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

Algorithm Theory

24.03.2025, 9:00-11:00

Name: ....

Matriculation No.: ....

Signature: ....

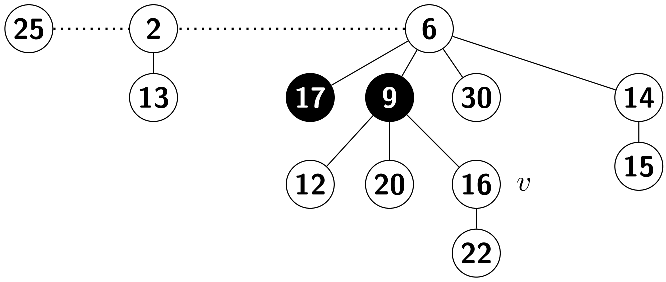
Do not open or turn until told so by the supervisor!

- Write your **name** and **matriculation number** on this page and **sign** the document.
- Your **signature** confirms that you have answered all exam questions yourself without any help, and that you have notified exam supervision of any interference.
- You are allowed to use a summary of **six handwritten A4 pages**.
- **No electronic devices** are allowed.
- 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.
- 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 or if you need additional sheets of paper.
- A total of **45 points** is sufficient to pass and a total of **90 points** is sufficient for the best grade.
- Write your name on **all sheets**!

Task	1	2	3	4	5	6	**Total**
Maximum	35	16	18	16	15	20	120
Points							

Task 1: Short Questions *(35 Points)*

(a) Consider the following Fibonacci heap (black nodes are marked, white nodes are unmarked). How does the given Fibonacci heap look after a `decrease-key(v, 1)` operation and how does it look after a subsequent `delete-min` operation? *(8 Points)*



(b) Given a rooted n-node **tree** $T = (V, E)$ and a weight function on the nodes $w : V \mapsto \mathbb{N}$, your task is to find a subset $S \subseteq V$ of the nodes, such that no two nodes in S are adjacent in T and so that S maximizes the sum $w(S) := \sum_{u \in S} w(u)$. Give an $O(n)$-time algorithm that returns the value $w(S)$ for such an optimal S. *(8 Points)*
Remark: You can assume that each node v stores a pointer to its parent as well as the list C_v of its children.

(c) Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Assume that p families join the dinner and that the i^{th} family has a_i members. Also assume that q tables are available and that the j^{th} table has a seating capacity of b_j. Show how to either find a valid seat assignment or determine that no such assignment exists in time polynomial in p and q. *(7 Points)*

(d) You are given n unique values $a_1 < a_2 < \cdots < a_n$. Your task is to compute:

1. All possible sum values that can be obtained by choosing k elements (with repetition allowed).
2. The number of ways to obtain each possible sum value.

When counting the number of ways to obtain each sum value, the order matters, that is, adding the same numbers in two different orders are two different ways to obtain the same sum. You should solve the above taks in time $O(W \cdot \log W \cdot \log k)$, where $W = k \cdot a_n$ is the maximum achievable sum. You can assume that your algorithm can manipulate exact real numbers and that adding or multiplying two real numbers takes constant time.

Example: Let $n = k = 2$ and $a_1 = 1, a_2 = 2$. The possible sums and their corresponding ways of achieving them are: $1 + 1 = 2, 1 + 2 = 3, 2 + 1 = 3, 2 + 2 = 4$. Thus, the possible sum values are $\{2, 3, 4\}$, with the following counts: 2 appears once, 3 appears twice and 4 appears once. Provide an algorithm that meets these constraints and analyze its complexity. *(12 Points)*
Hint: First try to come up with an algorithm that runs in time $O(W \cdot \log W \cdot k)$ and then try to refine it. You will get partial points for that intermediate step.

Solution Task 1

Task 2: 3-CNF *(16 Points)*

You are given a 3-CNF formula F with n variables and m clauses where each clause consists of exactly 3 literals[1]. We further restrict the formula such that every variable appears at most once in each clause (i.e., there are no clauses of the form $a \vee \neg a \vee b$ that are satisfied under each assignment). The task is to find an assignment of the variables such that as many clauses as possible are satisfied.

a) Let Y be the random variable that counts the number of clauses satisfied by a random assignment of the variables (i.e., setting a variable to `true` with probability 1/2 and `false` otherwise). Show that $E[Y] = \frac{7}{8} \cdot m$. *(4 Points)*

b) Show that $P(Y > \frac{3m}{4}) \geq \frac{1}{2}$. *(8 Points)*
Hint: For a non-negative random variable X, the Markov inequality states that for all $t > 0$ we have

$$P(X \geq t) \leq \frac{E[X]}{t}.$$

c) We say a random assignment of the variables is `good` if more than $3m/4$ of the clauses are satisfied and `bad` otherwise. How often should we repeat the random assignments such that at least one is a `good` assignment with probability at least $1 - 1/n$? *(4 Points)*

[1] An example of a 3-CNF formula with 3 clauses and 4 variables would be $(a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee d) \wedge (a \vee \neg c \vee \neg d)$

Solution Task 2

Task 3: Online *(18 Points)*

Consider the following online problem: Given an unlimited number of boxes where each box has (the same) integer capacity $C > 0$. You get a sequence of N items $x_1, x_2, \dots, x_N$ in online fashion together with a weight function w, that assigns integer weights $\in \{1, 2, \dots, C\}$ to the items. The task is to place all items into the boxes such that the sum of the weights of all items placed into the same box does not exceed the capacity C whereas you want to use as few boxes as possible. Note that you have to place items into boxes directly when you receive them and can not rearrange them later on.
Consider the *Next-Fit* strategy: We always keep a single box open, if the new item does not fit into it, we close the box and open a new one.

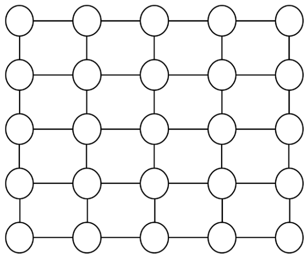
a) Give a sequence of items that shows that *Next-Fit* can not achieve a strict competitive ratio better than $2 - \frac{1}{C}$. *(8 Points)*
Hint: First show that the statement is true for $C = 2$ and then try to generalize it. You will get up to 4 points if you can show this easier approach.

b) Show that *Next-Fit* is 2-competitive. *(10 Points)*

Solution Task 3

Task 4: Approximation Algorithms *(16 Points)*

Suppose you are given a (node-)weighted graph $G = (V, E, w)$ where G has the form of an $n \times n$ grid graph (see figure below). Assume the weights $w(v)$ are non-negative integers. A maximum weighted independent set $S \subseteq V$ is a set of non-adjacent nodes from V such that the $\sum_{v \in S} w(v)$ is as large as possible.



Observe the following greedy algorithm to obtain an approximate solution to the maximum weighted independent set problem.

Algorithm 1 Weighted Independent Set Algorithm

$S \leftarrow \emptyset$
$F \leftarrow (V, E)$
while F is not empty **do**
    Find a vertex $u \in F$ with the largest weight $w(u)$
    $S \leftarrow S \cup \{u\}$
    Remove u and all its neighbors from F (i.e., all vertices v with an edge (u, v))
    Remove all edges ending at any of these deleted vertices from F
return S

(a) Show that even in a 2×2 grid graph and unique node-weights (i.e., no two nodes have the same weight), the greedy algorithm is not optimal. *(4 Points)*

(b) Prove that the greedy algorithm computes a 4-approximation for the maximum weighted independent set problem in grid graphs. *(12 Points)*
Remark: You will get up to 5 points if you can show a 5-approximation instead.

Solution Task 4

Task 5: Dynammic Programming *(15 Points)*

Let N be a positive integer representing bus stops numbered from 1 to N. A set of bus lines is given as intervals (s_i, t_i) for $i = 1, \dots, m$ (with $s_i, t_i \in \mathbb{N}$), where each bus line stops at every integer point in the interval $[s_i, t_i]$.

Find a subset of these bus lines such that every stop in $1, 2, \dots, N$ is covered by exactly one selected bus line. Determine whether such a subset exists. Assume that for each busline i it holds $1 \leq s_i \leq t_i \leq N$. Try to find an algorithm that is as efficient as possible and state its asymptotic running time in terms of N and m.

Solution Task 5

Task 6: Weightlifting *(20 Points)*

At the Sportprinz Fitnessclub Freiburg West, there are n female participants and n male participants. Each participant can lift a certain amount of weights (in kg). For the first general workout session, the goal of the coach is to form groups of two, where each group has to consist of one male and one female participant, such that: Each participant is in a group and the overall sum of absolute differences between the weights, the two participants of the group can lift, is *as small as possible.*[2]

(a) Give a polynomial time algorithm that achieves the goal of the coach. Argue correctness and state the run time. *(7 Points)*

For the second general workout session, the coach wants to achieve the same goal as above but this time the $2n$ participants have to be divided into arbitrary pairs, irrespective of gender.

(b) Give an algorithm that helps the coach in achieving the new goal in $O(n \log n)$ time. Argue correctness and runtime.
Hint: Try to solve the same problem in the case where each participant can lift a unique weight from $\{1, 2, 3, ..., 2n\}$ *kg.* *(13 Points)*

[2] More formal: Find a set of pairs P, where each participant is part of exactly one pair, and such that P minimizes $\sum_{(p,p') \in P} |w(p) - w(p')|$.

Solution Task 6