Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Johannes Lengler
Hongjie Chen
Kasper Lindberg

David Steurer
Manuel Wiedmer
Lucas Slot

Exam

Algorithmen und Datenstrukturen

January 21, 2025

DO NOT OPEN!

Student number: _____

Seat number: _____

Good luck!

	T1 (14P)	T2 (16P)	T3 (17P)	T4 (13P)	Prog. (40P)	**Σ** (100P)
Score						
Corrected by						

/ 14 P

Theory Task T1: Complexity and O-notation.

/ 6 P

a) *Asymptotic notation quiz:* For each of the following claims, state whether it is true or false. You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total. You don't have to justify your answer.

Assume $n \geq 4$.

Claim	**true**	**false**
$n^3 - n^2\sqrt{n} \leq O(\sqrt{n})$	☐	☐
$\frac{n^{2.5}}{1+3n} = \Theta(n)$	☐	☐
$2^{(2n)} \leq O(10^n)$	☐	☐
$\sum_{i=1}^{\lceil \log n \rceil} i \leq O(\log(n)^2)$	☐	☐
Suppose $a_n \geq \Omega(n^2)$ and $b_n \leq O(n)$ (and $a_n, b_n > 0$). It follows that $\frac{a_n}{b_n} \geq \Omega(n)$.	☐	☐
Suppose $a_1 = 10$ and $a_n = 4a_{n-1} + 2$ for all $n \geq 2$. It follows that $a_n \leq O(5^n)$.	☐	☐

/ 4 P

b) *Counting iterations:* For the following code snippets, give the exact number of times f is called as a function of n. For the exact number, you are allowed to use summation signs in your expression. Then, derive an asymptotic bound for the number of times f is called in Θ-notation. Simplify your expression as much as possible (in particular without summation signs).

i) Snippet 1:

Algorithm 1

```
for i = 1, 2, 3, ..., n^2 do
    for k = 1, 2, 3, ..., i do
        f()
```

Exact number of calls:

Asymptotic number of calls in simplified Θ-notation:

ii) Snippet 2:

Algorithm 2

```
i = 1
while i ≤ n do
    j = 1
    while j ≤ n do
        j ← 2j
        f()
    i ← i + 1
```

Exact number of calls:

Asymptotic number of calls in simplified Θ-notation:

/ 4 P

c) *Sorting and searching algorithms:* For each question below, there is exactly one correct answer. Please circle the correct answer for each question. You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total. You don't have to justify your answer.

i) `HeapSort` needs time at most $O(n \log n)$ to sort any array of length n.

(a) True

(b) False

ii) Assume $n \geq 10$. For which of the following arrays does `InsertionSort` take time $\Omega(n^2)$ (to sort the array in ascending order)?

(a) $[2, 1, 3, 4, 5, \ldots, n-2, n-1, n]$

(b) $[n, n-1, n-2, \ldots, 3, 2, 1]$

(c) For both

(d) For neither

iii) The implementation of `BubbleSort` from the lecture uses two nested for-loops to sort an array in ascending order. Which of the following statements is correct?

(a) After j iterations of the outer loop, the largest j elements of the array are in the correct place.

(b) After j iterations of the outer loop, the first j elements of the array are in (ascending) order.

iv) Let H be the binary decision tree associated to a comparison-based algorithm to search in sorted arrays of length n. Which of the following statements about the depth ℓ of H is certainly correct?

(a) $\ell \geq \Omega(n)$

(b) $\ell \leq O(n)$

(c) $\ell \geq \Omega(\log n)$

(d) $\ell \leq O(\log n)$

Theory Task T2: Graphs. **/ 16 P**

/ 5 P a) *Graph quiz:* For each of the following claims, state whether it is true or false. You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total. You don't have to justify your answer.

As a reminder, here are a few definitions. For an undirected graph $G = (V, E)$:

- A *(closed) Eulerian walk* is a (closed) walk which traverses every edge in E exactly once.
- A *Hamiltonian path* is a path that contains every vertex.
- A *spanning tree* of G is a subset $F \subseteq E$ such that (V, F) is a tree.

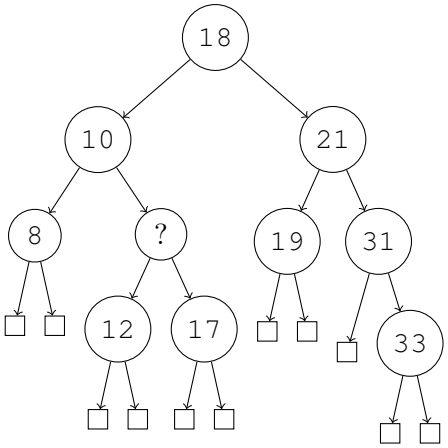
In a directed graph:

- A *sink* is a vertex with out-degree equal to zero.
- A *source* is a vertex with in-degree equal to zero.

Claim	**true**	**false**
If we run DFS on a connected directed graph, then the DFS tree is a shortest path tree.	☐	☐
Every undirected graph that contains a Hamiltonian path also contains an Eulerian walk.	☐	☐
If all vertices have even degrees in an undirected graph, then the graph contains a closed Eulerian walk.	☐	☐
A directed graph that contains a source must also contain a sink.	☐	☐
Let G be a connected undirected graph. Assume that G has a unique spanning tree. Then G must be a tree.	☐	☐

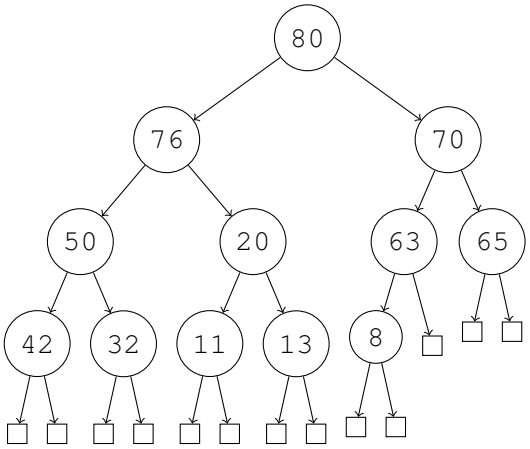
/ 3 P b) *Binary trees:*

i) Consider the following binary tree. What integer values can be the key of the node "?" such that the tree is a **binary search tree** with all keys being distinct? List all possible values. You don't have to justify your answer.



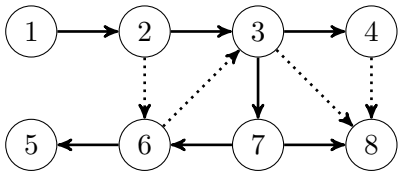
ii) Draw the **AVL tree** that is obtained from inserting the keys, 8, 13, 9, 12 and 10 into an initially empty tree. It suffices to only draw the final tree.

iii) Extract the maximum of the following **max-heap** and do all necessary operations to restore the heap condition (it suffices to only draw the final tree).



/ 4 P c) *Breadth/depth-first search:*

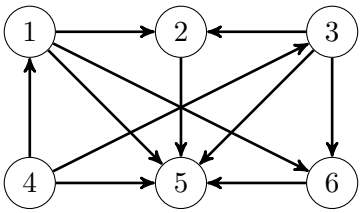
i) Consider the following directed graph:



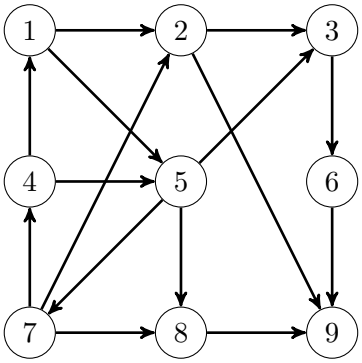
The solid edges indicate a depth-first tree. Classify the remaining edges (choose from "forward", "backward", and "cross").

- $(2, 6)$:
- $(3, 8)$:
- $(4, 8)$:
- $(6, 3)$:

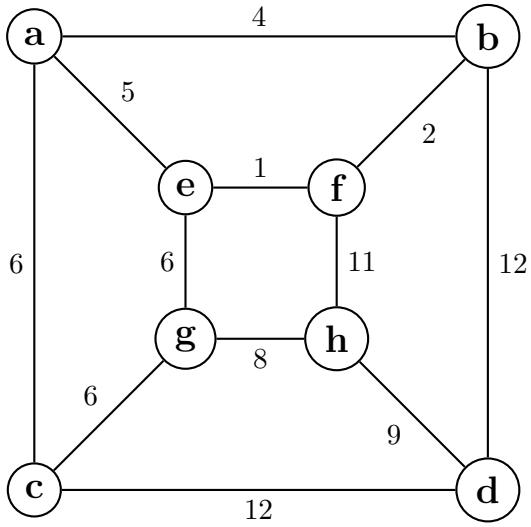
ii) Does the following graph have a topological sorting? If yes, write down a topological sorting. If no, give an argument.



iii) Consider the following graph. Draw the breadth-first search tree resulting from a breadth-first search starting from vertex 1. Process the neighbors of a vertex in increasing order.



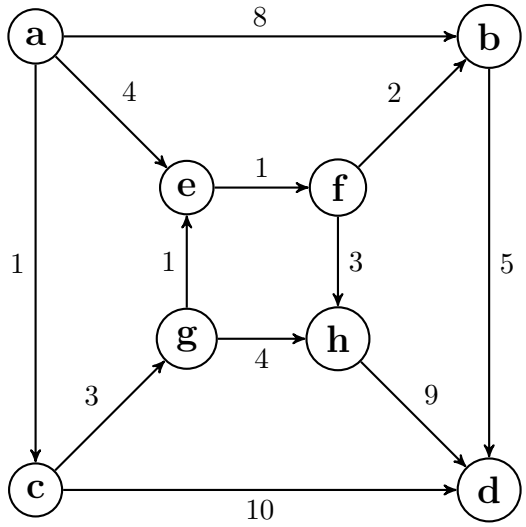
/ 2 P d) *Minimum Spanning Tree:* Consider the following graph:



i) Find a minimum spanning tree. You can either highlight its edges in the picture above, or draw the minimum spanning tree below. (If there is more than one minimum spanning tree, it suffices to just find one of them.)

ii) Is there a minimum spanning tree containing the edge {f,h}? If yes, draw the corresponding minimum spanning tree. If no, give an argument.

/ 2 P e) *Shortest Path Tree:* Consider the following graph:



i) Find a shortest-path tree rooted at vertex a (e.g., the set of edges selected by Dijkstra's algorithm starting from vertex a). You can either highlight its edges in the picture above, or draw the shortest-path tree below. (If there is more than one shortest-path tree, it suffices to just find one of them.)

ii) Now you can replace the weight 2 of the edge (f, b) by some integer x. What is the maximum value of x such that some shortest-path tree rooted at vertex a in the resulting graph includes the edge (f, b)? You don't have to justify your answer.

Theory Task T3: Algorithm Design. **/ 17 P**

/ 7 P a) *Dynamic Programming.* You are given a tree $T = (V, E)$ rooted at a vertex $r \in V$ where $V = [n]$. The *subtree rooted at a vertex* i, denoted T_i, is the subgraph of T with vertex i and all its descendants[1] as well as all edges between these vertices. For a vertex $i \in V$, let $C(i)$ denote the set of children[2] of i and $G(i)$ denote the set of grandchildren[3] of i.

A subset of vertices $S \subseteq V$ is called a *vertex cover* if every edge in the tree has at least one of its endpoints in S.

We want to find the size of a smallest vertex cover. We wish to solve this problem using *dynamic programming*. We use a DP table $M[1 \dots n]$ of size n and $M[i]$ will denote the size of a smallest vertex cover of the subtree T_i.

The recurrence relation for $M[i]$ is split into two cases. Let $A[i]$ denote the size of a smallest vertex cover of the subtree T_i where i is included in the vertex cover. Let $B[i]$ denote the size of a smallest vertex cover of the subtree T_i where i is not included. Then $M[i]$ will be the minimum of these two quantities.

i) What vertices of T will be base cases of the recursion (meaning their values do not depend on any other value of M) and what will their corresponding values be in the array M?

ii) The recursion for all other vertices i looks like

$$M[i] = \min\{A[i],\ B[i]\}.$$

Each of these terms can be written in the form

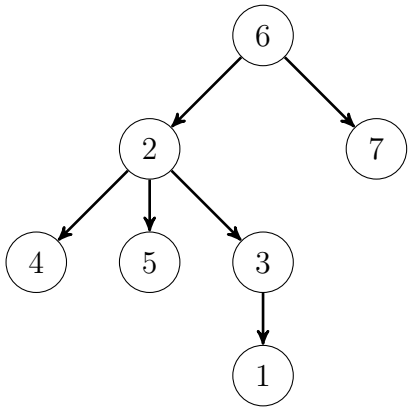
$$A[i] = \square + \sum_{j \in C(i)} \square + \sum_{j \in G(i)} \square,$$
$$B[i] = \square + \sum_{j \in C(i)} \square + \sum_{j \in G(i)} \square.$$

Write down what the correct choices for each box are so that the above equations correctly compute the values $A[i]$ and $B[i]$ (a sum over an empty set equals zero). You don't need to justify your answers.

Hint: *You can also write* 0 *into some boxes.*

[1] Nachkomme in German
[2] A child of v is a descendant of v which is a direct neighbor of v
[3] A grandchild of v is a child of a child of v

iii) For the following tree, write down a valid calculation order for the above recursion. You can either indicate it directly in the tree or write down a sequence of vertices.



/ 6 P

b) *Modelling.* There are n currencies $C_1, C_2, \ldots, C_n$ and for each $i, j \in \{1, \ldots, n\}$, $i \neq j$, there is an exchange which converts currency C_i to C_j with nonzero rate $W_{ij} > 0$. The two rates W_{ij} and W_{ji} can be unrelated. Determine if it is possible to start with 1 unit of currency C_1 and acquire at least 2 units of currency C_1 with a sequence of exchanges.

Model this task as a graph problem and describe an algorithm which outputs POSSIBLE if there exists a sequence of exchanges which converts 1 unit of currency C_1 to at least 2 units of currency C_1 and outputs IMPOSSIBLE otherwise. The runtime of your algorithm should be at most $O(n^3)$.

Hint: *If you want to decide for $a_1, \ldots, a_k > 0$ whether $a_1 \cdot a_2 \cdot \ldots \cdot a_k > 1$, it is equivalent to test whether $-(\log(a_1) + \log(a_2) + \ldots + \log(a_k)) < 0$. Design an algorithm which performs this latter test.*

Address the following elements:

1) The graph that you use. This includes: its vertex set and edge set; whether edges are directed or not; whether vertices or edges are weighted and if so, their weights.
2) The algorithm that you apply to this graph. You can use the algorithms covered in the lecture material as subroutines, and you can use their running time bounds without proof.
3) How you extract the solution from (the output of) your algorithm.
4) The total running time of your proposed algorithm, with a short justification.

Graph:

Algorithm:

Scheme continues on the next page.

Extraction:

Runtime:

/ 4 P

c) *Layering.* You are given a directed graph $G = (V, E)$ which includes two vertices v_{start} and v_{end}. For each $v \in V \setminus \{v_{\text{start}}\}$ the graph has a positive vertex weight $w_v > 0$ and $w_{v_{\text{start}}} = 0$. You are also given two positive integers $L, T \in \mathbb{N}$. We wish to determine if there exists a walk from v_{start} to v_{end} with exactly L edges and weight at most T where the weight is the sum of the weights of the vertices on the walk.

Consider a new graph H with vertex set $V \times \{0, 1, \ldots, L\} = \{(v, i) \mid v \in V,\, i \in \{0, 1, \ldots, L\}\}$. The plan is to run Dijkstra's algorithm on H starting at $(v_{\text{start}}, 0)$ and check whether the distance to (v_{end}, L) is at most T.

Give an edge set with corresponding edge weights for H so that this approach correctly solves the task above (you do not need to prove correctness).

Theory Task T4: Proofs. / 13 P

/ 4 P a) *Induction:* Let $a_1 = 4$, $a_2 = 16$ and

$$a_n = 3a_{n-1} + 4a_{n-2} - 4, \quad \text{for } n \geq 3.$$

Prove the following inequality by induction: for every integer $n \geq 1$,

$$a_n \leq 4^n.$$

Base case(s):

Induction hypothesis:

Induction step:

/ 5 P

b) *Many options?:* Let $G = (V, E)$ be an undirected, connected graph with $n = |V|$ vertices.

For each of the following statements, prove that they are correct, or give a counterexample and a brief argument why it is indeed a counterexample:

i) Let $k \in \mathbb{N}$ and suppose that $T_1, T_2, \ldots, T_k \subseteq E$ are (pairwise) disjoint spanning trees of G. Then $k \leq O(n)$.
(two spanning trees $T, T' \subseteq E$ are disjoint if $T \cap T' = \emptyset$, i.e., they share no edge.)

ii) Let $v, w \in V$. The number of distinct paths $P \subseteq E$ from v to w in G is at most $O(n^{10})$.
(two subsets $P_1, P_2 \subseteq E$ are distinct if $P_1 \neq P_2$.)

/ 4 P

c) *Lightest, shortest path:* You are given a directed graph $G = (V, E)$ with positive edge weights $w_e > 0$ for $e \in E$, and two vertices $u, v \in V$. You wish to find a path $P \subseteq E$ from u to v in G with the following properties:

- $|P| = \min\{|P'| : P' \text{ is a path from } u \text{ to } v \text{ in } G\}$.
- $w(P) = \min\{w(P') : P' \text{ is a path from } u \text{ to } v \text{ in } G \text{ and } |P'| = |P|\}$.

(*Recall that* $w(P) := \sum_{e \in P} w_e$*, and that* $|P|$ *is the number of edges in* P*.*)

Your colleague suggests the following algorithm: First, increase all weights by $C = 2 \sum_{e \in E} w_e$. Then, compute a shortest path (i.e., of minimum weight) from u to v in the new graph, and output this path.

Assuming there is at least one path from u to v in G, prove that this algorithm is correct.

Extra space for notes. If you write any answers here that should be graded, please indicate clearly to which task your notes belong, and make a note in the main body of the exam.

Extra space for notes. If you write any answers here that should be graded, please indicate clearly to which task your notes belong, and make a note in the main body of the exam.

Extra space for notes. If you write any answers here that should be graded, please indicate clearly to which task your notes belong, and make a note in the main body of the exam.