Department „Institut für Informatik“
Ludwig-Maximilians-Universität München

Prof. Dr. Hans-Peter Kriegel
Thomas Bernecker, Tobias Emrich

SS 2008
14.04.2008
Übungsblatt 1

Übungen zur Vorlesung

Effiziente Algorithmen

Aufgabe 1.1: *O-Notation*

Geben Sie an, für welche $x \in \Re^+$ die folgende Gleichung gilt:

$\sum_{i=0}^{n} x^i = O(n)$. Beweisen Sie Ihr Ergebnis.

Aufgabe 1.2: *O-Notation*

Zeigen oder widerlegen Sie:

a) Wenn P(n) ein Polynom in n ist, dann gilt: $O(\log P(n)) = O(\log n)$

b) Es gilt für jedes positive k: $\sum_{i=0}^{n} i^k = O(n^{k+1})$

c) $n^{1,5} + n\log_2 n = O(n^{1,5})$

d) Wenn *f(n)=O(s(n))* und *g(n)=O(r(n))* gilt, dann gilt auch *f(n)-g(n)=O(s(n)-r(n))*

e) $2^{(2^n)} = O(n^{(2^n)})$

Aufgabe 1.3: *Maximum-Subarray*

Das Maximum-Subarray Problem besteht darin, in einem Array aus ganzen Zahlen der Länge *n* die **nichtleere** Teilfolge von Zahlen zu finden, deren Summe maximal ist.

Beispiel: Für die Eingabefolge X[0..9]

12, -32, 38, 22, -27, 32, 17, -13 , -10, 10

ist die maximale Summe einer Teilfolge 82 für die Teilfolge von 2 bis 6.

Ein naiver Algorithmus zur Lösung dieses Problems mit der Eingabe X[N] ist der folgende:

```
maxTfsum := 0;
for (u = 0; u < N ; u++)
{
  for(o = 0; o < N; o++)
   {
    // Bestimme die Summe der Elemente der Teilfolge von X[u..o]
    Sum = 0;
    for(i = u; i <=o;i++)
    {
     Sum = Sum + X[i] ;
    }
    //Vergleiche die gefundene Summe mit der bis jetzt maximalen
    maxTfsum = max (Sum, maxTfsum);
  }
}
```

a) Geben Sie die Zeitkomplexität des obigen Algorithmus an und begründen Sie Ihre Angabe kurz.

b) Geben Sie die Idee für einen Algorithmus an, der obiges Problem in linearer Komplexität löst und demonstrieren Sie diese am obigen Zahlenbeispiel.

c) Implementieren Sie Ihre Lösungsidee aus Aufgabe b) in Java.

Das Programm soll dabei die Summe, das erste und das letzte Element der maximalen Teilfolge als Ergebnis ausgeben. (Die Eingabe des Inputarrays als Konstante ist ausreichend.)