Albert-Ludwigs-Universität
Institut für Informatik
Junior-Prof. Dr. Robert Elsässer
Marco Muñiz
Phillip Heidegger

23.3.2010

Klausur

Algorithmentheorie, WS 2009/2010

Die Klausur besteht aus 7 Aufgaben und 16 Seiten. Aus Ihrer Klausurpunktzahl und dem Bonus aus den Übungsaufgaben wird die Endnote berechnet. Der Bonus wird nicht angerechnet, wenn die Klausur nicht bestanden ist. In der Klausur können maximal 70 Punkte erreicht werden; mit mindestens 35 gilt die Klausur als bestanden. Zur Bearbeitung haben Sie 90 Minuten zeit.

Schreiben Sie bitte auf jedes Blatt Ihren Namen sowie Ihre Matrikelnummer.

Schreiben Sie Ihre Lösung bitte in die vorgesehenen Platzhalter nach jeder Aufgabe. Sollte der Platz nicht ausreichen, erhalten Sie auf Anfrage weiteres Papier.

Zugelassene Hilfsmittel: Nicht programmierbare Taschenrechner sind zugelassen.

Viel Erfolg!

Name: _____

Matrikel-Nr.: _____

Studienfach: _____

Punkte (nicht von Studierenden auszufühlen)

Aufgabe 1		von 10	
Aufgabe 2		von 10	
Aufgabe 3		von 10	
Aufgabe 4		von 10	
Aufgabe 5		von 10	
Aufgabe 6		von 10	
Aufgabe 7		von 10	
Bonuspunkte			
Summe		von 70	

Note:

Name: _____ Matrikel-Nr.: _____

Aufgabe 1 - Fast Fourier Transformation [Punkte: 3+7]

1. Die Fast Fourier Transformation nutzt die Eigenschaften der komplexen Einheitswurzeln. Erklären Sie, wie diese Eigenschaften den Einsatz des Divide-and-Conquer Ansatzes ermöglicht.

2. Berechnen Sie das Produkt der Polynomen $p(x) = 5x+2$ and $q(x) = 3x+7$ mit Hilfe der Fast Fourier Transformation.
   - Berechnen Sie die DFT von $p(x)$ und $q(x)$.
   - Geben Sie die Punkt-Wertdarstellung von pq an.
   - Berechnen Sie die Interpolation mit Hilfe des FFT Algorithmus.

Name: _____ Matrikel-Nr.: _____

Name: _____ Matrikel-Nr.: _____

Aufgabe 2 - Randomisierte Algorithmen [Punkte: 5+5]

1. Sei A ein Array der Länge n, das n ganze Zahlen enthält. Entwerfen Sie einen randomisierten Algorithmus, der 1 zurückgibt, wenn x in A vorkommt und 0 sonst. Die Fehlerwahrscheinlichkeit ihres Algorithmus darf nicht schlechter als $\frac{1}{\sqrt{e}}$ sein. Ihr Algorithmus darf höchstens $\frac{n}{2}$ mal auf A zugreifen. Sie dürfen die Funktion $rand(A)$ verwenden, die eine zufällige Stelle in A zurückliefert.
   Beweisen Sie, dass Ihr Algorithmus die gewünschten Eigenschaften erfüllt:

   - Der Algorithmus greift höchstens $\frac{n}{2}$ mal auf A zu.
   - Die Fehlerwahrscheinlichkeit ist höchstens $\frac{1}{\sqrt{e}}$.

   Hinweis: $(1 - \frac{1}{n})^n \leq \frac{1}{e}$

2. Betrachten Sie die Zahl $n = 1105$. Verwenden Sie den randomisierten Primzahltest-Algorithmus mit $a = 7$ um zu überprüfen, ob n möglicherweise eine Primzahl ist oder nicht. Geben Sie jeden rekursiven Aufruf und jeden Zwischenwert von result an.

```
bool isProbPrime;

bool primeTest(int n)
{
  isProbPrime = true;
  a = 7;
  result = power(a,n-1,n);
  if (result !=1 || !isProbPrime)
    return false;
  else
    return true;
}
```

```
int power(int a, int p, int n) {
  if (p==0)
    return 1;
  x = power(a,p/2,n);
  result = (x*x)%n;
  if (result==1 && x!=1 && x!=n-1) {
    isProbPrime = false;
  }
  if (p%2==1)
    result = (a*result)%n;
  return result;
}
```

Name: _____ Matrikel-Nr.: _____

Name: _____ Matrikel-Nr.: _____

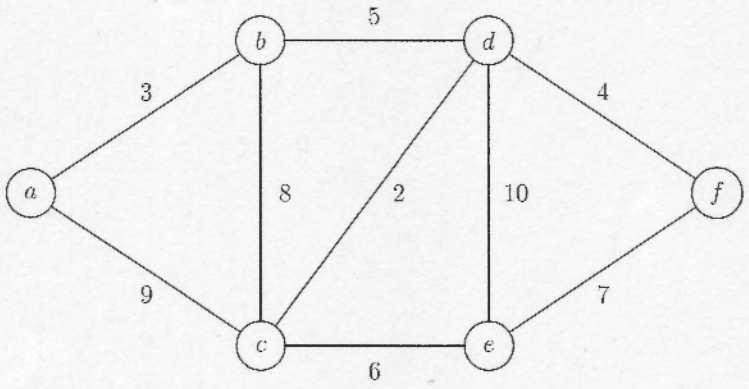
Name: _____ Matrikel-Nr.: _____

Aufgabe 3 - Minimale Spannbäume

[Punkte: 10]

Betrachten Sie den untenstehenden ungerichteten Graphen $G = (V, E, w)$. Berechnen Sie mithilfe des Prims Algorithmus und einem Fibonacci-Heap Q ausgehend vom Knoten a den minimalen Spannbaum T von G. Geben Sie zu jedem Schritt den entstandenen minimalen Spannbaum T' und den zugehörigen Fibonacci-Heap Q' an. Nehmen Sie folgendes für den Fibonacci-Heap an:

- Die Methoden *Q.insert* und *Q.consolidate* beginnen rechts vom aktuellen Minimum.
- Wenn zwei Knoten den gleichen Schlüsselwert haben, sind die Knoten alphabetisch nach ihren Namen zu ordnen (z.B. $a \prec b$).



Name: _____ Matrikel-Nr.: _____

Name: _____ Matrikel-Nr.: _____

Aufgabe 4 - RSA

[Punkte: 6+2+2]

Für eine RSA Verschlüsselung sei $p = 17$, $q = 23$ und $e = 5$.

1. Führen Sie den erweiterten Euklid–Algorithmus zur Berechnung der Zahl d aus und geben Sie die Ausgabe jedes rekursiven Aufrufes an.
2. Verschlüsseln Sie die Nachricht $M = 12$ mit Hilfe des öffentlichen Schlüssels.
3. Entschlüsseln Sie die Nachricht $M' = 53$.

Name: _____ Matrikel-Nr.: _____

Name: _____ Matrikel-Nr.: _____

Aufgabe 5 - Dynamische Programmierung [Punkte: 2+4+4]

1. Beschreiben Sie die Hauptprinzipien der Dynamischen Programmierung.

2. Seien $A_1, A_2, \ldots, A_{n-1}$ Matrizen mit den entsprechenden Dimensionen $p_1, p_2, \ldots, p_n$. Geben Sie die optimale Klammerung für $A_1 \cdot A_2 \cdot \ldots \cdot A_{n-1}$ an für den Fall, dass $p_i < p_{i+1}$ für $i \in \{1, \ldots, n\}$ gilt.

3. Beweisen Sie Ihre Antwort.

Name: _____ Matrikel-Nr.: _____

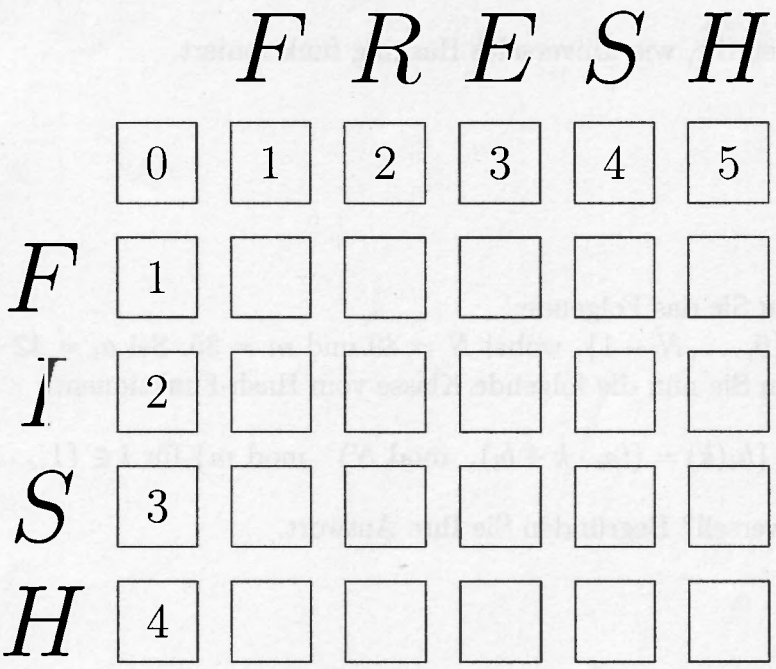
Name: _____ Matrikel-Nr.: _____

Aufgabe 6 - Editier-Distanz

[Punkte: 6+2+2]

Betrachten Sie die beiden Zeichenketten $A = FRESH$ und $B = FISH$.

1. Zeigen Sie den dazugehörigen Spurgraphen für die Transformation von A nach B. Für jeden Knoten sind nur die zulässigen Kanten einzuzeichnen (d.h. Kanten, die zum minimalen Wert des Knotens führen).



2. Markieren Sie einen optimalen Pfad (d.h. einen optimalen Pfad innerhalb des Spurgraphen).

3. Spezifizieren Sie die dazugehörige Sequenz von Editeroperationen und $D(A, B)$. Erklären Sie Ihre Lösung.

Name: _____ Matrikel-Nr.: _____

Aufgabe 7 - Hashing

[Punkte: 3+3+4]

1. Beschrieben Sie, wie perfektes Hashing funktioniert.

2. Beschreiben Sie, wie universales Hashing funktioniert.

3. Betrachten Sie das Folgende:
   Sei $U = \{0, \ldots, N-1\}$, wobei $N = 49$ und $m = 35$. Sei $a_i = 42 \cdot i$ und $b_i = 28 \cdot i$.
   Betrachten Sie nun die folgende Klasse vom Hash-Funktionen:

   $$\mathcal{H} = \{h_i(k) = ((a_i \cdot k + b_i) \mod N) \mod m\} \text{ für } i \in \{1, \ldots, N(N-1)\}$$

   Ist $\mathcal{H}$ universell? Begründen Sie Ihre Antwort.