Image Analysis Lab
Jun.-Prof. Dr.
Olaf Ronneberger
Claudius Korzen

Algorithmen und Datenstrukturen (ESE + IEMS) WS 2013 / 2014

http://lmb.informatik.uni-freiburg.de/lectures/

UNI FREIBURG

Klausur / Exam

Freitag, 28. Februar 2014, 16.00 Uhr - 18:00 Uhr, Gebäude 082, Raum 00-006 (Kinohörsaal)

Friday, Februar 28, 2014, 4.00 pm - 6.00 pm, building 082, room 00-006 (Kinohörsaal)

Nachname: / Last name: _____

Vorname: / First name: _____

Matrikelnummer: / Matriculation number: _____

Studiengang: / Degree course: ◯ **Bachelor** ◯ **Master**

Durch den Antritt dieser Prüfung erklären Sie sich für prüfungsfähig. Sollten Sie sich während der Prüfung nicht prüfungsfähig fühlen, können Sie aus gesundheitlichen Gründen auch während der Prüfung von dieser zurücktreten. Gemäß der Prüfungsordnungen sind Sie verpflichtet, die für den Rücktritt oder das Versäumnis geltend gemachten Gründe unverzüglich (innerhalb von 3 Tagen) dem Prüfungsamt durch ein Attest mit der Angabe der Symptome schriftlich anzuzeigen und glaubhaft zu machen. Weitere Informationen hierzu können auf den Internetseiten des Prüfungsamtes nachgelesen werden.

By taking part in this exam you agree that there are no health reasons that hinder you taking part in the exam. Should you feel that you have health problems during the exam, you may abort this exam. In such a case you must provide an official medical certificate by a doctor including the symptoms within 3 days. More details can be obtained at the examination office.

Unterschrift des Studenten: / Student signature: _____

Punktzahl/Score:

A1	A2	A3	A4	$\sum$
/15	/20	/20	/25	/80

Note/Grade: _____

Einsicht am: _____

Unterschriften der Prüfer: _____

Es gibt 4 Aufgaben. Sie haben insgesamt 120 Minuten Zeit. Das sind pro Aufgabe im Durchschnitt 30 Minuten.

Sie dürfen eine beliebige Menge an Papier, Büchern, etc. verwenden. Sie dürfen keinerlei elektronische Geräte wie Notebook, Mobiltelefon, etc. verwenden, insbesondere keine Geräte, mit denen Sie mit Dritten kommunizieren oder sich mit dem Internet verbinden können. Bei einem Täuschungsversuch wird die Prüfungsleistung mit *nicht ausreichend* (5,0) bewertet. Bereits der Besitz unzulässiger Hilfsmittel zählt als Täuschungsversuch.

Sie dürfen Ihre Lösungen entweder in Deutsch oder Englisch aufschreiben. Schreiben Sie Ihre Lösungen bitte wenn möglich auf die Klausur! Benutzen Sie dabei für jede Aufgabe zuerst die Vorderseite und dann die Rückseite. Wenn Sie zusätzliches Papier benötigen, schreiben Sie auf jedes dieser Blätter bitte oben rechts gut lesbar Ihren Namen (in BLOCKBUCHSTABEN) und Ihre Matrikelnummer.

Tipp: Verbringen Sie nicht zu viel Zeit mit einer einzelnen Teilaufgabe. Wenn Sie nicht weiterkommen, machen Sie erstmal mit einer anderen (Teil-)Aufgabe weiter. Widmen Sie sich dann am Ende, wenn Sie noch Zeit haben, den Teilen, bei denen Sie Schwierigkeiten hatten.

Wir wünschen Ihnen viel Erfolg!

There are 4 tasks. You have 120 minutes of time overall. That is 30 minutes per task on average.

You are allowed to use any amount of paper, books, etc. You are not allowed to use any computing devices or mobile phones, in particular nothing with which you can communicate with others or connect to the Internet. Any attempt to cheat will result in the grade “failed” (5.0). This already includes the possession of forbidden material.

You may write down your solutions in either English or German. Please write down your solutions on this hand-out! You can also use the back side of the pages. If you need additional pages, please write your matriculation number and your name (in PRINTED LETTERS) on each of them.

Hint: Don’t waste your time on a single subtask. If you are unable to solve a task, continue with another (sub-)task first. Come back to such a task at the end, if you have enough time left over.

Good luck!

Aufgabe 1 (Heaps, 15 Punkte)

1.1 (5 Punkte)

Der folgende vollständige binäre Min-Heap ist in einem linearen Array gespeichert. Zeichnen sie diesen Heap als Baum-Diagramm. Markieren Sie die Kanten, an denen die Heapeigenschaft verletzt ist.

The following complete binary min-heap is stored in a linear array. Draw this heap as tree diagram. Mark the edges where the heap property is violated.

```
[1,6,2,4,7,3,9,11,13,12,14,5,15,8,10,13,13]
```

1.2 (5 Punkte)

In dem folgenden binären Min-Heap wird Element 42 auf 3 reduziert. Welche Operationen müssen ausgeführt werden, um den Heap zu reparieren? Skizzieren Sie die Operationen mit Pfeilen am linearen Array und schreiben Sie den resultierenden Heap als lineares Array auf.

In the following binary min-heap, the element 42 is reduced to 3. Which operations are necessary to repair the heap? Draw the operations as arrows to the linear array, and write down the resulting heap as linear array.

```
[1,4,6,6,7,7,9,13,42,12,14,8,15,11,10,15,14,43]
```

1.3 (5 Punkte)

Geben Sie alle Repräsentationen von $\{1, 2, 3, 4\}$ in Form eines binären Min-Heaps an. Zeichnen Sie die Bäume und die dazugehörigen linearen Arrays.

Give all representations of $\{1, 2, 3, 4\}$ as a binary min-heap. Draw the trees and the corresponding linear arrays.

Aufgabe 2 (Hashing & Sortieren, 20 Punkte)

2.1 (4 Punkte)

Sei $h(x) = 3x \mod 7$. Zeichnen Sie den Zustand der zugehörigen Hashtabelle der Größe 7 nach dem Einfügen der Zahlen $\{1, 2, 3, 7, 8, 10, 15, 16\}$. Kollidierende Elemente sollen in einer verketteten Liste gespeichert werden.

Let $h(x) = 3x \mod 7$. Write down the state of the corresponding hash table of size 7 after inserting the numbers $\{1, 2, 3, 7, 8, 10, 15, 16\}$. Colliding elements are stored in a linked list.

2.2 (8 Punkte)

Gegeben sei das Universum $U = \{1, 2, \ldots, n\}$ und die Menge $H = \{h_1, h_2, \ldots, h_6\}$ von Hashfunktionen, mit:

$$h_a(x) = (a \cdot x) \bmod 7$$

Wie groß darf n maximal gewählt werden, damit H $\frac{7}{6}$-universell ist? Mit Begründung.

Given the universe $U = \{1, 2, \ldots, n\}$ and the set $H = \{h_1, h_2, \ldots, h_6\}$ of hash functions, with:

$$h_a(x) = (a \cdot x) \bmod 7$$

What is the maximum value for n, such that H is $\frac{7}{6}$-universal and why?

2.3 (8 Punkte)

Nachdem Sie aus Ihrem Urlaub zurückkommen, zeigt Ihnen die Anrufliste Ihres Telefons n entgangene Anrufe mit k unterschiedlichen Telefonnummern an, sortiert nach dem Anrufzeitpunkt. Sie möchten sich daraus eine neue Liste erstellen, die ausschließlich die k unterschiedlichen Telefonnummern, numerisch sortiert, enthält.

Schreiben Sie (in Java oder C++) dazu eine Methode *int[] reduceAndSort(int[] numbers)*, (statt *int[]* verwenden Sie in C++ bitte *std::vector<int>*), die diese Liste in Zeit $\mathcal{O}(k \log k + n)$ erzeugt. Begründen Sie die Laufzeit ihrer Methode.

Tipp: Benutzen Sie Hashing und Sortieren. Sie dürfen die Methoden der Standardbibliotheken benutzen, z.B. *HashMap* und *Collections.sort()* in Java bzw. *std::unordered_map* und *std::sort()* in C++.

Assume, that you are returning from your holiday. In the call list of your telephone, there are n missed calls with k different telephone numbers, which are sorted by arrival time. You would like to create a new list, containing only the k different numbers and where the numbers are sorted numerically.

Write down (in Java or C++) a method *int[] reduceAndSort(int[] numbers)*, (use *std::vector<int>* instead of *int*[] in C++), which creates such a list in time $\mathcal{O}(k \log k + n)$. Substantiate the runtime of your method.

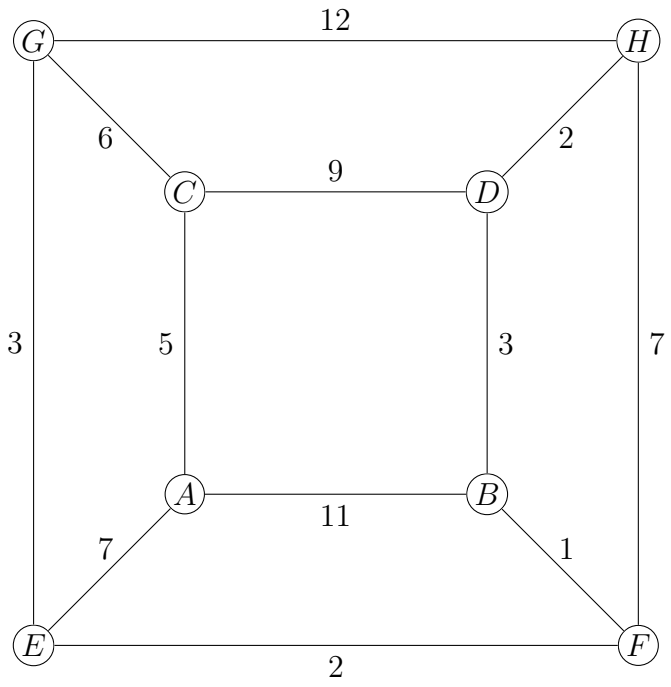
Hint: Use hashing and sorting. You are allowed to use the methods of the standard libraries, e.g. *HashMap* and *Collections.sort()* in Java / *std::unordered_map* and *std::sort()* in C++.

Aufgabe 3 (Graphen, 20 Punkte)

3.1 (5 Punkte)

Führen Sie Dijkstra's Algorithmus auf dem folgenden ungerichteten Graphen aus, mit Startknoten A und Zielknoten H. Veranschaulichen Sie den kompletten Fortgang des Algorithmus. Aus Ihrer Zeichnung muss ersichtlich sein: (1) welcher Knoten in welcher Iteration gelöst wird; (2) alle vom Algorithmus berechneten (Zwischen-)Distanzen auf dem Weg zur optimalen Lösung. Zeichnen Sie den kürzesten Weg von A nach H ein. Verwenden Sie der Lesbarkeit halber bitte mindestens zwei verschiedene Farben.

Consider the following undirected graph. Simulate a run of Dijkstra's algorithm from the start node A until the target node H. Demonstrate the complete run of the algorithm. It should be visible from your drawing: (1) which nodes were settled in which order; (2) all the tentative and final distances computed. Draw the shortest path from node A to node H. Please use at least two different colors.



3.2 (5 Punkte)

Beweisen oder widerlegen Sie folgende Behauptung (in beiden Fällen reicht eine kurze, aber präzise Begründung):

Dijkstra's Algorithmus findet auch dann den kürzesten Weg, wenn er in jedem Iterationsschritt folgendermaßen modifiziert wird: Falls innerhalb der aktiven Knoten keine Kante vom billigsten Knoten zum zweitbilligsten Knoten existiert, dann kann auch zuerst der zweitbilligste Knoten gelöst werden.

Prove or disprove the following claim (in both cases, give a short but precise argument):

Dijkstra's algorithm also finds the shortest path, if every iteration step is modified in the following manner: If there is no direct edge between the cheapest node and the second cheapest node within the active nodes, you can also settle the second cheapest node first.

3.3 (10 Punkte)

Sie sollen einen geeigneten Standort für die neue Feuerwache im Ort finden. Dieser Standort soll so gewählt werden, dass die Feuerwehr im Brandfall jedes Haus möglichst schnell erreichen kann. Das bedeutet, dass die mittlere Anfahrtszeit minimal sein soll.

Das Straßennetzwerk des Ortes sei gegeben durch den ungerichteten Graphen G. Die Knoten von G seien die n Häuser (indiziert von 0 bis $n-1$), die Kanten seien die Straßen und die Kantengewichte seien die Fahrzeiten. Schreiben Sie (in Java oder C++) eine Methode *int computeLocation(Graph G)*, die den optimalen Standort (den Index des Knoten) für die neue Feuerwache in G berechnet.

Sie dürfen dabei annehmen, dass Sie eine Methode *int[] computeDijkstra(Graph G, int i)* (in C++: *std::vector<int>* statt *int[]*) zur Verfügung haben, die für einen gegebenen ungerichteten Graphen G und einen gegebenen Knoten i die Fahrzeiten zwischen i und allen anderen Knoten im Graphen berechnet. Die Anzahl der Knoten erhalten Sie mit *G.numNodes()*.

You are asked to find a suitable location for the new fire station in the town. The location must be chosen in such a way that, in case of fire, the fire service can reach all buildings as quickly as possible. So the travel time should be minimal on average.

The road network of the town is given by the undirected graph G. The n buildings are given by the nodes of G (indexed from 0 to $n-1$), the roads are given by the edges of G and the travel times are given by the edge weights. Write down (in Java or C++) a method *int computeLocation(Graph G)*, which computes the optimal location (the index of the node) for the new fire station in G.

You can assume, that there is a method *int[] computeDijkstra(Graph G, int i)* (*std::vector<int>* instead of *int[]* in C++) which computes the travel times between the given node i and all other nodes in the given graph G. The number of nodes are available via *G.numNodes()*.

Aufgabe 4 ($\mathcal{O}$-Notation, Vollständige Induktion, 25 Punkte)

4.1 (7 Punkte)

Ordnen sie folgende Funktionen f_1, f_2, f_3, f_4, f_5 nach ihrer Laufzeitkomplexität so, dass $f_i = \mathcal{O}(f_{i+1})$ für $i = 1, 2, 3, 4$. Bestimmen Sie außerdem für welche i gilt, dass $f_i = \Theta(f_{i+1})$ und für welche nicht. Selbstverständlich sollen Sie alle Ihre Entscheidungen begründen, insbesondere auch die i wo *nicht* $f_i = \Theta(f_{i+1})$. Sie können für alle Ihre Begründungen die Grenzwert-Definition von $\mathcal{O}$ und Θ benutzen.

Sort the following functions f_1, f_2, f_3, f_4, f_5 according to their run time complexity, such that $f_i = \mathcal{O}(f_{i+1})$ for $i = 1, 2, 3, 4$. Beside that, determine for which i the condition $f_i = \Theta(f_{i+1})$ is fulfilled. Of course you must justify all your decisions, also for those i, where *not* $f_i = \Theta(f_{i+1})$. You may use for all justifications the limit definition of $\mathcal{O}$ and Θ.

$$n^2$$
$$n \log_{10} n$$
$$n^2 \log_2(n^2)$$
$$\sqrt{n}$$
$$n \log_2(n^2)$$

4.2 (5 Punkte)

Der Zugriff auf Datenstrukturen ist oft durch folgende Rekursion geregelt:

$$T(1) = a,\ T(n) = c + T(n/2)$$

Zeigen Sie, dass $T(n) = \mathcal{O}(\log n)$.

Access to data structures is often governed by the following recurrence:

$$T(1) = a,\ T(n) = c + T(n/2)$$

Show that $T(n) = \mathcal{O}(\log n)$.

4.3 (5 Punkte)

Beweisen Sie per vollständiger Induktion, dass die Reihe der ungeraden Zahlen immer eine Quadratzahl ergibt, also z.B.: $1 + 3 + 5 + 7 + 9 + 11 = 36 = 6^2$

Prove with complete induction, that the series of the odd numbers is always a square number, e.g. $1 + 3 + 5 + 7 + 9 + 11 = 36 = 6^2$

4.4 (8 Punkte)

Was berechnet folgender Algorithmus? Welche Laufzeitkomplexität hat er? Wie können Sie ihn deutlich beschleunigen? Welche Laufzeitkomplexität hat der verbesserte Algorithmus? (*Tipp*: Teilen und Herrschen hilft hier nicht)

What is computed in the following algorithm? What is the runtime complexity? How can you increase its speed significantly? What is the runtime complexity of the improved algorithm? (*Hint*: Divide and Conquer does not help here)

```
int[] f( int[] data, int k) {
  int n = data.length();
  int[] output = new int[n-k]; // array is initialized with zeros
  for (int i = 0; i < n-k; ++i) {
    for (int j = 0; j < k; ++j) {
      ouput[i] += data[i+j];
    }
  }
  return output;
}
```