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

Algorithm Theory

22nd of March 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	24	15	24	15	120
Points						

Task 1: Short Questions *(42 Points)*

(a) You are given a *strictly monotonically decreasing* function $f : \mathbb{N} \to \mathbb{R}$ with $f(1) > 0$. You know $f(n) < 0$ for some *unknown* value $n \in \mathbb{N}$. The goal is to compute the number $i \in \mathbb{N}$ such that $f(i) < 0$ but $f(i-1) \geq 0$, i.e. you want to know where f first becomes negative. Give an algorithm that computes this number in running time $\mathcal{O}(\log n)$. Briefly explain the running time. *(6 Points)*

(b) Let $G = (V, E)$ be a directed graph with the property that for each node v, the number of edges going into v is equal to the number of edges going out of v. Let s and t be two nodes of G such that there are k pairwise edge-disjoint paths from s to t. Show that there are also k pairwise edge-disjoint paths from t to s. *(8 Points)*

(c) Suppose we "simplify" Fibonacci heaps such that we do *not* mark any nodes that have lost a child and consequentially also do *not* cut marked parents of a node that needs to be cut out due to a `decrease-key`-operation. Is the *amortized* running time

(i) ... of the `decrease-key`-operation still $\mathcal{O}(1)$?

(ii) ... of the `delete-min`-operation still $\mathcal{O}(\log n)$?

Explain your answers. *(6 Points)*

(d) Let $G = (V, E)$ be a directed graph with source s and sink t and integral capacities. Let $(S, V \setminus S)$ be a minimum (s, t)-cut. Give a polynomial time algorithm to determine whether $(S, V \setminus S)$ is the unique minimum (s, t)-cut, i.e., whether it has a capacity strictly less than all other (s, t)-cuts. Explain the correctness of your algorithm. *(8 Points)*

(e) Assume you are given a randomized algorithm $\mathcal{A}$ that given a graph G with n nodes and a maximum matching of size s, computes an integer $k \leq s$ in time $T(n)$ such that with probability at least $p(n)$ we have $k = s$.

Give an algorithm with running time $\mathcal{O}\big(p(n)^{-1} \cdot T(n) \cdot \log n\big)$ that computes the size of a maximum matching of a graph with n nodes with probability at least $1 - \frac{1}{n}$. Prove the success probability. *(7 Points)*

(f) Consider a naive greedy algorithm `greedy-flow` for the maximum flow problem:

As long as possible, find an s-t path with free capacity and add as much flow as possible to the path.

In particular, `greedy-flow` never reduces the flow through an edge, i.e., it makes no use of backward edges. We saw in the lecture that `greedy-flow` does not always compute a maximum flow.

Show that `greedy-flow` can perform arbitrarily bad. That is, for any $\alpha > 1$, there is a flow network such that the value of the maximum flow is more than α times the value of the flow computed by some execution of `greedy-flow`. *(7 Points)*

Solution Task 1

Task 2: Subset Sum *(24 Points)*

Let $S = \{s_1, \ldots, s_n\}$ be a set of *positive integers*. Given a *target value* $T \in \mathbb{N}_{\geq 1}$ we want to answer the question if there is a subset $S' \subseteq S$ that sums up to T, i.e., $\sum_{s \in S'} s = T$. The answer should either be *true* or *false*.

(a) Give an algorithm that solves this task with running time $\mathcal{O}(n \cdot T)$. Explain the running time. *(8 Points)*

(b) Let $\varepsilon > 0$ and let $\hat{s}_i := \lceil s_i \cdot \alpha \rceil$ for some *scaling factor* α. Let $\hat{S} := \{\hat{s}_1, \ldots, \hat{s}_n\}$. Consider the case that there exists a subset $S' \subseteq S$ with $\sum_{s_i \in S'} s_i = T$. Express α with variables T, n, ε such that *in this case* there is a subset $\hat{S}' \subseteq \hat{S}$ such that the following holds

$$\frac{n}{\varepsilon} \leq \sum_{\hat{s}_i \in \hat{S}'} \hat{s}_i \leq (1+\varepsilon)\frac{n}{\varepsilon}. \tag{1}$$

Prove Inequality (1) for that value α. *(6 Points)*

(c) Now consider the case that there exists a subset $\hat{S}' \subseteq \hat{S}$ that fulfills Inequality (1) (for α and $\hat{S} := \{\hat{s}_1, \ldots, \hat{s}_n\}$ defined in part (b)). Prove that *in this case* there is also a subset $S' \subseteq S$ with $(1-\varepsilon)T \leq \sum_{s \in S'} s \leq (1+\varepsilon)T$. *(6 Points)*

(d) Give an algorithm that has running time polynomial in n/ε that outputs *true* if there exists a subset $S' \subseteq S$ with $\sum_{s \in S'} s = T$ and false if there *does not exist* any subset $S' \subseteq S$ with $(1-\varepsilon)T \leq \sum_{s \in S'} s \leq (1+\varepsilon)T$. In other cases the output can be arbitrary. Briefly explain correctness and running time. *(4 Points)*

Hint: You can use the results of (a),(b),(c) even if you did not succeed there.

Solution Task 2

Task 3: Graph Connectivity *(15 Points)*

Recall that the vertex connectivity of a graph G is the maximum integer κ such that G is κ-vertex-connected. Similarly, the edge connectivity of a graph G is the maximum integer λ such that G is λ-edge-connected.

(a) Let $G = (V, E)$ be an *undirected, unweighted* graph with *vertex* connectivity κ. Define G' as a graph G where we add a new node $v \notin V$ that is connected to κ nodes in G. Prove that G' also has vertex connectivity κ. *(10 Points)*

(b) Give an *undirected, unweighted* graph with *vertex* connectivity κ and *edge* connectivity λ such that $\kappa < \lambda$. *(5 Points)*

Solution Task 3

Task 4: Greedy Vertex Cover Algorithms *(24 Points)*

Let $G = (V, E)$ be an *undirected, unweighted* graph. Consider the following algorithms that give approximate solutions to the minimum vertex cover problem.

Algorithm 1 `alg1`

```
1: S ← ∅                                  ▷ create an empty set
2: while E ≠ ∅ do
3:     pick some edge {u, v} ∈ E
4:     S ← S ∪ {u, v}
5:     remove edges incident to u or v from E
6: return S
```

Algorithm 2 `alg2`

```
1: S ← ∅                                  ▷ create an empty set
2: while E ≠ ∅ do
3:     pick vertex v ∈ V of maximal degree
4:     S ← S ∪ {v}
5:     remove edges incident to v from E
6: return S
```

(a) Show that `alg1` and `alg2` output valid vertex covers. *(4 Points)*

(b) Argue why `alg1` provides a 2-approximation of the minimum vertex cover problem. *(3 Points)*

Hint: You can use results that we proved in the lecture.

(c) Argue why `alg2` provides a $O(\log n)$-approximation of the minimum vertex cover problem. *(5 Points)*

Hint: You can use results that we proved in the lecture.

(d) Show that the solution provided by `alg2` is only a $\Theta(\log n)$ approximation for some graphs. *(12 Points)*

Hint: Give a bipartite graph with node set $V = L \cup R$ and $|L| = k$ and $|R| = \Theta(k \log k)$, where `alg2` outputs R but the best solution would be L.

Solution Task 4

Task 5: Randomized Algorithms *(15 Points)*

Consider the Find-bill problem which is stated as follows. There are n boxes numbered from 1 to n and exactly one box contains a dollar bill. The other boxes are empty. A probe is defined as opening a box to see if it contains the dollar bill. The objective is to locate the box containing the dollar bill while minimizing the number of probes performed. Consider the following randomized algorithm:

Algorithm 3 `rand`

1: Select $x \in \{\textit{heads}, \textit{tail}\}$ uniformly at random
2: **if** $x = \textit{heads}$ **then**
3: Probe boxes in order $1, \ldots, n$ and stop if bill is located
4: **else**
5: Probe boxes in order $n, \ldots, 1$ and stop if bill is located

(a) Show that for any input (that is, for any position of the bill), the expected number of probes that `rand` does is $\frac{n+1}{2}$. *(5 Points)*

(b) Show that there is no better randomized algorithm than `rand`. That is, for any randomized algorithm $\mathcal{A}$ for the Find-bill problem, there is an input such that the expected number of probes that $\mathcal{A}$ does on this input is at least $\frac{n+1}{2}$.

Hint: Use Yao's Principle. *(10 Points)*

Solution Task 5