University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

Algorithm Theory

4th September 2019, 2-4 pm

Name: .....

Matriculation No.: .....

Signature: .....

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

- Put your **student ID** on the table next to you so we can check it.
- Write your **name** and **matriculation number** on this page and **sign** the document.
- Your **signature** confirms that you have answered all exam questions without any help, and that you have notified exam supervision of any interference.
- You are allowed to use a summary of **five handwritten, single-sided A4 pages**.
- Write legibly and only use a pen (ink or ball point). Do **not use red**! Do **not use a pencil**!
- You may write your answers in **English or German** language.
- **No electronic devices** are allowed.
- Only **one solution per task** is considered! Make sure to strike out alternative solutions, otherwise the one yielding the minimal number of points is considered.
- **Detailed steps** might help you to get more points in case your final result is incorrect.
- The keywords **Show...**, **Prove...**, **Explain...** or **Argue...** indicate that you need to prove or explain your answer carefully and in sufficient detail.
- The keywords **Give...**, **State...** or **Describe...** indicate that you need to provide an answer solving the task at hand but without proof or deep explanation (except when stated otherwise).
- You may use information given in a **Hint** without further explanation.
- **Read each task thoroughly** and make sure you understand what is expected from you.
- **Raise your hand** if you have a question regarding the formulation of a task.
- Write your name on **all sheets**!

Task	1	2	3	4	5	**Total**
Maximum	40	20	22	12	26	120
Points						

Task 1: Short Questions *(40 Points)*

(a) Consider a parallel machine with n memory cells $c_1, \ldots, c_n$. Assume there is a value x in memory cell c_1, which must be copied to the other $n-1$ cells $c_2, \ldots, c_n$ of the shared memory. There are $p \geq n$ processors. Formulate parallel algorithms that solve this task as fast as possible for the EREW and the CREW PRAM computational models, respectively. State the according work and depth of your algorithms. *(7 Points)*

(b) We are given a set of n time intervals $[s_1, e_1], \ldots, [s_n, e_n]$ such that starting and ending times, i.e., $s_1, s_2, \ldots, s_n, e_1, e_2, \ldots, e_n$ are $2n$ distinct integers. The goal is to select a largest possible non-overlapping set of intervals. Consider the following strategy; as far as we can, we repeatedly choose the interval with the largest starting time that is not overlapping the already chosen intervals. Prove or disprove whether this greedy approach computes an optimal solution. *(6 Points)*

(c) We are given a weighted directed graph $G = (V, \vec{E})$, where $w : \vec{E} \to \mathbb{R}^+$ defines weights of the edges. We would like to find the maximum-weight acyclic subgraph of G. Provide a $1/2$-approximation algorithm for the problem. Show the correctness of your algorithm.

Hint: Start with an arbitrary global order of the nodes. *(8 Points)*

(d) We are given a directed weighted graph $G = (V, E)$, where $w : \vec{E} \to \mathbb{R}^+$ defines weights of the edges. Consider also a function $b : V \to N$ that defines some indegree bound for each node. We would like to find a subset $E' \subseteq E$ of maximum total weight such that every node $u \in V$ has indegree at most $b(u)$ in graph $G' = (V, E')$. Show that the set of feasible solutions form a matroid and thus, this problem can be solved by using the greedy algorithm for matroids. *(7 Points)*

(e) In the lecture, we studied a binary counter that supports a single operation for incrementing the counter by 1 and where the cost of such an `increment` operation equals the number of bits that need to be changed in the binary representation of the counter value. Suppose the counter is initialized to a value containing b 1s in its binary representation. Show that the cost of performing n `increment` operations is $O(n + b)$ (do not assume b is constant). *(6 Points)*

(f) Given a regular, bipartite graph, show that it has a perfect matching. *(6 Points)*

Solution Task 1

Task 2: Load Balancing *(20 Points)*

Consider the load balancing problem from the lecture, where n jobs with processing times $t_1, \ldots, t_n$ must be scheduled on m machines $M_1, \ldots, M_m$ to minimize the maximum total load on any machine (i.e., minimize the makespan T). For this task, assume that $t_{i+1} \leq \alpha t_i$ for all $1 \leq i < n$ and some constant $0 < \alpha < 1$.

We use the simple "longest processing time first" (LPT) scheduling algorithm, where we iterate through the jobs by order of descending processing time and assign the current job to the machine with the currently lowest load.

Hint: $\sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha}$.

(a) Show that LPT is optimal for $\alpha \leq \frac{1}{2}$. *(5 Points)*

(b) Show that LPT gives a $\left(\frac{\alpha}{1-\alpha}\right)$-approximation for $\alpha > \frac{1}{2}$. *(6 Points)*

(c) Show that LPT gives a $\left(\frac{1+2\alpha}{1+\alpha}\right)$-approximation (better than the above for $\alpha > \frac{1}{2}$ close to 1). *(9 Points)*

Solution Task 2

Task 3: Online Bin packing *(22 Points)*

Consider the following online bin packing problem. A sequence of items each with some size in the interval $(0, 1]$ arrives one-by-one in an online fashion. Additionally there are bins where all items must be packed into. The sum of the sizes of items inside any bin can be at most 1. Each item must be packed into a bin as soon as it arrives and that decision is final. The goal is to minimize the number of required bins to package all items.

We introduce the Next First (NF) algorithm: Initially take an empty bin as current bin. Put arriving items into that bin as long as they fit. If an arriving item does not fit, close the bin and take a new, empty bin as current bin and put the item in there. For a given sequence of items I, let NF(I) be the number of bins required by the online Next First algorithm and let OPT(I) be the number of bins required by an optimal *offline* algorithm.

(a) Show that NF is strictly 2-competitive (i.e., show that for every sequence of items I, NF$(I) \leq 2 \cdot$OPT(I)). *(7 Points)*

(b) Show that there is a fixed constant $c > 0$, such that for any positive n, there is an input sequence I of n items, where NF$(I) \geq 2 \cdot$OPT$(I) - c$. *(8 Points)*

(c) Show that for *every deterministic* online bin packing algorithm $\mathcal{A}$, there is an input sequence I for which the algorithm uses at least $\mathcal{A}(I) \geq \frac{3}{2} \cdot$OPT$(I)$ bins. *(7 Points)*

Hint: Think of a short worst case sequence. If you can show the same statement for a smaller constant $1 < c < 3/2$, you will get partial points.

Solution Task 3

Task 4: Workflow *(12 Points)*

A company has to handle three projects, P_1, P_2, P_3, over the next 4 months. Each project has a *release month*, a *deadline month* and a *work requirement* measured in person-months (one person-month is the amount of work that a single person can do in one month).

Project	release	deadline	work req.
P_1	1	2	11
P_2	2	4	10
P_3	2	3	9

Each month, *8 employees are available*. Employees can work on a project from its release month to its deadline month (each including). Due to the internal structure of the company, *no more than 6 engineers can work* at the same time *on the same project*.

The aim is to find a feasible workforce plan, i.e., to determine for each project and month the number of employees working on it, subject to the above restrictions, such that all projects get completed.

(a) Formalize the above problem as a flow problem. Draw the flow network and state what flow is required to get a feasible solution. *(8 Points)*

(b) Give a solution to the given workflow problem or argue why it is unsolvable. *(4 Points)*

Solution Task 4

Task 5: Randomized Maximum 3-SAT *(26 Points)*

Consider the 3-SAT problem with n clauses over a set of variables $x_1, \ldots, x_k$ with $k \leq 3n$.[1]

Remark: In the following, if you are not able to prove one of the parts, you can always assume the claimed statement of the part to solve the subsequent parts.

(a) Show that a uniformly random assignment satisfies $\frac{7}{8}n$ clauses in expectation. *(5 Points)*

(b) Use (a) to show that for $0 < \varepsilon \leq \frac{1}{2}$ the probability that at least $\frac{7}{8}n(1-\frac{\varepsilon}{7})$ clauses are satisfied is at least $\frac{\varepsilon}{2}$. *(7 Points)*

Hint: $\frac{1}{1+\varepsilon} \leq 1-\frac{\varepsilon}{2}$.

(c) Use (b) for an appropriate value of ε to show that at least $\frac{7}{8}n$ clauses are satisfied with probability at least $\frac{1}{4n}$. *(7 Points)*

Hint: Note that the actual number of satisfied clauses is an integer.

(d) Use (c) to show that there is an algorithm that finds an assignment of variables that satisfies at least $\frac{7}{8}n$ clauses in time $\mathcal{O}(n^2 \log n)$ with high probability. *(7 Points)*

Hint: $\forall m \geq 1 : \ (1 - \frac{1}{m})^m \leq \frac{1}{e}$.

[1] An instance of the n-clause 3-SAT problem over a set of variables $x_1, \ldots, x_k$ is a boolean formula $f(x_1, \ldots, x_k) = C_1 \wedge \ldots \wedge C_n$, where the C_i are *3-clauses*. A *3-clause* C_i is of the form $C_i = l_1 \vee l_2 \vee l_3$ with *literals* $l_1, l_2, l_3 \in \{x_1, \ldots x_k, \overline{x}_1, \ldots \overline{x}_k\}$ ($\overline{x}_j$ denotes the negation of x_j). An assignment $(x_1, \ldots, x_k) \in \{0,1\}^k$ satisfies a clause C_i if C_i evaluates to 1 under the assignment. The goal of the Maximum 3-SAT problem is to satisfy as many clauses as possible.

Solution Task 5