dann auch wirklich vollständig bearbeitet werden kann. Fast alle Lösungen können mit sehr einfachen Umformungen der Gleichung $f(x) = 1000 \cdot f(n)$ gefunden werden. Folgendes Beispiel berechnet für (b) $\log_2 n$ und $n = 1000$ das gesuchte x.

$$\log_2 x = 1000 \cdot \log_2 1000 \tag{1}$$
$$2^{\log_2 x} = 2^{1000 \cdot \log_2 1000} \tag{2}$$
$$x = 2^{\log_2 1000^{1000}} \tag{3}$$
$$x = 1000^{1000} \tag{4}$$
$$x = 10^{3000} \tag{5}$$

Für (g) ist das Umformen dieser Gleichung nicht so einfach. Es bietet sich an, ein kleines Programm dafür zu schreiben, hier ein Beispiel in Haskell dafür: Die verwendete Formel ist dann: $x = max\{x | f(x) \leq 1000 \cdot f(n)\}$.

```
formel :: Float -> Float
formel 1 = x
formel x
        | (x*(log x)*(log x) <= 1000*20*(log 20)*(log 20)) = x
        | otherwise = formel (x-1)
```

Als Parameter muss der Funktion eine ganze Zahl übergeben werden, die größer als das zu erwartende x ist. Die Funktion findet dann das maximale x für das gilt: $f(x) \leq 1000 \cdot f(n)$.

Aufgabe 2

Für Software im Privatbereich trifft es wohl zu, dass die Computer heute schnell genug sind, um dem User lästiges Warten größtenteils zu ersparen. Solange es jedoch auch noch Algorithmen mit exponentieller Laufzeit gibt, nutzt es nicht viel die Prozessoren zu verbessern, da diese immer nur um einen gewissen Faktor schneller werden (z.B. x2 alle 18 Monate), was jedoch bei exponentiellen Problemen keinen entsprechenden Geschwindigkeitsgewinn bringt. Je langsamer ein Algorithmus ist, desto weniger wirken sich Prozessor- und Speicherverbesserungen aus, wie man in Aufgabe 1b sieht.
Außerdem wird die Hardware-Entwicklung irgendwann an ihre physikalischen Grenzen stoßen.

Aufgabe 8

(a) $3n^2 - 4n + 32 + 27\,n \cdot \lceil \log_2 n \rceil / 2 \quad = \Theta(n^2)$
Der quadratische Term wächst für $n \to \infty$ deutlich stärker als der lineare bzw der logarithmische, daher können die beiden letzteren vernachlässigt werden.

(b) $\max\{n\lceil \log_2 n \rceil, (\lceil \log_2 n \rceil)^4\} \qquad = \Theta(n \log_2 n)$
$(\log_2 n)^x$ mit $x \in \mathbf{N}$ wächst für $n \to \infty$ immer langsamer als ein linearer Term.

(c) $2^{2n+\lceil \log_2 n \rceil} = 2^{2n} \cdot 2^{\lceil \log_2 n \rceil} \leq 4^n \cdot n = \Theta(n \cdot 4^n)$
Potenzgesetze!

Aufgabe 9

```
public class Sortieren
{
  public static void main(String[] args)
  {
    int [] liste= {97,34,44,12,88,7,-3};
    int element=0;

    while (element<liste.length)
    {
    int min = element;
    for (int i=element+1; i<liste.length; i++) if (liste[i]<liste[min]) min=i;

    //vertauschen:
    int hilfsvar=liste[element];
    liste[element]=liste[min];
    liste[min]=hilfsvar;
    element++;
    }

    //Ausgabe:
    for (int x=0;x<liste.length;x++) System.out.println(liste[x]);

  }
}
```

Die Variable 'element' ist der Index, man beginnt bei einem Array immer mit 0. 'min' ist der Index des (derzeit) kleinsten Elementes. In der for-Schleife wird nun für alle in der Liste folgenden Elemente verglichen, ob diese kleiner sind als das derzeitige Minimum. Ist dies der Fall, so bekommt die Variable 'min' den Index-Wert des kleineren Elementes. Anschließend wird das Minimum mit dem ersten (bzw. element.) Wert der Liste vertauscht. Das ganze wird solange wiederholt bis die ganze Liste durchlaufen wurde.