Algorithmen und Datenstrukturen I

WS 2010/11, **1. Aufgabenblatt**, Abgabe 3.11.2010

Aufgabe 1

10 Punkte

Gegeben sind die folgenden 8 Funktionen:

a) $n \mapsto n\sqrt{n}$ b) $n \mapsto 7n^2 + n\sin(\frac{\pi}{n})$ c) $n \mapsto 2^{\frac{n}{4}}$

d) $n \mapsto (2n+16)(n^2 - \frac{1}{2}n)$ e) $n \mapsto \frac{n}{n+100}$ f) $n \mapsto \frac{n^3}{\log(n^2+1)}$

g) $n \mapsto \sqrt[3]{n} + 2^{\log(\frac{3}{n+1})}$ h) $n \mapsto \log(n^{2n}) + 1024n$

Bringen Sie die Funktionen in eine Reihenfolge $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8$, so dass $f_1 \in O(f_2)$, $f_2 \in O(f_3)$, $f_3 \in O(f_4)$ usw. Berechnen Sie hierzu die (kleinste) Komplexitätsklasse $(O(f_i))$ für jede Funktion f_i.

Aufgabe 2

6 Punkte

Gegeben seien die folgenden Rekursionsgleichungen jeweils für eine Funktion T.

a) $T(n) = 27T(n/3) + 16n^2 + n$

b) $T(n) = 32T(n/2) + n^5 + \log(n)$

c) $T(n) = nT(n-1)$

Bestimmen Sie jeweils eine exakte Schranke für das Verhalten von T. Finden Sie also eine Funktion f, so dass $T \in \Theta(f)$. In den Fällen, wo die polynomiale Version des Master-Theorems anwendbar ist, benutzen Sie dieses zum Finden von f und geben a, b und k an.

Aufgabe 3

4 Punkte

Gegeben sind folgende Funktionsaufrufe:
a)

```
int function ( int n ) {

   int sum = 0;

   for(int i=1; i<=n; i=2*i) {
      for(int j=n; j>0; j--) {
         for(int k=0; k<2*n; k++) {
            sum = sum + i*(j+k);
         }
      }
   }

   return sum;

}
```

b)

```
int N = 0;

int function ( int n, int N) {

   if( n <= 3 ) return N;

   N++;

   function ( n/3, N);

}
```

Geben Sie für jeden Funktionsaufruf die obere Schranke an die Zeitkomplexität und Speicherkomplexität dieser Funktionen in Abhängigkeit von n mittels O-Notation an. Begründen Sie Ihre Antwort.

Aufgabe 4

4 Punkte

Gegeben sind folgende Funktionsaufrufe:
a)

```
int function ( int[] N, int[] M) {
   D = int[][];
   for (int i=1; i<=N.length; i++) {
      for (int j=1; j<=M.length; j++) {
         if (N[i] == M[j]) { a = D[i-1][j-i]+s }
         else {              a = D[i-1][j-i]+r }
         b = D[i-1][j]+g;
         c = D[i][j-1]+g;
         D[i][j]=max(a,max(b,c));
      }
   }
   return D[N.length][M.length];
}
```

b)

```
int function ( int[] N, int[] M) {
   for (int i=1; i<=N.length; i++) {
      A= int[];
      B= int[];
      for (int j=1; j<=M.length; j++) {
         if (N[i] == M[j]) { a = A[i-1]+s }
         else              { a = A[i-1]+r }
         b = B[i-1]+g;
         c = A[i]+g;
         B[i]=max(a,max(b,c));
      }
      A=B;
      B={};
   }
   return A[N.length];
}
```

Geben Sie für jeden Funktionsaufruf die obere Schranke an die Zeitkomplexität und Speicherkomplexität dieser Funktionen in Abhängigkeit von n (Länge der Liste N) und m (Länge der Liste M) mittels O-Notation an. Begründen Sie Ihre Antwort. Beachten Sie, `s`,`r` und `g` sind Konstanten.

Aufgabe 5

10 Punkte

In der Vorlesung haben Sie Verfahren zum effizienten Suchen in sortierten Listen kennen gelernt. Die Binärsuche lässt sich u.a. wie folgt rekursiv implementieren:

```
int binSearch(int[] f, int x, int l, int r) {

   p = (l+r)/2;

   if(f[p] == x) return(p);
   else if(f[p] < x) c = 1;
   else c = -1;
   if(l == r) return -1;
   if(c < 0) {
      if(p>l) return(binSearch(f, x, l, p-1));
      else return -1;
   } else {
      if(p<r) return(binSearch(f, x, p+1, r));
      else return -1;
   }
}
```

Geben Sie für `f[] = [1, 4, 8, 10, 16, 22, 27, 34, 38, 46, 50, 62, 74, 82, 86, 97];` an, wie die Methode `binSearch` auf f arbeitet, wenn nach folgenden Elementen gesucht wird:

(a) x=27

(b) x=89

Geben Sie für beide Teilaufgaben, die initialen Funktionsaufrufe von `binSearch` an, d.h. überlegen Sie sich wie l und r anfangs zu belegen sind und protokollieren Sie für jeden Rekursionsaufruf die Werte der Variablen l, r und p, sowie den Rückgabewert von `binSearch`.