Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Beispielblatt 1

Abzugeben bis Dienstag, 01. April 2014 um 23:59.

Geben Sie bis **spätestens Dienstag, 01.04.2014 um 23:59** Ihre ausgearbeiteten Beispiele in Form **eines** PDF-Dokuments über TUWEL ab. Gehen Sie dabei folgendermaßen vor:

- TUWEL (`https://tuwel.tuwien.ac.at`)
  Kurs *186.813 Algorithmen und Datenstrukturen 1 (VU 6.0)*
- Thema *Beispielblatt 1*
- Button *Datei hochladen*
  (Nur PDF-Format erlaubt, Größe maximal 10 MB.)

Bitte beachten Sie:

- Sie können **vor** der Deadline beliebig oft Ihre Lösung hochladen, aber **nach** der Deadline ist **kein** Hochladen mehr möglich!
- Die Beispielblätter werden von Tutoren durchgesehen und kommentiert. Sie sehen diese Kommentare als Feedback in TUWEL unter *Beispielblatt 1*. Diese Kommentare sollen Ihnen helfen, Ihr Verständnis für die Beispiele zu vertiefen, sollen aber nicht als Musterlösungen angesehen werden.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen einreichen.
- Bitte geben Sie Ihren Namen, Matrikelnummer und E-Mail-Adresse in den Ausarbeitungen an.

Aufgabe 1 Die Folge der sogenannten *Fibonacci-Zahlen* $\langle f_n \rangle = \langle 1, 1, 2, 3, 5, 8, \ldots \rangle$ kann mit dem folgenden Algorithmus berechnet werden:

```
fib(n):
  falls n == 1 ∨ n == 2 dann {
    retourniere 1;
  } sonst {
    retourniere fib(n − 1) + fib(n − 2);
  }
```

- Geben Sie die Anzahl der vorgenommenen rekursiven Aufrufe bei der Berechnung von f_{10}

  f_{10} *benötigt 109 Aufrufe.*

- Geben Sie eine obere Schranke für f_n in O-Notation in Abhängigkeit von n an und beweisen Sie die Gültigkeit der gewählten Schranke. Ist Ihre obere Schranke auch gleichzeitig eine untere Schranke?

  $\text{fib}(n) = O(2^n)$

  2^n ist keine untere Schranke

- Schreiben Sie einen Algorithmus in detailliertem Pseudocode der f_n in $O(n)$ Schritten berechnet.

  Algorithmus: `fibonacci iterativ(` A`)`
  Eingabe: n
  Rückgabewert: r Berechneter fibonacci Wert

  falls $n == 1$ oder $n == 2$ **dann {**
    $r = 1;$
  } sonst {
    $n1 = n2 = 1$
    $i = 3$
    solange $i < n + 1$ **{**
      $n3 = n1 + n2$
      $n1 = n2$
      $r = n2 = n3$
      $i = i + 1$
    }
  }

Aufgabe 2 Bestimmen Sie die Laufzeiten der unten angegebenen Algorithmen in Abhängigkeit von n in Θ-Notation. Verwenden Sie hierfür einen möglichst einfachen Term.

A

$a = 10000 * \text{Zufallszahl} \in 1, ..., 10;$
solange $a > 0$ **{**
  für $b = 1, \ldots, n/30$ **{**
    $c = 2b;$
  }
  $a = \lfloor a/2 \rfloor;$
}

B

$i = 1;$
$a = 1;$
wiederhole
  $i = i * 2;$
  $a = a * a;$
bis $i \geq \log_2 n;$
für $j = 0, \ldots, a$ **{**
  $x = x + j;$
}

A) $\Theta(n)$
B) $\Theta(ld(ld(n)))$

Aufgabe 3 Ergänzen Sie die folgenden Algorithmen in beliebiger Weise so, dass sie die angegebenen Laufzeiten erreichen:

C $\Theta(n^2 \cdot \log_2 n)$

für $l = 1, \ldots, n^2$ **{**
  $k = n;$
  wiederhole
    $k = \lfloor k/2 \rfloor;$
    $m = n * k;$
  bis $k \leq 1$
}

D $\Theta(\log^2 n)$

$x = \log^2 n;$
solange $x > 1$ **{**
  $y = n^2 - x;$
  $x = x - 1;$
}

Aufgabe 4 Zeigen Sie, dass

$$10^n = O(n!)$$

gilt, indem sie konkrete Werte für n_0 und c finden, und beweisen, dass die entsprechenden Ungleichungen mit diesen Konstanten erfüllt werden können.

$O(g(n)) = \{f(n) | (\exists c, n_0 > 0), (\forall n \geq n_0) : 0 \leq f(n) \leq cg(n)\}$

$0 \leq 10^n \leq n! * c$
$n = 11, c = 3000$ ✓
$10^{n+1} \leq c * (n+1)!$
$10^n * 10 \leq c * n * n!$
$10^n * \frac{10}{n} \leq c * n!$
$\lim_{n \to \infty} 10^n * \frac{10}{n} \leq c * n!$
$10^n * 0 \leq c * n!$ ✓

Aufgabe 5 Schreiben Sie den Pseudocode für ein neues, rekursives Sortierverfahren. Der Algorithmus wird mit einer Folge A der Länge l und zwei Indizes i und j aufgerufen. Es gilt $1 \leq i \leq j \leq l$. Das Prinzip des Algorithmus soll es sein, erst das kleinste Element in der Teilfolge $A[i]$ bis $A[j]$ an Position i zu bringen und das größte an Position j. Dann werden die Elemente zwischen i und j rekursiv sortiert, indem der Algorithmus sich selbst aufruft. Achten Sie darauf, dass man eine Folge aus einem einzelnen Element nicht sortieren muss.

Algorithmus: `new sort`$(var\ A, i, j)$)
Eingabe: Feld A, Index i, j in Feld A
Rückgabewert: Sortiertes Feld A

falls $j - i \leq 1$ **dann {**
  return A;
} sonst {
  für $x = i; x < j; x\ ++$ **{**
    falls $A[x].key \leq A[low].key$ **dann {**
      $low = x$
    }
  }Vertausche $A[i]$ mit $A[low]$;
  für $x = j - 1; x > i; x - -$ **{**
    falls $A[x].key \geq A[high].key$ **dann {**
      $high = x$
    }
  }Vertausche $A[j]$ mit $A[high]$;
  $sort(A, i + 1, j - 1)$;
}

Aufgabe 6 Sortieren Sie die unten angegebene Folge von Zahlen *aufsteigend* mit Hilfe von *Sortieren durch Verschmelzen (Merge-Sort)* und geben Sie die Zahlenfolge nach jedem kompletten Verschmelzungsschritt an. Markieren Sie (auch in der Eingabefolge) die jeweiligen Grenzen zwischen den bereits sortierten Teilfolgen.

$$\langle 14, 3, 5, 12, 7, 2, 19, 12, 1 \rangle$$

[14	3	5	12]	[7	2	19	12	1]
[14	3]	[5	12]	[7	2]	[19	12	1]
[14]	[3]	[5]	[12]	[7]	[2]	[19]	[12	1]
[14]	[3]	[5]	[12]	[7]	[2]	[19]	[12]	[1]
ab hier sind die Teillisten sortiert								
[3	14]	[5	12]	[2	7]	[19	1	12]
[3	5	12	14]	[2	7]	[1	12	19]
[1	2	3	5	7	12	12	14	19]

Aufgabe 7 Sortieren Sie die folgende Zahlenfolge mit Hilfe von *Quicksort*

$$\langle 111, 11, 1, 1, 110, 111, 1 \rangle.$$

Schreiben Sie dabei nach jedem Schritt des Algorithmus die entstandene Zahlenfolge auf und markieren Sie jeweils die aktuellen Pivotelemente, alle bereits sortierten Elemente und die im jeweiligen Schritt vertauschten Elemente.

Geben Sie eine beliebige Eingabefolge an, die für Quicksort (Implementierung laut Skriptum, Pivotlement ist das rechteste Element) einen *Worst-Case* bezogen auf die Anzahl der Schlüsselvergleiche darstellt. Wie viele Schlüsselbewegungen und Schlüsselvergleiche, in Θ-Notation in Abhängigkeit der Eingabegröße n, werden für Ihre *Worst-Case* Eingabefolge benötigt?

[.] = bereits sortierte Elemente
$\downarrow$ = vertauschte Elemente
$\underline{1}$ = pivot Element

$\downarrow$			$\downarrow$			
111	11	1	1	110	111	$\underline{1}$
	$\downarrow$	$\downarrow$				
1	11	1	111	110	111	$\underline{1}$
		$\downarrow$				$\downarrow$
1	1	11	111	110	111	$\underline{1}$
			$\downarrow$			$\downarrow$
1	$\underline{1}$	[1]	111	110	111	$\underline{11}$
[1	1	1	11]	110	111	$\underline{111}$
[1	1	1	11]	110	$\underline{111}$	[111]
[1	1	1	11	110	111	111]

Worst-case Beispiel: aufsteigend sortierte Folge
$\langle 1, 2, 3, 4, 5, 6, 7, 8, 10 \rangle$.

$C_{worst}(n) = \Theta(n^2)$
$M_{worst}(n) = \Theta(n * log(n))$

Aufgabe 8 Gegeben sei der Algorithmus `sort(`A, i`)`, wobei i ein Index im Feld A mit den zu sortierenden Elementen $A[1] \dots A[n]$ ist. Der Schlüsselwert eines Elementes $A[i]$ wird im Attribut $A[i].key$ gespeichert. Der Algorithmus wird durch den Aufruf `sort(`A, n`)` gestartet.

Algorithmus: `sort(var` A, i`)`
Eingabe: Feld A, Index i in Feld A
Rückgabewert: Sortiertes Feld A

```
falls i > 1 dann {
  max = i;
  j = i − 1;
  solange j > 0 {
    falls A[j].key > A[max].key dann {
      max = j;
    }
    j = j − 1;
  }
  maxelement = A[max];
  A[max] = A[i];
  A[i] = maxelement;
  i = i − 1;
  A = sort(A, i);
}
return A;
```

- Auf welchem aus der Vorlesung bekannten Sortierverfahren beruht `sort()`?
  Selectionsort

- Geben Sie die Laufzeit von `sort()` im Best- und im Worst-Case in Θ-Notation in Abhängigkeit der Anzahl der zu sortierenden Element n an.

C_{best}	C_{worst}	M_{best}	M_{worst}
$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(1)$	$\Theta(n)$

- Wenden Sie den Algorithmus `sort()` auf die Zahlenfolge $\langle 9, 9, 1, 5, 1\rangle$ an. Geben Sie die Zahlenfolge jeweils bei dem Aufruf von `sort()` an.
  $\langle 9, 9, 1, 5, 1\rangle$
  $\langle 1, 9, 1, 5, 9\rangle$
  $\langle 1, 5, 1, 9, 9\rangle$
  $\langle 1, 1, 5, 9, 9\rangle$

- Wie müssten Sie den Algorithmus abändern, damit die Elemente der Felder in umgekehrter Reihenfolge sortiert werden?

  In der 5en Zeile, bei Falls ... müsste der Vergleichsoperator umgedreht werden.