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

Algorithm Theory

3rd of September 2021, 9:00 - 11:00

Name: .....

Matriculation No.: .....

Signature: .....

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

- Fill out the **Corona Ordinance form**.
- 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 **five handwritten, single-sided 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**!

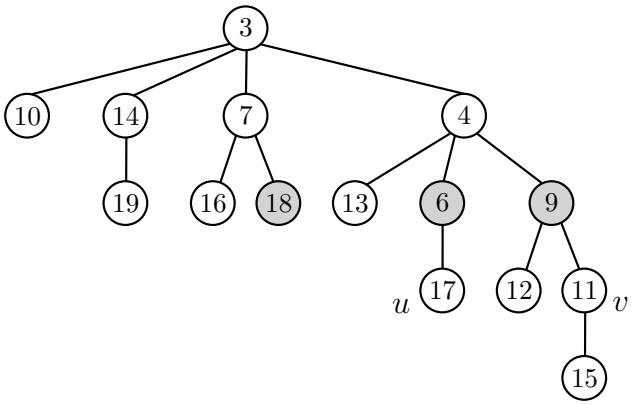
Task	1	2	3	4	5	**Total**
Maximum	42	16	22	18	22	120
Points						

Task 1: Short Questions *(42 Points)*

(a) We are given an undirected graph $G = (V, E)$ where each node $v \in V$ represents an inhabitant of Freiburg and $\{u, v\} \in E$ if and only if u and v have direct contact with each other and $n := |V|, m := |E|$. Imagine that person $x \in V$ is vulnerable, but cannot be vaccinated against an ongoing pandemic. Also imagine that person $y \in V$ that does not have direct contact with x is already infected and cannot be vaccinated anymore. To protect x, we can vaccinate other people (not x, y) who then will not pass on the virus to their contacts (i.e., neighbors in G).

Describe an efficient algorithm to compute the *minimum number* of people who need to be vaccinated in order to interrupt all possible infection chains from y to x. Give the running time. *(6 Points)*

(b) Consider the following Fibonacci heap with marked nodes shown in gray and two dedicated nodes u, v. Give the state of the Fibonacci heap after conducting the operation `Decrease-Key`$(v, 8)$. Then conduct `Decrease-Key`$(u, 5)$ on the resulting Fibonacci heap and give the state of it. *(8 Points)*



(c) Let T be a rooted tree with n nodes. Describe an algorithm that computes a *minimum* vertex cover of T with a running time that is *polynomial* in n. Explain the running time. *(10 Points)*

Remark: A vertex cover is a set of nodes that contains at least one endpoint of every edge.

(d) Let $G = (V, E)$ be an undirected graph with maximum degree Δ. An independent set of a graph G is a subset $S \subseteq V$ of nodes such that for all $x, y \in S$ we have $\{x, y\} \notin E$, i.e., each two nodes in S are non-adjacent. Consider the following algorithm. Select any vertex, add it to the independent set S, then delete this vertex and all of its neighbors from the graph. Repeat this process for the remaining subgraph until the subgraph becomes empty. Show that the algorithm outputs an independent set S that is a $\frac{1}{\Delta}$-approximation of an independent set of maximum size. *(10 Points)*

Remark: You get partial points if you show that S is a $\frac{1}{\Delta+1}$ - approximation.

(e) In the weighted matching problem, we are given an undirected weighted graph $G = (V, E, c)$ with weight function $c : E \to \mathbb{R}$. The weight of a matching $M \subseteq E$ is defined as $c(M) = \sum_{e \in M} c(e)$. The goal is to find a matching of maximum weight.

We consider the weighted matching problem as an online problem. The edges of the graph are presented in an online fashion and we have to maintain a matching M of G. After each arrival of an edge e, we must decide whether or not to add e to M. Show that there is no strictly c-competitive online algorithm for weighted matching for any $c \geq 1$. *(8 Points)*

Remark: For a maximization problem, an online algorithm ALG is strictly c-competitive if for any input I we have $ALG(I) \geq \frac{1}{c} OPT(I)$.

Solution Task 1

Task 2: Graph Algorithms *(16 Points)*

Suppose there is a breakout of a pandemic and you have been asked to help find solutions to stop it. You are given a map of the country, with coordinates of all cities and the roads between them. There are X infected cities, Y uninfected cities and R roads. Each of the R roads directly connects two cities (but not all pairs of cities necessarily have a direct connection). One of the cities is the capital. A city is reachable from the capital if you can follow one or more roads from the capital to that city.

To start the vaccination process in the uninfected cities quickly, 10 doctors must be stationed in each uninfected city. There are arbitrarily many doctors available, but all are located in the capital. They can travel along the roads through uninfected as well as infected cities. However, in order to limit disease risk to doctors, each *infected* city has the restriction that at most 50 doctors can travel through it. Describe an algorithm to decide whether we can send 10 doctors to each uninfected city. Analyze the running time of your algorithm in terms of X, Y, and R.

Solution Task 2

Task 3: Online Vertex Cover - Amortized Analysis *(22 Points)*

We consider an online version of the minimum vertex cover problem. The edges of a graph arrive in an online fashion and we have to maintain a vertex cover S of the graph defined by the edges that have arrived so far. After each arrival of an edge, we can decide to add nodes to S. Consider the following algorithm: We start with $S = \emptyset$ and then do the following.

Algorithm 1

For each incoming edge $\{u, v\}$, if $u \notin S$ and $v \notin S$, add either u or v to S (chosen arbitrarily).

(a) How bad can our computed vertex cover S be compared to a minimum vertex cover S^*? Explain your answer. *(6 Points)*

In order to avoid that S becomes too large, we do the following: Every k steps (i.e., after the arrival of k edges), we compute a minimum vertex cover S^* of the graph given so far, remove all $v \in S \setminus S^*$ from S and add all $v \in S^* \setminus S$ to S. In all other steps we proceed as in Algorithm 1.

The cost of one step is defined as the *number of nodes* which are *removed* from or *inserted* to S in that step (in particular, the time to find a minimum vertex cover is not taken into account).

(b) What is the amortized cost of each step as a function of n (number of nodes) and k? Explain your answer. *(6 Points)*

Now consider the following procedure: In each step, if S is twice as large as a minimum vertex cover S^*, we replace S by S^*. Otherwise we proceed as in Algorithm 1.

(c) Show that the amortized cost of each step is constant. *(10 Points)*

Solution Task 3

Task 4: Computing Water Amounts *(18 Points)*

Given a sequence of integers $H = [h_1 \cdots h_n]$, where each h_i represents the hight of vertical bar number i. Each bar i has dimensions $1 \times h_i$. For example, consider a sequence:

$$H = [60, 40, 80, 20, 30, 50, 40, 30, 40, 20]$$

Imagine that someone pours water on these bars which is then captured as (2-dimensional) lakes between bars as shown below (Figure 1). Your task is to compute the maximum volume of water (which we define as the total area of all blue colored areas in the picture below) that will be kept between the bars.

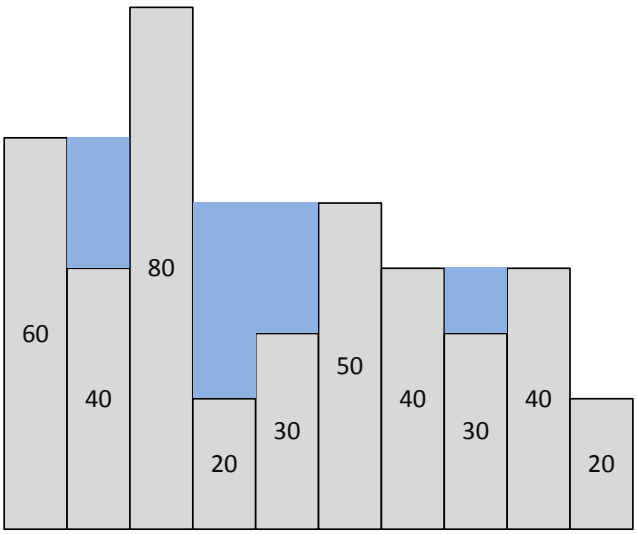


Figure 1: Water volumes kept by bars.

Describe an algorithm with runtime $O(n \log n)$ to find the maximum volume of water kept between the bars, defined by a input sequence H. Explain the runtime.

Hint: There are different approaches that work, e.g., Divide and Conquer.

Solution Task 4

Task 5: Randomized Coloring *(22 Points)*

Let $G = (V, E)$ be a simple, undirected graph with maximum degree Δ. A vertex coloring of a graph is an assignment of colors to the vertices such that adjacent vertices have different colors. More formally, a coloring ϕ is a mapping $\phi : V \to C$ from V to a color space C such that $\phi(u) \neq \phi(v)$ if $\{u, v\} \in E$.

Consider the following randomized algorithm to compute a coloring of G with 2Δ colors, i.e., a coloring $\phi : V \to \{1, \ldots, 2\Delta\}$.
Each uncolored node v assigns itself a tentative color $c_v \in \{1, \ldots, 2\Delta\}$ uniformly at random. If v has no neighbor with the same (tentative or permanent) color, it keeps c_v permanently. Otherwise it uncolors itself again. Repeat until all nodes are colored. In pseudocode:

Algorithm 2

```
1: for v ∈ V do
2:     φ(v) = ⊥                                   ▷ each node is initially uncolored
3: while there is a v with φ(v) = ⊥ do
4:     for each u with φ(u) = ⊥ do
5:         Choose c_u ∈ {1, ..., 2Δ} uniformly at random
6:     for each u with φ(u) = ⊥ do
7:         if u has no neighbor w with c_u = c_w or c_u = φ(w) then
8:             φ(u) := c_u
```

We call one run of the while-loop in line 3 a *round*.

(a) Show that for each round and each uncolored node u, the probability that the condition in line 7 is true (i.e., u permanently chooses a color) is at least $1/2$. *(8 Points)*

(b) Show that in each round, in expectation, the number of uncolored nodes is at least halved. *(5 Points)*

Hint: Use part (a).

(c) Show that Algorithm 2 terminates in $2 \cdot \log n$ rounds with probability at least $1 - \frac{1}{n}$. *(9 Points)*

Hint: Use part (a).

Solution Task 5