Eidgenössische
Technische Hochschule
Zürich

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

Department Informatik
Markus Püschel
David Steurer

Johannes Lengler
Gleb Novikov
Chris Wendler

Repetition Exam

Algorithmen und Datenstrukturen

SUMMER 2020

DO NOT OPEN!

Last name, first name: _____

Student number: _____

With my signature I confirm that I can participate in the exam under regular conditions. I will act honestly during the exam, and I will not use any forbidden means.

Signature: _____

Good luck!

	P1: 24P	P2: 16P	T1: 19P	T2: 21P	T3: 8P	T4: 12P	**Σ**: 100P
Score							
Sign.							

Theory Task T1. **/ 19 P**

Notes:

1) In this problem, you have to provide **solutions only**. You do not need to justify your answer.
2) We assume letters to be ordered alphabetically and numbers to be ordered ascendingly, according to their values.

/ 4 P a) *Landau notation:* Fill out the quiz about asymptotic notation below. 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.

claim	**true**	**false**
$(2n + n^2 + 3)^2 = \Theta(n^4)$	☐	☐
$\frac{n}{\log n} \leq \mathcal{O}(\sqrt{n})$	☐	☐
$\log(n!) = \Theta(n \log n)$	☐	☐
$\sum_{i=1}^{\log_5 n} 5^i \geq \Omega(n \log n)$	☐	☐

/ 6 P b) *Graph quiz:* In the following, for a given (undirected) graph $G = (V, E)$, let $n = |V|$, $m = |E|$, and let $A \in \{0, 1\}^{n \times n}$ be the adjacency matrix of G. Indicate whether the following statements are true for all graphs G.

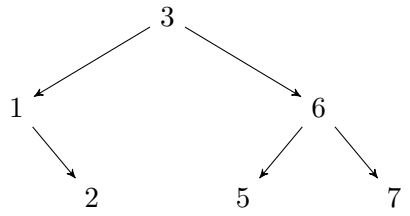
You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total.

claim	**true**	**false**
A tree with n vertices must have $n - 1$ edges.	☐	☐
The complete graph K_d with $n = d$ vertices cannot be coloured with less than d colours.	☐	☐
An Eulerian walk visits every edge exactly once.	☐	☐
Let $B = A^k$ be the adjacency matrix raised to the k-th power. If $B_{ij} > 0$ then there is a *path* of length k from i to j.	☐	☐
In an unweighted graph, both BFS and DFS can be used to determine shortest paths.	☐	☐
A binary tree of height h (the root has height 0) has at most 2^h leaves.	☐	☐

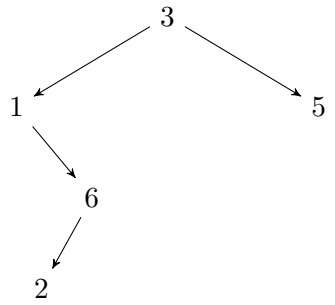
/ 4 P c) *Search trees:*

i) Draw the binary search tree that is obtained when inserting the keys $3, 6, 1, 5, 7, 2$ in this order into an empty tree.

ii) Draw the binary search tree obtained from the following tree by performing the operation `DELETE`(3).

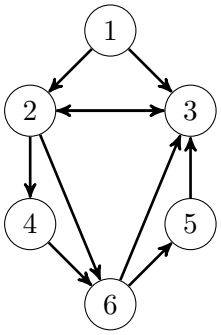


iii) Draw the **AVL tree** that is obtained from the following tree by restoring the AVL-condition.



iv) Draw the binary tree that has the pre-order traversal A, B, D, E, C, F, G and the post-order traversal D, E, B, G, F, C, A.

/ 3 P d) *Depth-first search / breadth first search:* Consider the following directed graph:



i) Draw the depth-first tree resulting from a depth-first search starting from vertex 1. When processing the neighbors of a vertex, process them in increasing order.

ii) Draw the breadth-first search tree resulting from a breadth-first search starting from vertex 1. When processing the neighbors of a vertex, process them in increasing order.

iii) Remove one edge from the graph such that it can be topologically sorted, and give a topological ordering of the resulting graph.

/ 2 P e) *Sorting algorithms:*

Below you see four sequences of snapshots, each obtained during the execution of one of the following five algorithms: `InsertionSort`, `SelectionSort`, `QuickSort`, `MergeSort`, and `BubbleSort`. For each sequence, write down the corresponding algorithm.

8	6	4	2	5	1	3	7
6	4	2	5	1	3	7	8
4	2	5	1	3	6	7	8

Algorithm:

8	6	4	2	5	1	3	7
1	6	4	2	5	8	3	7
1	2	4	6	5	8	3	7

Algorithm:

8	6	4	2	5	1	3	7
6	8	2	4	1	5	3	7
2	4	6	8	1	3	5	7

Algorithm:

8	6	4	2	5	1	3	7
6	8	4	2	5	1	3	7
4	6	8	2	5	1	3	7

Algorithm:

/ 21 P

Theory Task T2.

Notes: In this part, you should **justify your answers briefly**.

/ 2 P a) *Induction:* Prove by mathematical induction that for any positive integer $n \geq 3$,

$$n^2 \geq 2n + 3\,.$$

/ 3 P b) *Factorial:*

i) Provide pseudo code using a `for`-loop for the computation of the factorial of a positive integer n:

$$n! = n(n-1)(n-2)\cdots 1.$$

ii) Prove the correctness of your algorithm via mathematical induction for all $n \in \mathbb{N}$.

/ 3 P c) *Recurrence relations:*

For this exercise, you may use the following master theorem from exercise sheet 4:

Theorem 1 (Master theorem) *Let $T : \mathbb{N} \to \mathbb{R}^+$ be a non-decreasing function such that for all $k \in \mathbb{N}$ and $n = 2^k$,*

$$T(n) \leq aT(n/2) + \mathcal{O}(n^b)$$

for some constants $a > 0$ and $b \geq 0$. Then

- *If $b > \log_2 a$, $T(n) \in \mathcal{O}(n^b)$.*
- *If $b = \log_2 a$, $T(n) \in \mathcal{O}(n^{\log_2 a} \cdot \log n)$.*
- *If $b < \log_2 a$, $T(n) \in \mathcal{O}(n^{\log_2 a})$.*

i) Consider the following recursive function that takes as an input a positive integer m that is a power of 2. Further, let $c \geq 4$ be a power of two.

Algorithm 1 $g(m)$

```
if m > 1 then
    for i = 1, 2, 3, 4, ..., c do
        g(m/2)
        for j = 1, 2, 4, 8, ..., c do
            f()
else
    return 0
```

Let $T(m)$ be the number of calls of the function f in $g(m)$. Give a recursive formula for $T(m)$.

Determine $T(m)$ in $\mathcal{O}$-notation.

/ 6 P

d) *Shortest Paths with BFS:* Consider a directed weighted graph G with vertices V, edges E and positive integer edge weights $w : E \to \mathbb{N}$.

i) State the name of an efficient algorithm for computing the lengths of all shortest paths starting from $s \in V$.

ii) What is the asymptotic running time of the algorithm?

When the edge weights are small positive integers, i.e., $w\colon E \to \{1, 2, \ldots, d\}$, it is possible to modify the graph such that the shortest path problem can be solved using BFS on the

modified graph $G' = (V', E')$.

iii) Explain how to modify the graph G with small positive integer weights such that the lengths of the shortest paths starting from $s \in V$ can be computed using BFS. How do you read off the length of a shortest path from s to v in G from the BFS in G'?

iv) What is the number of vertices and edges in the modified graph G'?

v) What is the running time of BFS on G'?

/ 7 P

e) *Top k Elements:* You are given an array of n integers. The goal is to extract the k largest elements from that sequence. Assume that both $k, n \in \mathbb{N}$ are known and that n is significantly larger than k.

i) Which data structure presented in the lecture can be used to solve this task efficiently?

ii) Provide an algorithm (using pseudo code) that utilizes the data structure from i) to extract the k largest elements from a given input array $x_1, \ldots, x_n$ of integers.

iii) Analyze the running time (asymptotic amount of operations) and memory requirements of your solution.

iv) Now, consider the modified problem: Instead of a finite array $x_1, x_2, x_3, \ldots, x_n$, you now have to process an infinite sequence $x_1, x_2, x_3, \ldots$. Formally, at each time step $t = 1, 2, 3, \ldots$, you get a new integer x_t. Thus, n is not known anymore but k is still known. At some point, you will be asked to provide the k largest elements of $x_1, \ldots, x_t$. But you do not know in advance when that request will come. You may assume that the first request occurs for some $t > k$.

Give an efficient solution to this task by completing the following pseudo code such that each call of the function `TopK` prints the k currently largest elements, i.e., at time t the k largest elements within $x_1, \ldots, x_t$. Your solution should require at most $\mathcal{O}(k)$ memory.

Algorithm 2 processElement(datastructure, x)

// Process the integer x. Update your data structure if necessary.

return datastructure

Algorithm 3 InfiniteTopK(k, $x_1, x_2, x_3, \dots$)

// Process the integers $x_1, x_2, x_3, \dots$ such that `TopK` exhibits the desired behavior.
// Initialize your data structure:
datastructure $\leftarrow$

for $t = 1, 2, \dots$ **do**
    datastructure $\leftarrow$ `processElement`(datastructure, x_t)

Algorithm 4 TopK(datastructure, k)

// Print the k largest elements that have been processed so far.

Theory Task T3. **/ 8 P**

Consider the following procedure that takes two sorted (in ascending order) integer arrays $A = A[1, \ldots, n]$ and $B = B[1, \ldots, m]$ as input and merges them:

```
procedure MERGE(A, B)
    M[1], ..., M[n], M[n+1], ..., M[n+m] ← 0, ..., 0
    i, j ← 1, 1
    for 1 ≤ k ≤ n + m do
        if i ≤ n and (j > m or A[i] ≤ B[j]) then
            M[k] ← A[i]
            i ← i + 1
        else if j ≤ m and (i > n or A[i] > B[j]) then
            M[k] ← B[j]
            j ← j + 1
    return M
```

Note: if $j > m$, then ($j > m$ or $A[i] \leq B[j]$) always evaluates to true, no matter whether $B[j]$ is well-defined, and, similarly for ($i > n$ or $A[i] > B[j]$).

Recall that an array C is the result of merging the sorted arrays A and B if C is sorted and contains the same values as A and B (preserving duplicates). For example, if we merge $[1, 2, 4, 6]$ and $[1, 3, 6, 7, 10]$, we obtain $[1, 1, 2, 3, 4, 6, 6, 7, 10]$.

Show that the pseudocode above satisfies the following loop invariant INV(ℓ) for $1 \leq \ell \leq n + m$: After ℓ iterations of the for-loop,

1) the subarray $M[1, \ldots, \ell]$ is the result of merging the subarrays $A[1, \ldots, i-1]$ and $B[1, \ldots, j-1]$.
2) all values in $M[1, \ldots, \ell]$ are at most as large as any of the values in $A[i, \ldots, n]$ and $B[j, \ldots, m]$.

Specifically, prove the following 3 assertions.

i) INV(1) holds.

ii) If INV(ℓ) holds, then INV($\ell + 1$) holds (for all $1 \leq \ell < n + m$).

iii) INV($n + m$) implies that the algorithm correctly merges A and B.

Finally, state the running time of the procedure Merge described above in Θ-notation in terms of n and m (as simplified as possible).

Proof of i).

Proof of ii).

Proof of iii).

Running time:

/ 12 P

Theory Task T4.

Assume that there are n towns $T_1, \ldots, T_n$ in the country Examistan. For each pair of distinct towns T_i and T_j, there is exactly one road from T_i to T_j. All of the roads in Examistan are one-way. This implies that there is always a road from T_i to T_j and another road from T_j to T_i. Each road has a nonnegative integer cost that you need to pay to use this road.

For simplicity you can assume that each town T_i is represented by its index i.

/ 1 P

a) Model the n towns, the roads and their costs as a directed weighted graph: give a precise description of the vertices, edges and the weights of the edges of the graph $G = (V, E, w)$ involved (if possible, in words and not formal). What are $|V|$ and $|E|$ in terms of n?

In the following subtasks b) and c), you can assume that the directed graph in a) is represented by a data structure that allows you to traverse the direct successors and direct predecessors of a vertex u in time $\mathcal{O}(\deg_+(u))$ and $\mathcal{O}(\deg_-(u))$ respectively, where $\deg_-(u)$ is the in-degree of vertex u and $\deg_+(u)$ is the out-degree of vertex u.

/ 6 P

b) Due to the epidemiological situation in Examistan, the authorities decided to reduce the number of trips between different towns. Now the only way to get from one town to another is to use the roads. Moreover, if you want to travel from town T_i to the other town T_j, you must visit a test center during your trip (in T_i or T_j or elsewhere with a detour). Since test centers are expensive, there are only $k < n$ of them, and they are located only in the first k towns $T_1, \ldots, T_k$ (i.e., one test center in each of these towns).

Assume that you need to fill the table of minimal costs required to travel between all pairs of towns, which takes into account the new rules of travelling. Provide an as efficient as possible algorithm that takes as input a graph G from task a) and a number k, and outputs a table C such that $C[i][j]$ is the minimal total cost of roads that one can use to get from T_i to T_j while also visiting a test center. You can assume that for all $1 \leq i \leq n$, $C[i][i] = 0$.

What is the running time of your algorithm in concise Θ-notation in terms of n and k? Justify your answer.

/ 5 P

c) Now suppose that before building the test centers in towns $T_1, \ldots, T_k$, the authorities had made the roads between all different T_i and T_j among $T_1, \ldots, T_k$ free of charge (i.e. their cost is now 0). Solve the problem from subtask b) assuming this condition.

That is, provide an as efficient as possible algorithm that takes as input a graph G from task a) and a number k, and outputs a table C such that $C[i][j]$ is the minimal total cost of roads that one should use to get from T_i to T_j with visiting a test center.

What is the running time of your algorithm in concise Θ-notation in terms of n and k? Justify your answer.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.

Extra space. Please indicate clearly to which task your notes belong. Please cross out all notes that you do not want to be graded.