Surname:
First Name:
Matriculation No.:

Karlsruhe Institute of Technology
Institute for Theoretical Informatics

Prof. Dr. Marvin Künnemann

12.03.2026

Exam Algorithms II

Problem 1.	Miscellaneous Tasks	12 Points
Problem 2.	Max-Flow: You shall not pass (more than once)!	10 Points
Problem 3.	Shortest Paths: Pairwise Shortest Paths	9 Points
Problem 4.	Randomized: Sibling Sabotage	9 Points
Problem 5.	Geometric Algorithms: Asteroids	9 Points
Problem 6.	Stringology: Evenly Spaced Patterns	11 Points

Please note:

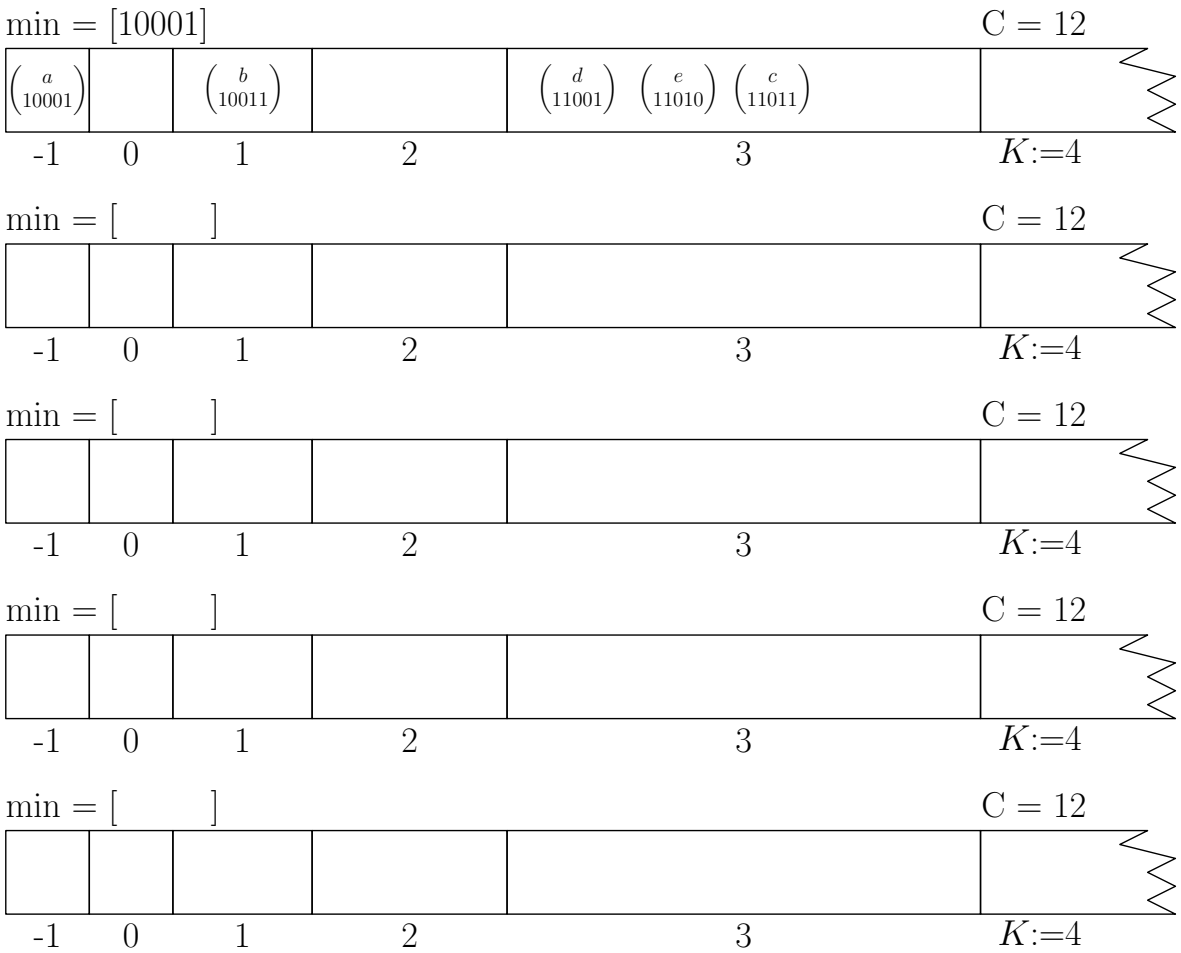
- The only allowed aid is **one** A4 sheet with your **handwritten** notes.
- Write your answers in blue or black ink only, using an indelible pen.
- **Write** your **Matriculation No.** on **all** sheets of the exam and additional pages.
- The duration of the exam is **120 minutes**.
- You may write on the backside of the sheets. If you do so, please reference the exercise that you answer.
- The exam contains **15 pages**.

Problem	**1**	**2**	**3**	**4**	**5**	**6**	**Total**
max. Points	12	10	9	9	9	11	60
Achieved							
						Grade:	

Problem 1. Miscellaneous Tasks **(_ /12 P.)**

a. Given the Radix Heap below with $C = 12$ and $K = 4$, perform the following operations on it. Draw the Radix Heap after every operation. Do not forget to state the value of min (denoted d^* in the lecture). **(_ /4 P.)**
Note: It suffices to give the keys (a-f) without the numerical values when filling in the buckets.

- `deleteMin()`
- `insert(f,11111)`
- `deleteMin()`
- `deleteMin()`



b. Consider the following running times for input size n and parameter $k \in \mathbb{N}$. State whether they are fixed-parameter tractable and justify your answer. **(_ /2 P.)**

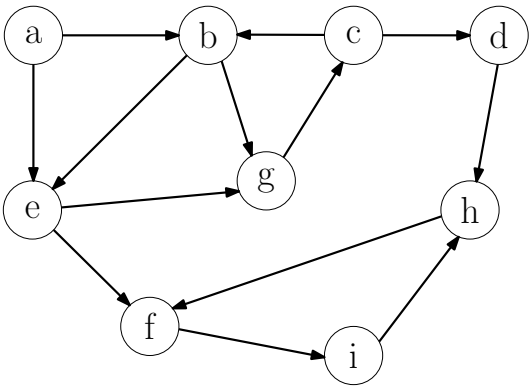
1. $\left(2^{k \log k}\right)^{\log^2 n}$
2. $k^k \cdot n^{\frac{1}{\sqrt{k}}}$

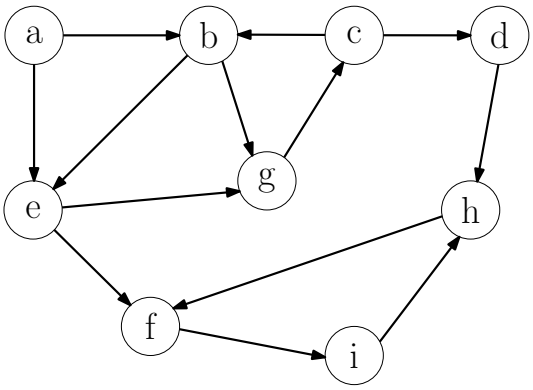
c. Consider approximation algorithms with the following running times for input size n and approximation factor $(1+\varepsilon)$ for some minimization problems. State whether they are a PTAS, FPTAS or neither and justify your answer. **(_ /2 P.)**

1. $n^{\frac{1}{\varepsilon}} \cdot \log \frac{1}{\varepsilon}$
2. $\left(n + \frac{1}{\varepsilon}\right)^2 \cdot \log(\frac{n}{\varepsilon})$

d. Mark all strongly connected components in the given graph and draw the shrunken graph. **(_ /2 P.)**

Two copies are provided below. Clearly mark the one that should be graded if you use both copies. Otherwise, we grade this part with 0 points.





e. Consider an operator $\star : \mathbb{N}^2 \to \mathbb{N}$ defined as $x \star y := x + y + xy$. **(_ /2 P.)**
Let $a_1, \ldots, a_n$ be integers. Show that one can compute $(\ldots((a_1 \star a_2) \star a_3) \star \cdots \star a_n)$ in time $\mathcal{O}(n/p + \log p)$ using p parallel processors.

Problem 2. Max-Flow: You shall not pass (more than once)! **(_ /10 P.)**

In a game show called "You shall not pass (more than once)!", k participants are placed in an $n \times n$ grid-like maze of rooms connected by doors. To win, the participants must reach an *exit*, defined as any room located on the boundary of the maze. The maze features a unique restriction: each *door* can only be passed through **once** in total. Once any participant traverses a door, it is locked and becomes impassable for everyone else.
Formally, we represent the maze as an $n \times n$ undirected (incomplete) grid graph $G = (V, E)$, where V is the set of rooms and E is the set of available doors.

Rooms (V): Each room is identified by its coordinates (x, y), s.t. $V = \{v_{(x,y)} \mid 1 \leq x, y \leq n\}$

Doors (E): An edge $\{v_{(x_1,y_1)}, v_{(x_2,y_2)}\}$ exists if the corresponding door is open.

The rooms have to be adjacent in the grid i.e., $|x_1 - x_2| + |y_1 - y_2| = 1$.

Exits (B): The exit boundary $B \subseteq V$ consists of all rooms on the edges of the grid:

$$B = \{v_{(x,y)} \in V \mid x \in \{1, n\} \text{ or } y \in \{1, n\}\}$$

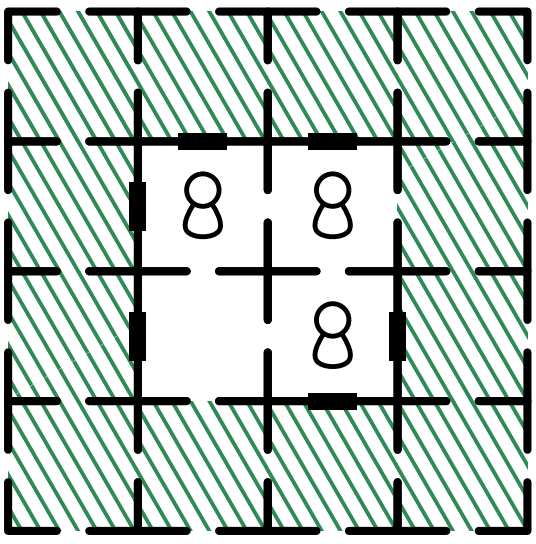
Starting Positions: There are k participants, each starting in a distinct room $s_1, \ldots, s_k \in V$.

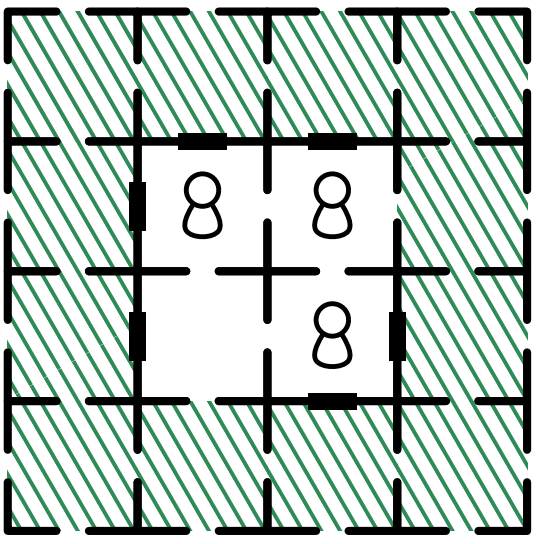
As an avid fan of the show, you are interested in determining for any instance, given as the $n \times n$ maze $G = (V, E)$ and the k starting positions $s_1, \ldots, s_k$, the maximum number of participants that can reach the exit at the same time.

Note: in the following, $T_{maxflow}(n, m)$ denotes the optimal time needed to compute the maximum flow in a network with n vertices and m edges.

a. Determine for the maze below with 3 participants how many can escape at the same time. It suffices to draw the paths that the participants can take to reach the exit. The black rectangles denote closed doors; rooms on the boundary are shaded. **(_ /1 P.)**

There are two copies provided. Clearly mark the one that should be graded if you use both copies. Otherwise, we grade this part with 0 points.





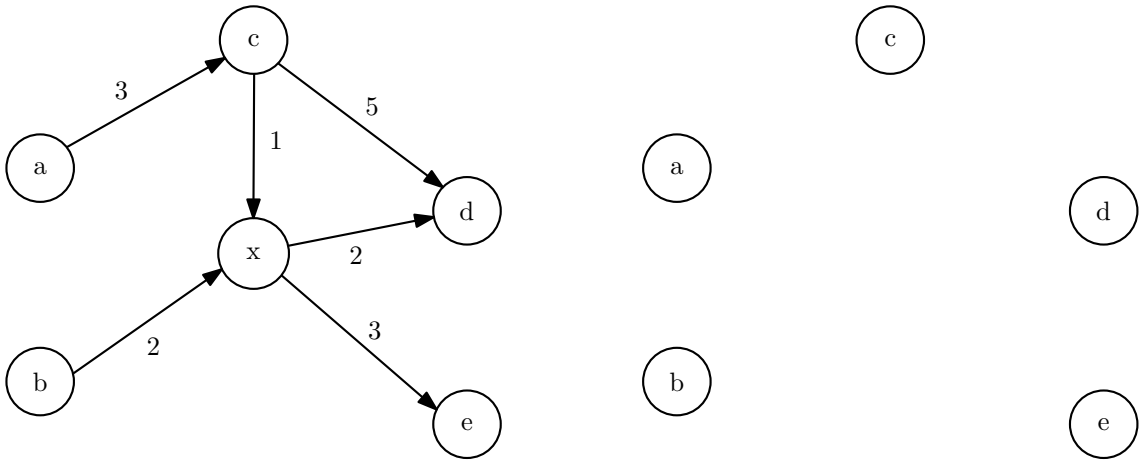
b. While you really like puzzling out how many participants can escape, your inner computer scientist wants to solve this problem algorithmically. **(_ /5 P.)**
Design an $\mathcal{O}\big(T_{\text{maxflow}}(n^2, m)\big)$-time algorithm that determines the maximum number of participants that can reach the exit for a given instance as described above. Prove the correctness (specifically, why your formulation as a max-flow problem models the problem correctly) and analyze the runtime of your algorithm.

c. After you posted your very successful algorithm online, the shows' network is quite unhappy: people are no longer interested in the game as your algorithm spoils the answer too easily! To keep it interesting, they devise a new rule: As soon as a player leaves a room, all doors of that room are locked. **(_ /4 P.)**
Design an algorithm that determines the maximum number of participants that can reach the exit with the new ruleset. Again, your algorithm should run in time $\mathcal{O}\big(T_{\text{maxflow}}(n^2, m)\big)$ with the same input. Prove the correctness and analyze the runtime of your algorithm.

Problem 3. Shortest Paths: Pairwise Shortest Paths **(_ /9 P.)**

In the following we will work with an edge-weighted, directed graph $G = (V, E, c)$ with n vertices, m edges and positive edge weights $c : E \to \mathbb{N}$. We denote by $d_G(u, v)$ the length of the shortest path in G between vertices u and v.

a. Consider the edge-weighted graph given below. Remove vertex x and draw the resulting graph such that the length of all shortest paths is maintained for all $u, v \in V \setminus \{x\}$. **(_ /1 P.)**



b. Given $G = (V, E, c)$, design an algorithm that constructs a graph $G_x = (V \setminus \{x\}, E', c')$ such that $d_G(u, v) = d_{G_x}(u, v)$ for all $u, v \in V \setminus \{x\}$. Your algorithm should run in time $\mathcal{O}(n^2)$. Prove the correctness and analyze the runtime of your algorithm. **(_ /3 P.)**

c. Given G and a vertex x, let G_x be as described in the previous exercise. Design an algorithm that, given $G = (V, E, c)$, $x \in V$ and the pairwise shortest distances d_{G_x} of G_x, computes all shortest distances from x to $u \in V \setminus \{x\}$. Your algorithm should run in time $\mathcal{O}(n^2)$. Prove the correctness and analyze the runtime of your algorithm. **(_ /3 P.)**

d. Denote the algorithms from part **b.** and **c.** as B and C. Can you get $\mathcal{O}(n^{2-\varepsilon})$-time algorithms with $\varepsilon > 0$ for B and C? Otherwise, show that this is unlikely under one of the fine-grained hypotheses introduced in the lecture. **(_ /2 P.)**

Hint: You may use without proof that the following algorithm correctly computes all pairwise shortest distances d_G in a given graph G:

If G contains a single vertex v then return $d_G(v, v) = 0$.
Otherwise:

1. *Pick some arbitrary vertex $x \in V$*
2. *Compute G_x with algorithm B, removing vertex x.*
3. *Recurse on the graph G_x to compute all pairwise distances d_{G_x}.*
4. *Use algorithm C with d_{G_x} to compute the all pairwise distances d_G.*

Problem 4. Randomized: Sibling Sabotage **(_ /9 P.)**

You are a world-class music producer. You have just finished processing a raw audio sample vector $x \in \mathbb{F}_2^n$ through your signature digital effects matrix $M \in \mathbb{F}_2^{n \times n}$ to create the final track $y = Mx$ i.e., y is the matrix-vector product of M and x.
However, your little brother is furious that you didn't take him to the local carnival. In an act of vengeance, he accesses your workstation. He either does not find your masterpiece y, or he trashes it significantly. You are **promised** that the resulting vector y' follows one of two cases:

- **Case 1 (The Masterpiece):** $y' = Mx$.
- **Case 2 (The Sabotage):** At least $n/2$ entries are changed (in other words: the Hamming-Distance is $d_H(Mx, y') \geq n/2$).

The task is: Given M, x and y' but in particular not y, you want to decide which case occurred without recomputing $y = Mx$.

a. Assume you want to be 100% certain whether the track is Case 1 or Case 2. Prove that any deterministic algorithm must, in the worst case, examine at least $\Omega(n)$ entries of the vector y'. **(_ /2 P.)**
Hint: Consider an adversary who only modifies bits that your algorithm has not yet queried.

b. Design a randomized algorithm that: **(_ /4 P.)**

1. Always outputs the correct classification.
2. Has an **expected** runtime of $\mathcal{O}(n)$ when the track is sabotaged.

Prove the correctness and analyze the (expected) runtime of your algorithm.

c. The record label is demanding the file immediately. You switch to a **Monte Carlo** approach. You will sample k indices $\{i_1, i_2, \ldots, i_k\}$ independently and uniformly at random from $\{1, \ldots, n\}$. You consider the file y' the (untampered) Masterpiece if it holds that $(Mx)_{i_j} = y'_{i_j}$ for all $1 \leq j \leq k$. Otherwise, you consider the track sabotaged. **(_ /3 P.)**

1. Determine the minimum number of samples k required such that the probability of a False Positive (claiming the track is intact when it is actually sabotaged) is less than $1/1000$.
2. Show that there is an $\mathcal{O}(n \log n)$-time algorithm determining whether the track is sabotaged with error probability at most $\frac{1}{n}$.

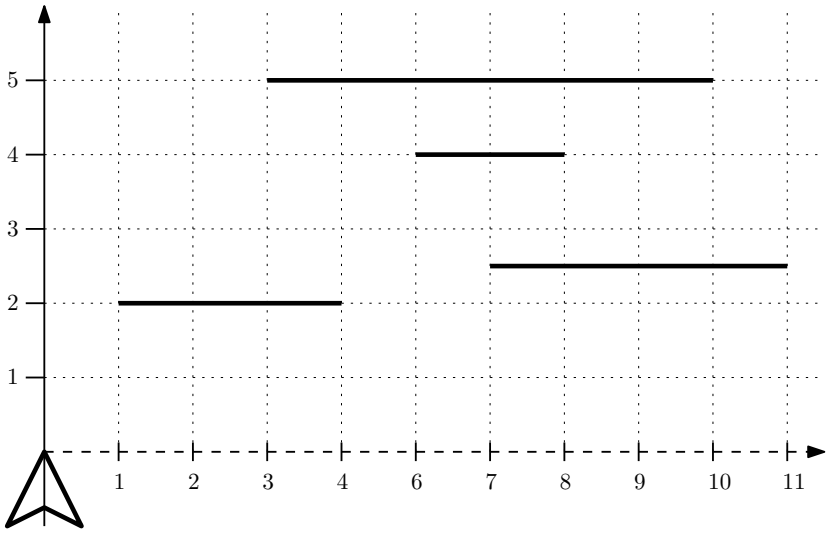
Problem 5. Geometric Algorithms: Asteroids **(_ /9 P.)**

In the basement of your grandparents' house, you found an old arcade machine, loaded up with a bootleg version of the famous "Asteroids" game. You control a spaceship equipped with powerful blasters in a 2-dimensional plane and your goal is to shoot down asteroids. Because this is a bootleg version, it works slightly differently:

- The ship may only move along the x-axis (its y-coordinate is always 0).
- Every asteroid is a static line segment parallel to the x-axis, i.e. $A_i = \overline{(\ell_i, y_i)(r_i, y_i)}$ with $y_i \geq 1$. The start and endpoint are included in the line segment.
- A shot fired from $(x, 0)$ travels instantly along the vertical line $\{x\} \times \mathbb{R}$ and only hits and destroys the first asteroid it intersects.

The spaceship starts at the origin $(0, 0)$.

a. Given the configuration below, give a sequence of x-coordinates from which to fire such that every asteroid is hit exactly once. **(_ /1 P.)**



b. Let $x_1 < x_2 < \cdots < x_n$ be a **monotone** sequence of firing positions and let $A_1, \ldots, A_m$ be the asteroids, each given as a segment $\overline{(\ell_i, y_i)(r_i, y_i)}$ with $y_i \geq 1$. **(_ /4 P.)**
Design an $\mathcal{O}(n \log n)$-time algorithm that decides whether all asteroids are destroyed. Prove the correctness and analyze the runtime.

c. After collecting some upgrades, your ship can place a powerful space bomb that explodes in a $W \times W$ square, *annihilating* and completely destroying any asteroid that is fully contained in that $W \times W$ square. A space bomb can be described by its bottom left corner (s_x, s_y), where the area covered is given by the square $[s_x, s_x + W] + [s_y, s_y + W]$. **(_ /4 P.)**
Your goal is to find the perfect position to drop the bomb, as you want to make the most use of it. Assume for the following that all asteroids are of fixed length $\alpha < W$.
Given n positions $S_1, \ldots, S_n$ for $W \times W$ space bombs and n asteroids $A_1, \ldots A_n$, each given as a segment $\overline{(\ell_i, y_i)(r_i, y_i)}$ with $y_i \geq 1$ and $r_i - \ell_i = \alpha$, Design an $\mathcal{O}(n \log n)$-time algorithm that determines the maximum number of asteroids you can *annihilate* by placing the space bomb at some position S_i with $i \in [n]$. Prove the correctness and analyze the runtime.

Problem 6. Stringology: Evenly Spaced Patterns **(_ /11 P.)**

Recall the Pattern Matching problem introduced in the lecture. We want to find more general patterns now: Given a binary string $x \in \{0,1\}^n$ of length n, determine whether there exists a pattern of three evenly spaced 1s, i.e. such that they are separated by the same number of characters.

Examples: **1**0**1**0**1**, **111**00 or **1**00**1**10**1**.
Non-Examples: 101001 or 11010001.

a. State whether the string 0001011011 contains three evenly spaced 1s. **(_ /1 P.)**

b. Prove the following claim: **(_ /2 P.)**

A string $x[1 \dots n]$ contains a pattern of three evenly spaced 1s if and only if there exist indices $i < j < k \in [n]$ with $i + k = 2j$ such that $x[i] = x[j] = x[k] = 1$.

c. Design an algorithm that determines whether there occur three evenly spaced 1s in a binary string x of length n in time $\mathcal{O}(n \log n)$. Prove the correctness and analyze the runtime of your algorithm. **(_ /5 P.)**

Hint: use the claim from b)

d. Let Σ be a finite alphabet and $W_1, W_2 \in \Sigma^k$. We are interested in detecting the pattern $P = W_1 ?^k W_2$ where '?' denotes a *wildcard* that matches any character of the alphabet Σ, i.e. W_1 and W_2 are separated by any k characters. **(_ /3 P.)**
Given a string $x \in \Sigma^n$, a number $k \in \mathbb{N}$ and $W_1, W_2 \in \Sigma^k$, design an algorithm running in time $\mathcal{O}(n+k)$ that determines whether the pattern $P = W_1 ?^k W_2$ occurs in x. Prove the correctness and analyze the runtime of your algorithm.

Example: $\underbrace{\texttt{110}}_{W_1}\underbrace{\texttt{010}}_{?^3}\underbrace{\texttt{101}}_{W_2}\texttt{01}$ contains $P = \texttt{110???101}$.

Draft Paper