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

Algorithm Theory

Monday, May 4 2020, 09:00-11:00

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 (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	6	**Total**
Maximum	30	18	12	15	25	20	120
Points							

Task 1: Short Questions *(30 Points)*

(a) *(8 Points)* Imagine a long, straight country road with houses scattered sparsely along it (we can picture the road as a line segment with an eastern endpoint and a western endpoint). Our aim is to place cell phone base stations at certain points along the road, so that every house is within 4 miles of one of the base stations.

Give an efficient algorithm that achieves this goal, using a minimum number of base stations. Prove that your algorithm is optimal, i.e., one cannot cover all houses with less stations.

(b) *(7 Points)* Suppose that you wish to find, among all minimum s-t cuts in a flow network G with non-negative integer capacities, one with the smallest number of edges. How can we modify the capacities of G to create a new flow network G' such that any minimum s-t cut in G' is a minimum cut with the smallest number of edges in G? Explain your answer.

(c) *(7 Points)* Imagine a 2-dimensional grid modeled by $\mathbb{Z} \times \mathbb{Z}$. You stand at point $(0, 0)$ and somewhere on the grid there is a treasure that you would like to find. You can only see the treasure if you are standing on its exact coordinates. In each time step, you can move one step on the grid in either of the four directions, i.e., if you are standing at (m, n), you can move to either $(m + 1, n)$, $(m - 1, n)$, $(m, n + 1)$ or $(m, n - 1)$.

Let OPT be the minimum number of steps needed to go from $(0, 0)$ to the treasure. We define the competitive ratio of an algorithm that finds the treasure within ALG steps as $\frac{\text{ALG}}{\text{OPT}}$.

(i) Describe a deterministic algorithm with a competitive ratio of $O(\text{OPT})$ (a picture can be a sufficient description of the algorithm). Prove the competitive ratio.

(ii) Prove that there is no deterministic algorithm with a competitive ratio smaller than OPT.

(d) *(8 Points)* Assume that we are given an integer array A of size n and let $k := \lceil n/10 \rceil$. The goal is to compute an integer array B of length n such that

$$\forall i \in \{0, \ldots, n-1\} \ : \ B[i] = \sum_{j=\max\{0,i-k\}}^{\min\{n-1,i+k\}} A[j].$$

Assume further that B has to be computed on an EREW PRAM with p processors. Describe an efficient parallel algorithm to compute B and give the asymptotic running time of your algorithm as a function of n and p.

Solution Task 1

Task 2: Data Structures and Amortized Analysis *(18 Points)*

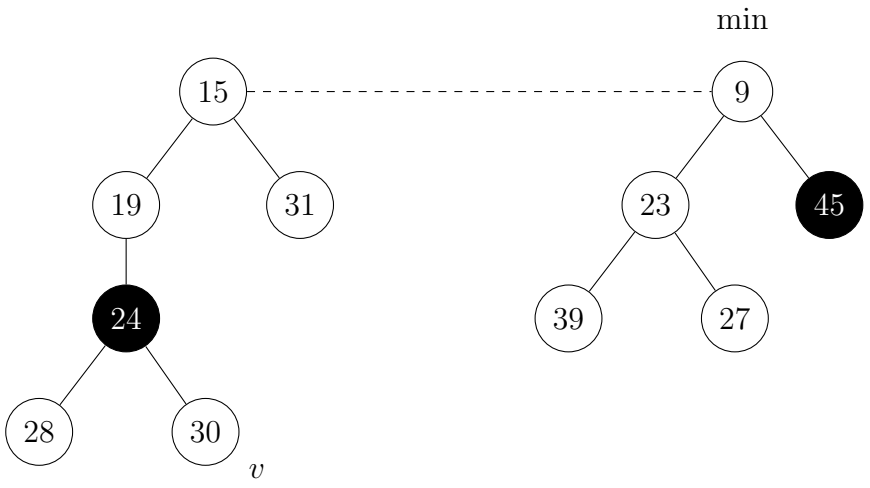
(a) *(10 Points)* Consider a binary min-heap data structure that supports the two operations `insert` and `delete-min`. The heap is initially empty and we assume that its number of elements never exceeds n.

(i) Use the potential function method to show that we can consider the amortized cost of `insert` to be $O(\log n)$ and the amortized cost of `delete-min` to be $O(1)$.

(ii) We would like to amortize the costs differently such that the amortized cost of `insert` is $O(1)$ and the amortized cost of `delete-min` is $O(\log n)$. Either define a feasible potential function that yields these amortized costs or argue why this is not possible.

You get partial points if you use the accounting method instead of a potential function.

(b) *(8 Points)* Consider the following Fibonacci heap (black nodes are *marked*, white nodes are *unmarked*). How does the given Fibonacci heap look after after a **decrease-key**$(v, 18)$ operation and how does it look after a subsequent **delete-min** operation?



Solution Task 2

Task 3: Bin Covering *(12 Points)*

The bin covering problem is the following: We are given n items with weights $w_1, \ldots, w_n \in (0, 1)$ and an unlimited number of bins with unlimited capacities. We want to assign the items to the bins such that we maximize the number of bins with total weight at least 1.

Assume the items arrive in an online fashion, i.e., we have to assign each item to a bin immediately after the item arrives without knowing the number of future items or their weights.

Give a deterministic algorithm for this problem that has the optimal strict competitive ratio. You should prove the competitive ratio of your algorithm and also show that no deterministic algorithm for this problem has a better strict competitive ratio.

Solution Task 3

Task 4: Work Schedule *(15 Points)*

Assume you want to design a work schedule for a hospital for the next n days. The hospital employs k doctors. On day i, exactly p_i doctors need to be present, for $i = 1, \ldots, n$. Each doctor j provides a set $L_j \subseteq \{1, \ldots, n\}$ of days on which he or she is willing to work.

(a) *(9 Points)* Describe a polynomial-time algorithm that either

- returns a list $L'_j \subseteq L_j$ of working days for each doctor j such that on day i, exactly p_i doctors are present or
- reports that there is no such set of lists that fulfills the given constraints.

(b) *(6 Points)* The hospital finds that the doctors tend to submit lists that are much too restrictive, and so it often happens that there is no feasible working schedule. Thus, the hospital relaxes the requirements in the following way. For some number $c > 0$, each doctor j can be forced to work on c days which are not in his/her list L_j.

Give a polynomial-time algorithm to solve this problem, i.e., the algorithm should either

- return a list L'_j of working days for each doctor j with $|L'_j \setminus L_j| \leq c$ such that on day i, exactly p_i doctors are present or
- report that there is no such set of lists that fulfills the given constraints.

Solution Task 4

Task 5: Randomization *(25 Points)*

Given a set S of n pairwise distinct numbers and a number $k \in \{1, \ldots, n\}$, we want to define a function that returns the k^{th} largest element of S.

To this end, the following randomized divide and conquer algorithm is given:

Algorithm 1 `select`(S, k)

```
1: Choose a pivot x ∈ S uniformly at random
2: S⁻ = ∅
3: S⁺ = ∅
4: for all y ∈ S \ {x} do
5:     if y < x then
6:         S⁻ = S⁻ ∪ {y}
7:     else
8:         S⁺ = S⁺ ∪ {y}
9: if |S⁻| = k − 1 then
10:    return x
11: else if |S⁻| ≥ k then
12:    return select(S⁻, k)
13: else if |S⁻| < k − 1 then
14:    return select(S⁺, k − |S⁻| − 1)
```

(a) *(4 Points)* Shortly explain why `select`(S, k) is correct, i.e., returns the k^{th} largest element of S, and explain the worst-case runtime.

(b) *(3 Points)* What is the probability that both $|S^+| \leq \frac{3}{4}|S|$ and $|S^-| \leq \frac{3}{4}|S|$ (after one iteration)? Explain your answer. For simplicity, you may assume that $|S|$ is a multiple of 4.

(c) *(6 Points)* Give an upper bound on the expected time until S shrinks at least by a factor $\frac{3}{4}$, i.e., until `select`(S, k) makes a recursive call on a set of size at most $\frac{3}{4}|S|$.

Comment: With time we mean the number of basic steps and not just the number of recursive calls. Giving the expected time asymptotically (in O-notation) is sufficient.

Hint: Use part (b)

(d) *(6 Points)* Give an upper bound on the expected runtime of `select`(S, k). Explain your answer.

Hint: Use part (c)

(e) *(6 Points)* Show that `select`(S, k) terminates in time $O(n \log n)$ with probability at least $1 - \frac{1}{n}$.

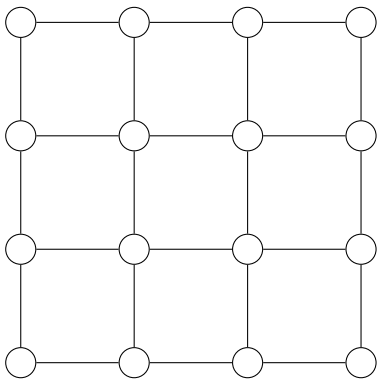
Hint: Use part (b) and Chernoff's Bound: If $X_1, \ldots, X_n$ is a sequence of independent 0-1 random variables, $X = \sum X_i$ and $\mu = E[X]$, then for any $0 < \delta < 1$ we have

$$\Pr(X \leq (1 - \delta)E[X]) \leq e^{-\frac{\delta^2}{2}E[X]} .$$

Solution Task 5

Task 6: Maximum Independent Set Approximation *(20 Points)*

An *independent set* of a graph $G = (V, E)$ is a subset $I \subseteq V$ of nodes such that for all $x, y \in I$ we have $\{x, y\} \notin E$, i.e., each two nodes in I are non-adjacent. A *maximum independent set* is an independent set of maximum size. We consider the problem of finding a maximum independent set in $n \times n$-node *grid graphs* like the following:



(a) *(6 Points)* Let S be a maximum independent set of a given grid graph G. Further assume that we compute an independent set I of G by the following simple greedy algorithm. We start with $I = \emptyset$. We then iterate through the nodes of G in an arbitrary order. When processing node v, we add v to I if no neighbor of v has already been added to I. Show that $|I| \geq \frac{3}{10} \cdot |S|$.

Hint: Find an upper bound for $|S|$ and a lower bound for $|I|$, both in terms of n. Note that the ratio $3/10$ is not tight, one could even show that $|I| > \frac{2}{5} \cdot |S|$.

Now we consider a weighted version of the above problem. Given a grid graph $G = (V, E)$, assume that each node v has a weight $w(v) \in \mathbb{N}$. The weight of a set $S \subseteq V$ is defined as $w(S) = \sum_{v \in S} w(v)$. The goal is to find an independent set of maximum weight.

Consider the following greedy algorithm for this problem:

Algorithm 2 `heaviest-first`

$I = \emptyset$
while G is not empty **do**
  Pick a node v of maximum weight.
  Add v to I.
  Delete v and v's neighbors from G.
return I

(b) *(3 Points)* Let I be the independent set returned by `heaviest-first`. Show that for each node $v \in V$, either $v \in I$ or v is adjacent to a node $v' \in I$ with $w(v) \leq w(v')$.

(c) *(6 Points)* Let S be an independent set with maximum weight. Show that `heaviest-first` computes an independent set I with $w(I) \geq w(S)/4$.

Hint: Iterate through the nodes of S and use part (b).

(d) *(5 Points)* Show that this approximation ratio of `heaviest-first` is tight, i.e., for any constant $c > 1/4$, there is a graph on which the `heaviest-first` algorithm computes a solution I with $w(I) < c \cdot w(S)$, where S is a maximum weight independent set.

Solution Task 6