Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn

Exam Algorithm Theory

Tuesday, March 28, 2017, 09:00-11:00

Name: .....

Matriculation Nr.: .....

Signature: .....

Do not open or turn until told so by the supervisor!

Instructions:

- Write your name and matriculation number on the cover page of the exam and sign the document! Write your name on all sheets!
- Your signature confirms that you have answered all exam questions without any help, and that you have notified exam supervision of any interference.
- Write legibly and only use a pen (ink or ball point). Do **not** use **red**! Do **not** use a pencil!
- You are **not** allowed to use any material except for a dictionary and a hand-written summary of at most 5 A4 pages (corresponds to 5 single-sided A4 sheets!).
- There are 5 problems (with several questions per problem) and there is a total of 120 points. At most 40% are needed to pass the exam, and 80% will net you the best grade, i.e., 24 points are bonus points.
- Use a separate sheet of paper for each of the 5 problems.
- Only one solution per question is graded! Make sure to strike out any solutions that you do not want to be considered!
- **Explain your solutions! Just writing down the end result is not sufficient unless otherwise indicated.**

Question	Achieved Points	Max Points
1		44
2		20
3		20
4		16
5		20
Total		120

Task 1: Short questions (44 Points)

(a) **(6 points)** Use the Master Theorem to solve the following reccurence relation.

$$T(n) = 27 \cdot T\left(\frac{n}{3}\right) + n^3 \log^5(n) \text{ and } T(1) = O(1).$$

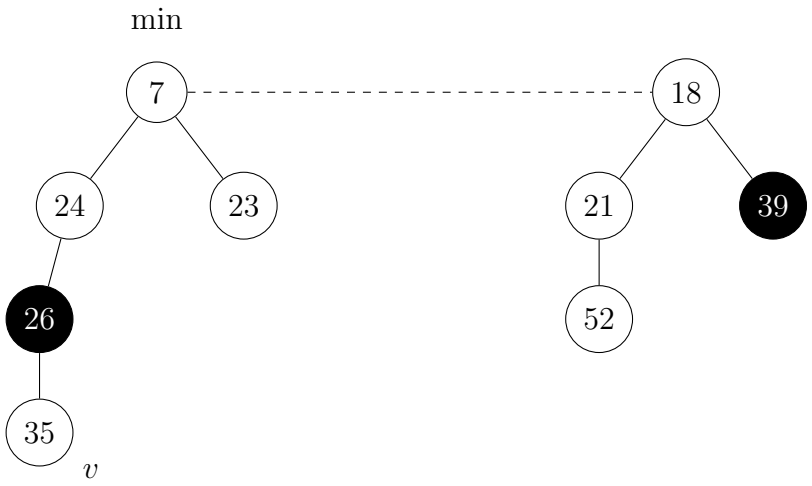
(b) **(7 points)** Suppose we perform a sequence of stack operations *push* and *pop* on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes by applying an operation *copy*. The actual cost of each *push* and *pop* is 1, the actual cost for *copy* is 1 per element in the stack.

Use the **potential function method** to show that there is an assignment of amortized costs such that for a sequence of n operations, each operation, including copying the stack, has only amortized cost at most 2.

(c) **(7 points)** Show that any r-regular bipartite graph has a perfect matching.

Hint: You can use results/theorems proven in the lecture as a blackbox.

(d) **(6 points)** Consider the following Fibonacci heap (black nodes are *marked*, white nodes are *unmarked*). How does the given Fibonacci heap look after **inserting** value 8 and how does it look after a subsequent **decrease-key**$(v, 5)$ operation?



(e) **(6 points)** Let c be a positive integer. A c-edge coloring of a graph (V, E) is a map $\phi : E \to C$ for some set C of cardinality c. We call $\phi(e)$ the color of edge e. A node is called **satisfied** in an edge coloring if all of its incident edges have distinct colors.

Suppose you are given a 3-regular graph[1] with n nodes and m edges and you want to 4-color its edges. Let each edge pick one of the 4 colors uniformly at random.

What is the expected number of satisfied nodes after performing the above algorithm?

[1] A graph is called r-regular if all nodes have degree r. The degree of a node is the number of its neighbors.

(f) **(4 points)** For some project you are supposed to decide between binary heaps and Fibonacci heaps. You can assume that you are given a library implementation of both kinds of heaps.

- Give at least one reason to choose binary heaps.
- Give at least one reason to choose Fibonacci heaps.

(g) **(8 points)**You are given a randomized algorithm called ALG that takes as input an undirected graph G and outputs in linear time a number k with the following property:

With probability at least $1/n$, the number k is the size of a minimum cut of G.

Someone now has the idea to increase the probability of getting the size of a minimum cut by running ALG n times (which results in an $O(n^2)$ algorithm). Is this a reasonable approach and what is the main difference of the above algorithm to the contraction algorithm discussed in the lecture?

Aufgabe 1: Kurze Fragen (44 Punkte)

(a) **(6 Punkte)** Lösen Sie mithilfe des Master Theorems die folgende Rekursionsgleichung:

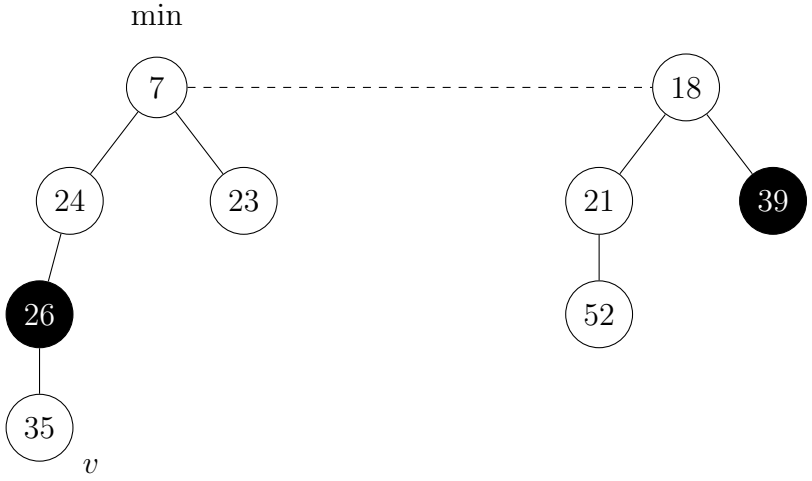
$$T(n) = 27 \cdot T\left(\frac{n}{3}\right) + n^3 \log^5(n) \text{ and } T(1) = O(1).$$

(b) **(7 Punkte)** Gegeben sei eine Folge von *push* und *pop* Operationen auf einem Stack, dessen Größe den Wert k niemals überschreitet. Nach jeweils k Operationen macht man eine Kopie des Stacks mithilfe der Operation *copy*. Die tatsächlichen Kosten der Operationen *push* und *pop* betragen 1, die von *copy* beträgt 1 pro Element im Stack.

Zeigen Sie durch Definition einer geeigneten **Potentialfunktion**, dass für jede Sequenz von Operationen jede der drei Operationen amortisierte Kosten von höchstens 2 hat.

(c) **(7 Punkte)** Zeigen Sie, dass jeder r-reguläre bipartite Graph ein perfektes Matching hat. *Hinweis: Sie können Resultate aus der Vorlesung als Blackbox benutzen.*

(d) **(6 Punkte)** Betrachten Sie folgenden Fibonacci Heap (schwarze Knoten sind markiert, weiße unmarkiert). Wie sieht dieser Fibonacci Heap nach Einfügen (**insert**) des Wertes 8 aus? Wie sieht er nach einer darauf folgenden **decrase-key**$(v, 5)$-Operation aus?



(e) **(6 Punkte)** Sei $c \geq 1$ eine natürliche Zahl. Eine c-Kantenfärbung eines Graphen (V, E) ist eine Abbildung $\phi : E \to C$ von E in eine Menge C der Kardinalität c. Wir nennen $\phi(e)$ die Farbe der Kante e. Ein Knoten heißt **glücklich** in einer Kantenfärbung, wenn alle seine angrenzenden Kanten verschiedene Farben haben.

Sei nun ein 3-regulärer Graph[2] mit n Knoten und m Kanten gegeben. Man konstruiert eine 4-Färbung seiner Kanten, indem man jeder Kante zufällig mit jeweils gleicher Wahrscheinlichkeit eine Farbe zuordnet.

Was ist die resultierende erwartete Anzahl an glücklichen Knoten?

[2]Ein Graph heißt r-regulär, wenn alle Knoten Grad r haben. Der Grad eines Knotens ist die Anzahl seiner Nachbarn.

(f) **(4 Punkte)** Für ein Projekt müssen Sie sich zwischen binären Heaps und Fibonacci Heaps entscheiden. Sie können davon ausgehen, dass eine Bibliotheks-Implementierung für beide Arten von Heaps zur Verfügung steht.

- Geben Sie mindestens einen Grund, binäre Heaps zu verwenden.
- Geben Sie mindestens einen Grund, Fibonacci Heaps zu verwenden.

(g) **(8 Punkte)** Sei ALG ein randomisierter Algorithmus, welcher als Input einen ungerichteten Graphen G erhält und in linearer Zeit eine Zahl k mit folgender Eigenschaft ausgibt:

Mit einer Wahrscheinlichkeit von mindestens $1/n$ *ist* k *die Größe eines minimalen Schnitts von* G*.*

Ein Student hat nun die Idee, die Wahrscheinlichkeit, einen minimalen Schnitt zu erhalten, zu erhöhen, indem man ALG n Mal für den Graphen G ausführt (was einen $\mathcal{O}(n^2)$ Algorithmus ergibt). Ist dies eine vielversprechende Idee? Was ist der hauptsächliche Unterschied zwischen ALG und dem "Contraction"-Algorithmus aus der Vorlesung?

Task 2: Maximal Subarray (20 Points)

Given an array $A[1, \ldots, n]$ with real numbers as entries, you want to find indices $i < j$ such that *sum*$(A[i, \ldots, j])$ is maximal. In mathematical terms that means $\sum_{k=i}^{j} A[k] = \max \left\{ \sum_{k=i'}^{j'} A[k] \;\middle|\; 1 \leq i' < j' \leq n \right\}$. We call $A[i, \ldots, j]$ a maximal subarray.

(a) **(3 points)** Describe a naive brute-force solution of the problem and explain its running time.

(b) **(7 points)** Devise an $\mathcal{O}(n)$-function that finds a maximal subarray $A[i, \ldots, j]$ of $A[1, \ldots, n]$ in the above sense under the additional restriction that $i \leq \lfloor \frac{n}{2} \rfloor < j$ has to hold.

(c) **(10 points)** Devise an $\mathcal{O}(n \log n)$ algorithm to solve the maximal subarray problem and show that it meets the running time.
Hint: If you did not solve part (b), you can use the function described in (b) as a blackbox.

Aufgabe 2: Maximales Subarray (20 Punkte)

Für ein Array $A[1, \ldots, n]$ reeller Zahlen möchte man Indizes $i < j$ finden, sodass *sum*$(A[i, \ldots, j])$ maximal ist. Mathematisch ausgedrückt sollen i und j also so gewählt sein, dass

$$\sum_{k=i}^{j} A[k] = \max \left\{ \sum_{k=i'}^{j'} A[k] \;\middle|\; 1 \leq i' < j' \leq n \right\}.$$

Man nennt $A[i, \ldots, j]$ ein maximales Subarray.

(a) **(3 Punkte)** Beschreiben Sie eine naive Brute-Force Methode um dieses Problem zu lösen und erklären Sie die Laufzeit davon.

(b) **(7 Punkte)** Geben Sie eine $\mathcal{O}(n)$-Funktion an, welche ein maximales Subarray $A[i, \ldots, j]$ von $A[1, \ldots, n]$ berechnet mit der zusätzlichen Eigenschaft, dass $i \leq \lfloor \frac{n}{2} \rfloor < j$ gilt.

(c) **(10 Punkte)** Geben Sie einen $\mathcal{O}(n \log n)$-Algorithmus an, welcher ein maximales Subarray berechnet, und zeigen Sie, dass die Laufzeit Ihres Algorithmus' $\mathcal{O}(n \log n)$ ist.
Hinweis: Wenn Sie Aufgabenteil (b) nicht gelöst haben, können Sie die dort beschriebene Funktion hier als Blackbox verwenden.

Task 3: Bloody Exercise (20 Points)

Enthusiastic celebration of a sunny day at a prominent southwestern university has resulted in the arrival at the university's medical clinic of 169 students in need of emergency treatment. Each of the 169 students requires a transfusion of one unit of blood. The clinic has supplies of 170 units of blood. The number of units of blood available in each of the four major blood groups (A, B, O, AB) and the distribution of patients among the groups is summarized below.

Blood type	A	B	O	AB
Supply	46	34	45	45
Demand	39	38	42	50

Type A patients can only receive type A or O blood, type B patients can receive only type B or O blood, type O patients can receive only type O, and type AB patients can receive any of the four types.

(a) **(5 points)** Give a maximum flow formulation of the above problem. Draw the directed graph and write the edge capacity above each edge. Your network should have 10 vertices: a source (named 0), a supply node for each of the four blood types (named 1 for A, 2 for B, 3 for O and 4 for AB), a demand node for each blood type (named 5 to 8 in the same order), and a sink (named 9).

(b) **(6 points)** Solve the maximum flow problem using the Ford-Fulkerson augmenting path algorithm. Do the first augmentation on the path 0-2-8-9. (This will force you to use a backwards edge at some point during the rest of the algorithm.) Afterwards, always choose the augmenting path with the fewest number of edges, breaking ties in favor of the lexicographically smallest path (e.g., choose 0-2-7-9 over 0-4-6-9). List each of the augmenting paths below. Also, write and circle the final flow values on each edge in of the network.

(c) **(5 points)** Calculate a min s-t cut in the network above, i.e., list the vertices on the source side of the cut.

(d) **(4 points)** Use the min-cut to deduce a rigorous and concise explanation of why not all of the patients can receive a whole blood unit from the available supply. Your explanation should be understandable to the hospital administrators who have no knowledge of network flow theory.

Aufgabe 3: Bloody Exercise (20 Punkte)

Nach einer ausufernden Feier an einer berühmten Universität im Südwesten des Landes benötigen 169 Studierende eine medizinsche Notversorgung in der Universitätsklinik. Jede(r) der 169 Studierenden benötigt eine Einheit Blut zur Transfusion. Die Klinik verfügt über 170 Einheiten Blut. Die verfügbaren und benötigten Bluteinheiten in jeder der vier Blutgruppen A, B, 0, AB sind in folgender Tabelle dargestellt:

Blutgruppe	A	B	0	AB
verfügbar	46	34	45	45
benötigt	39	38	42	50

Typ A Patienten können nur Blut vrom Typ A oder 0 erhalten, Typ B Patienten können nur Typ B oder 0 erhalten, Typ 0 Patienten können nur Typ 0 erhalten und Typ AB Patienten können alle vier Typen erhalten.

(a) **(5 Punkte)** Übersetzen Sie das obige Verteilungsproblem in ein Maximum Flow Problem. Zeichnen Sie dazu den entsprechenden gerichteten Graphen und schreiben Sie an jede Kante ihre Kapazität. Ihr Netzwerk sollte aus 10 Knoten bestehen: Ein Quelle (Nummer 0), einen Verfügbarkeits-Knoten für jede Blutgruppe (Nummer 1 für A, 2 für B, 3 für 0 und 4 für AB), einen Bedarfs-Knoten für jede Blutgruppe (Nummern 5 bis 8 in gleicher Reihenfolge) und eine Senke (Nummer 9).

(b) **(6 Punkte)** Ermitteln Sie einen maximalen Fluss durch Anwendung des Ford-Fulkerson Algorithmus. Führen Sie die erste Augmentation am Pfad 0-2-8-9 durch (dadurch werden Sie später im Algorithmus Rückwärts-Kanten verwenden müssen). Wählen Sie danach stets denjenigen augmentierenden Pfad mit den wenigsten Kanten. Wählen Sie bei gleicher Anzahl den lexikographisch kleineren Pfad (z.B. 0-2-7-9 vor 0-4-6-9). Listen Sie alle augmentierenden Pfade auf. Schreiben Sie ebenso alle finalen Flusswerte an die jeweilige Kante im Netzwerk.

(c) **(5 Punkte)** Berechnen Sie einen minimalen s-t Schnitt im obigen Netzwerk. Listen Sie dazu die Knoten auf, die sich auf der Seite der Quelle befinden.

(d) **(4 Punkte)** Erklären Sie mithilfe des minimalen Schnitts, warum nicht alle Patienten eine (vollständige) Blut-Konserve erhalten können. Ihre Erklärung sollte für Krankenhaus-Personal ohne Wissen über Netzwerk-Flüsse verständlich sein.

Task 4: Node and Edge Coloring (16 Points)

Let $G = (V, E)$ be an undirected graph. For every $v \in V$, the degree of v, $\deg(v)$, is defined as the number of neighbors of v. We call $\Delta := \max\{\deg(v) |\ v \in V\}$ the maximum degree among all nodes in V.

(a) **(6 points)** Construct an algorithm which validly colors a graph with $\Delta + 1$ colors.[3]

Let c be a positive integer. A c-edge coloring of a graph (V, E) is a map $\phi : E \to C$ for some set C of cardinality c. We call $\phi(e)$ the color of edge e. We call an edge coloring **valid** if no two adjacent edges have the same color.

(b) **(10 points)** Construct a greedy approximation algorithm which sequentially colors the edges of a given graph and produces a valid edge coloring. The approximation ratio of your algorithm (w.r.t. the number of used colors) should be less than or equal to 2. Do not forget to prove the approximation ratio.

[3] A valid coloring of a graph is a map $\phi : V \to C$ to some set of colors C such that any two neighboring nodes have different colors, i.e., for each $\{u, v\} \in E$ we have $\phi(u) \neq \phi(v)$.

Aufgabe 4: Knoten- und Kantenfärbungen (16 Punkte)

Sei $G = (V, E)$ ein ungerichteter Graph. Für jedes $v \in V$ ist der Grad von v, $\deg(v)$, definert als die Anzahl der Nachbarn von v. Es ist $\Delta := \max\{\deg(v) | \ v \in V\}$ der maximale Grad.

(a) **(6 Punkte)** Konstruieren Sie einen Algorithmus, der eine gültige Färbung eines Graphen mit $\Delta + 1$ Farben berechnet[4].

Sei $c \geq 1$ eine nat"urliche Zahl. Eine c-Kantenfärbung eines Graphen (V, E) ist eine Abbildung $\phi : E \to C$ von E in eine Menge C der Kardinalität c. Wir nennen $\phi(e)$ die Farbe der Kante e. Eine Kantefärbung heißt **gültig**, falls je zwei adjazente Kanten verschiedene Farben haben.

(b) **(10 Punkte)** Konstruieren Sie einen Greedy Approximationsalgorithmus, welcher eine gültige Kantenfärbung eines Graphen erzeugt. Die Approximations-Rate Ihres Algorithmus (bzgl. der Anzahl der benutzten Farben) soll kleiner oder gleich 2 betragen. Vergessen Sie nicht, die Approximations-Rate zu beweisen.

[4]Eine gültige Färbung eines Graphen ist eine Abbildung $\phi : V \to C$ von V in eine Farbenmenge C, so dass zwei benachbarte Knoten jeweils verschiedene Farben haben. Für $\{u, v\} \in E$ gilt also $\phi(u) \neq \phi(v)$.

Task 5: Online Bin-Packing (20 Points)

The *Online Bin-Packing* problem is a variant of the Knapsack problem. We are given an unlimited number of bins, each of size 1. We get a sequence of items one by one (each of size at most 1), and are required to place them into bins as we receive them. Our goal is to minimize the number of bins we use, subject to the constraint that no bin should be filled to more than its capacity.

In this question we will consider a simple online algorithm for this problem called **First-Fit (FF)**. FF orders the bins arbitrarily, and places each item into the first bin that has enough space to hold the item.

(a) **(8 points)** Show that FF does not provide a competitive ratio better than $3/2$.

(b) **(12 points)** Prove that FF has competitive ratio at most 2.

Aufgabe 5: Online Bin Packing (20 Punkte)

Das *Online Bin-Packing* Problem ist eine Variante des Knapsack-Problems. Gegeben sind beliebig viele Behälter mit der Kapazität 1. Man erhält nun nach und nach Objekte der Größe höchstens 1, die man in die Behälter packen muss. Ziel ist es, die Anzahl der benutzten Behälter zu minimieren unter der Bedingung, dass jeder Behälter nur bis zu seiner Kapazität gefüllt werden darf.

Wir betrachten dazu einen einfachen Online-Algorithmus mit dem Namen **First-Fit (FF)**. FF ordnet die Behälter beliebig und füllt jedes Objekt in den ersten Behälter, der noch genügend Platz dafür hat.

(a) **(8 Punkte)** Zeigen Sie, dass die Competitive Ratio von FF nicht besser als $3/2$ ist.

(b) **(12 Punkte)** Zeigen Sie, dass FF Competitive Ratio höchstens 2 hat.