Surname:
First Name:
Matriculation No.:

Karlsruhe Institute of Technology
Institute for Theoretical Informatics

Prof. Dr. Marvin Künnemann

18.09.2026

Exam Algorithms II

Problem 1.	Miscellaneous Tasks	13 Points
Problem 2.	Max-Flow: Dynamic Max Flow	10 Points
Problem 3.	Shortest Paths: Alien Abduction	8 Points
Problem 4.	Stringology: Patterns and Prefixes	9 Points
Problem 5.	Geometric: Discrete Fishing	10 Points
Problem 6.	Approximation: Tourist Taxi Toll	10 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**.

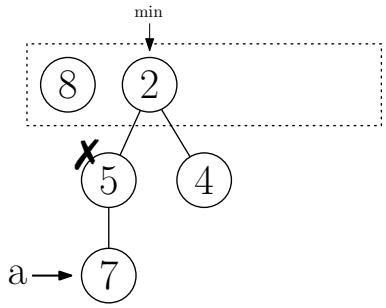
Problem	**1**	**2**	**3**	**4**	**5**	**6**	**Total**
max. Points	13	10	8	9	10	10	60
Achieved							
						Grade:	

Problem 1. Miscellaneous Tasks **(_ /13 P.)**

a. Given a Fibonacci Heap in the state depicted below, perform the following operations on it. The ✗ denotes a node as marked and the min-ptr the current minimum. Draw the resulting state (including any marks and the min-ptr) after every operation **and right before union-by-rank**. **(_ /4 P.)**

Hint: You should draw 4 states in total.

1. `decreaseKey(a,3)`
2. `deleteMin()`
3. `insert(1)`



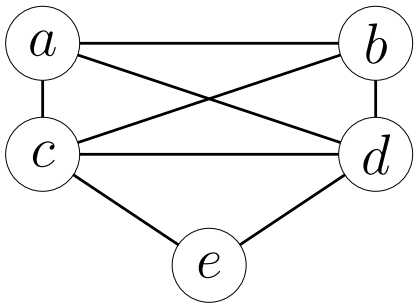
b. Consider algorithms with the following running times for input size n and parameter $k \in \mathbb{N}$. State whether they are fixed-parameter tractable times and justify your answer. **(_ /3 P.)**

1. $(k!) \cdot n^{1+\log k}$
2. $\frac{(n-k)^k}{2^k}$
3. $(k \cdot n \log n)^3$

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 an FPTAS, a PTAS but not an FPTAS, or neither. Justify your answer. **(_ /2 P.)**

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

d. Consider the following unweighted and undirected graph. **(_ /2 P.)**
Give a contraction order for the edges, such that Karger's algorithm yields a minimum cut of the graph (when contracting edges in this order). Draw the resulting multigraph.
Tip: You can denote the edges by their vertices in the starting graph even after contracting.



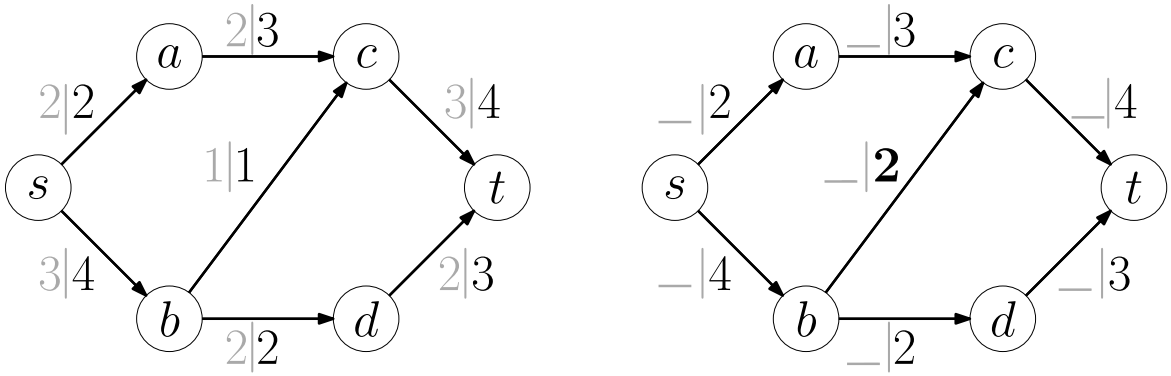
contraction order = _____

e. Assume you are given a large list S of n primitive (comparable) elements that does not fit into main memory of size M. Let B be the block size. Show that you can count the number of distinct elements in S with $\mathcal{O}\left(\frac{n}{B}\left(1+\log_{M/B}\frac{n}{M}\right)\right)$ I/O-accesses. **(_ /2 P.)**

Problem 2. Max-Flow: Dynamic Max Flow **(_ /10 P.)**

In the following we are working with a flow network $(G = (V,E), c, s, t)$ with integer capacities $c : E \to \mathbb{N}_0$. As usual, we denote the source and sink by $s, t \in V$.

a. Given the following network with a maximum flow f with edges labelled by $f(e)|c(e)$, compute the new maximum flow after increasing the edge capacity as shown below. **(_ /1 P.)**



In the following tasks, you are given the network and an integer maximum flow f in G.

b. Design an algorithm that, given an edge $e \in E$, **increases** its capacity by 1 and updates f such that it is still a maximum flow. Your algorithm should run in time $\mathcal{O}(n+m)$. Prove the correctness of your algorithm and analyze its runtime. **(_ /3 P.)**

c. Design an algorithm that, given an edge $e \in E$ with $c(e) \geq 1$, **decreases** its capacity by 1 and updates f such that it is still a maximum flow. Your algorithm should run in time $\mathcal{O}(n+m)$. Prove the correctness of your algorithm and analyze its runtime. **(_ /4 P.)**

Note: If it helps, you may assume that there is no cycle with all edges having positive flow under f.

d. Show that there exists an infinite sequence of networks $G_1, G_2, \ldots$ where network G_x contains some edge e such that increasing the capacity of e by x results in x distinct augmenting paths in the residual graph. It suffices to describe how to construct G_x for any given x. You do not need to prove that G_x satisfies this property. **(_ /2 P.)**

Problem 3. Shortest Paths: Alien Abduction **(_ /8 P.)**

Two scientists have been tracking alien abductions throughout Baden-Württemberg. They managed to precisely calculate the probability of **not** being abducted by aliens when traveling from one city to another.

To this end, they have constructed a directed graph $G = (V, E, p)$, where the vertices V represent cities and the edges E represent direct connections between cities. Each edge $(u, v) \in E$ is labelled by the probability $p(u, v)$ of not being abducted when traveling from city u to city v. As the aliens are very peculiar about numerical values, the probabilities only take values $p(u, v) = \frac{1}{2^k}$ for some $k \in \mathbb{N}_0$.

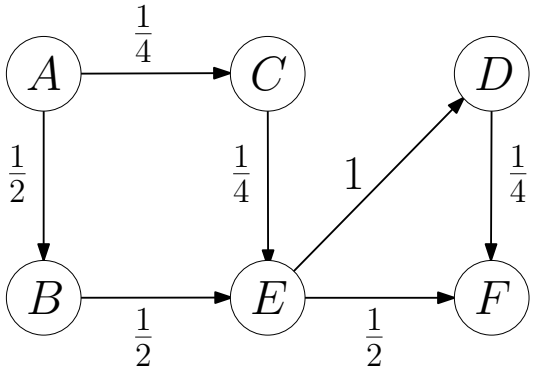
For a starting city and a destination city, the probability of not being abducted is given by the product of the probabilities along the edges of the path taken. We denote the probability of not being abducted on a path $P = s, e_1, v_1, \ldots, v_{l-1}, e_l, t$ as its *safety*:

$$\text{safety}(P) = \prod_{i=1}^{l} p(e_i).$$

Finally, for two cities s, t we denote by $\text{safety}(s,t)$ the maximum safety over all s-t-paths in G. If there is no path from s to t we define $\text{safety}(s,t) = 0$. Finally, for a city $s \in V$ we define $\text{safety}(s,s) = 1$.

Note: You can use the previous subtasks to solve later problems, even if you did not answer them.

a. Consider the labelled graph given below. Compute the safest path from city A to city F and state $\text{safety}(A, F)$. **(_ /1 P.)**



path = _____

safety = _____

b. Given $G = (V, E, p)$ where $p : E \to \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8} \ldots\}$, design an algorithm that, given a starting city $s \in V$, computes $\text{safety}(s,t)$ for all $t \in V \setminus \{s\}$ in time $\mathcal{O}(m + n \log n)$. Prove the correctness and analyze the runtime of your algorithm. **(_ /3 P.)**

c. Due to unforeseen circumstances, the scientists are split up and find themselves in two different cities s_1 and s_2. They want to meet in a city t while trying to ensure the most dangerous of the two journeys is as safe as possible. **(_ /3 P.)**
Given $G = (V, E, p)$ where $p : E \to \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8} \dots\}$ and starting cities $s_1, s_2 \in V$, design an algorithm that computes the safest meeting point $t \in V$ such that $\min(\text{safety}(s_1, t), \text{safety}(s_2, t))$ is maximized. Your algorithm should run in $\mathcal{O}(m + n \log n)$ time. Prove the correctness and analyze the runtime of your algorithm.

d. Can there exist a safest s-t-path (not necessarily simple) that contains a cycle? Briefly argue your answer. **(_ /1 P.)**

Problem 4. Stringology: Patterns and Prefixes **(_ /9 P.)**

We introduce a generalization of the String Matching Problem, in which the pattern P is allowed to have the wildcard-symbol $\star$ which matches any number of characters (including the empty string).

A pattern $P \in (\Sigma \cup \{\star\})^*$ with wildcards $\star$ matches a string $X \in \Sigma^*$ if by replacing each occurrence of a wildcard $\star$ in P by some (individual) string over Σ, we can obtain a substring of X.

a. Consider the following string $X =$ `ABERACADABERA`. Finish the following table by indicating whether the pattern matches X: **(_ /1 P.)**

Pattern	matches X?
`D⋆C`	✗
`AB⋆RACADAB⋆RA`	✓
`BE⋆C⋆BE`	✓
`A⋆B⋆C⋆D⋆E`	

b. Describe an algorithm that, given a string $X \in \Sigma^*$ of length n and pattern $P \in (\Sigma \cup \{\star\})^*$ of length m that contains q wildcards $\star$, decides whether P matches X. **(_ /4 P.)**

Your algorithm should run in $\mathcal{O}((q+1)n+m)$ time. Prove the correctness and analyze the running time.

c. You are given a dictionary of strings $D \subseteq \Sigma^*$ stored in a trie. We want to compute for a given string $w \in \Sigma^*$ the number of strings $d \in D$ such that w is a prefix of d or d is a prefix of w. **(_ /4 P.)**

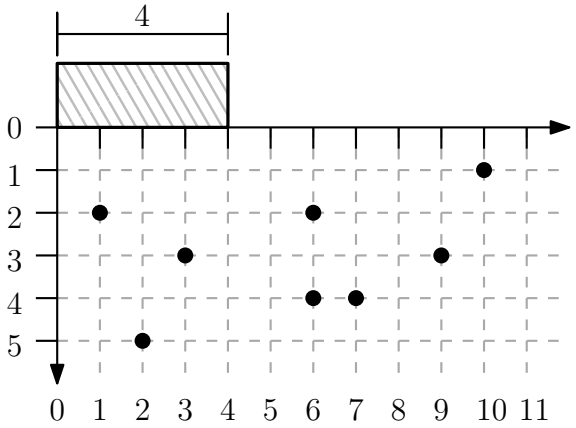
You are allowed to save additional information to the nodes of the trie, if your modifications do not change the asymptotic running times of its operations.

Describe an algorithm (and modifications to the data structure) that answers such a query in time $\mathcal{O}(|w|)$. Prove the correctness and analyze the running time of your algorithm.

Problem 5. Geometric: Discrete Fishing **(_ /10 P.)**

We want to help a crew of fishers secure their dinner in the Integer-Ocean spanning the coordinates $\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0}$. The ship sails along the surface at $y = 0$, has a length of $L \in \mathbb{N}$, and its position is defined by the coordinate $t \in \mathbb{Z}_{\geq 0}$ of its leftmost side. Thus at position t, the ship *covers* the $L+1$ integer x-coordinates $\{t, t+1, \ldots, t+L\}$.
There are m fish in the sea, located at (integer) coordinates $F = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. The crew is equipped with highly advanced, infinite-length fishing rods that drop straight down parallel to the y-axis. If a rod is *deployed* at a surface coordinate x_r currently covered by the ship, it will catch all fish located at (x_r, y) for $y \in \mathbb{Z}_{\geq 0}$. As the infinite-length fishing line is very delicate, they can only throw out their rods if the ship is at rest.

a. Assuming the crew can deploy an unlimited number of fishing-rods, what is the optimal position t for the ship such that the maximum number of fish can be caught? **(_ /1 P.)**



b. Assume the crew can deploy an unlimited number of fishing-rods. Design an algorithm running in time $\mathcal{O}(m \log m)$ that, given ship length L and fish coordinates F, determines an optimal discrete position t such that the largest number of fish can be caught at this position. Prove the correctness and analyze the runtime. **(_ /4 P.)**

c. It turns out the crew members are very peculiar: they only sit at fixed integer offsets $P \subseteq \{0,\dots,L\}$ relative to the left side of the ship. If the ship is at position t, the crew member at offset $p \in P$ can deploy their fishing-rod only at position $t+p$. **(_ /5 P.)**

Let $X = \max_{(x_i,y_i)\in F} x_i$ be the maximum x-coordinate of any fish.

Design an algorithm that, given ship length L, fish coordinates F and crew seats P, computes a ship position $t \in \mathbb{Z}_{\geq 0}$ that maximizes the total number of fish caught by the crew. Your algorithm should run in time $\mathcal{O}(m + (L+X)\log(L+X))$.

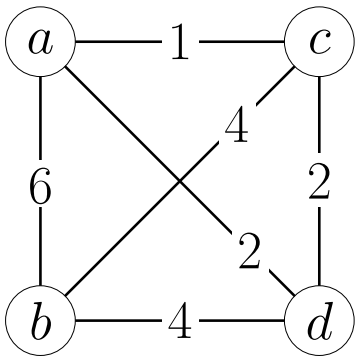
Prove the correctness and analyze the runtime.

Problem 6. Approximation: Tourist Taxi Toll **(_ /10 P.)**

You are a taxi driver who offers sightseeing tours for tourists. Instead of taking the shortest path, you want to find a route that visits every landmark exactly once and returns to the starting vertex while **maximizing** the total fare of the tour (luckily the tourists are too busy being amazed by the sights to care).

Let $G = (V, E)$ be a complete undirected graph with edge weights $w : E \to \mathbb{N}_0$, which models the landmarks as vertices, the roads as edges and the fare between two landmarks as the edge weight. You want to find the *maximum fare tour* $T \subseteq E$ that visits every vertex once and returns to the starting vertex, while maximizing the total edge weight $w(T) = \sum_{e \in T} w(e)$.

a. Given the graph below, state the total fare of the maximum fare tour. **(_ /1 P.)**



total fare = _____

b. The local authorities are getting suspicious of your exorbitant fares. You want to prove to them that it is in fact (under plausible assumptions) too hard to maximize the fare, so their suspicion is unfounded. **(_ /4 P.)**

Show that there exists no polynomial time $\alpha(n)$-approximation algorithm A with a ratio $\alpha(n) > \frac{n-1}{n}$ for finding the maximum fare tour, unless P = NP.

Hint: You can use that Hamiltonian Cycle is NP*-complete: Given an undirected graph, the problem asks whether there is a tour T that visits every vertex once.*

The authorities believed your proof and dismissed their suspicions.
Unfazed and undisturbed you proceed with your daily hustle using the following greedy algorithm:

Algorithm $\mathscr{A}$

Input: Complete undirected graph $G = (V, E)$ with non-negative edge weights w
Output: approximate maximum fare tour $T \subseteq E$

```
1: T ← ∅
2: U ← V                                   ▷ Set of unvisited vertices
3: Pick an arbitrary starting vertex s ∈ U and let v := s
4: U ← U \ {v}
5: while U ≠ ∅ do
6:     Pick a vertex v' ∈ U that maximizes w({v, v'})
7:     T ← T ∪ {{v, v'}}
8:     U ← U \ {v'}
9:     v ← v'
10: T ← T ∪ {{v, s}}
11: return T
```

Prove that algorithm $\mathscr{A}$ reaches a $\frac{1}{2}$-approximation ratio. To this end, order the vertices $\{v_1, \ldots, v_n\}$ as visited by the greedy tour T. We now assign the following sets of edges to each vertex v_i for $i \in \{1, \ldots, n-1\}$ based on this ordering:

$$E_i = \{\{v_i, v_j\} \mid j > i\}.$$

Let OPT be an optimal tour in the following.

c. Show that $|\text{OPT} \cap E_i| \leq 2$ for every $i \in \{1, \ldots, n-1\}$. **(_ /1 P.)**

d. Show for $i \in \{1, \ldots, n-1\}$ that $\max_{e \in E_i} w(e) \leq w(\{v_i, v_{i+1}\})$ for the given ordering. **(_ /1 P.)**

e. Using the previous subtasks, show that $\mathscr{A}$ has an approximation ratio of $\frac{1}{2}$. **(_ /3 P.)**

Draft Paper