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

Exam

Algorithmen und Datenstrukturen

January 21, 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!

	T1 (26P)	T2 (12P)	T3 (9P)	T4 (13P)	P1 (24P)	P2 (16P)	**Σ** (100P)
Score							
Corrected by							

Theory Task T1.

/ 27 P

In this problem, you have to provide **solutions only**. You do not need to justify your answer.

/ 5 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.

Claim	**true**	**false**
$5n^{2.5} + 2n^2 + n \leq \mathcal{O}(n^3)$	☐	☐
$\sqrt{n} \geq \Omega(\frac{n}{\log n})$	☐	☐
$\log_4 n^4 = \Theta(\log_6 n^6)$	☐	☐
$\sum_{i=1}^{n} \log_2 i = \Theta(n \log n)$	☐	☐
$\sum_{i=1}^{n} \sqrt{i} \cdot \log_2^2 i \geq \Omega(n\sqrt{n} \log^2 n)$	☐	☐

/ 4 P

b) *Graph quiz:* For each of the following claims, state whether it is true in general, i.e., whether it is true for all graphs.

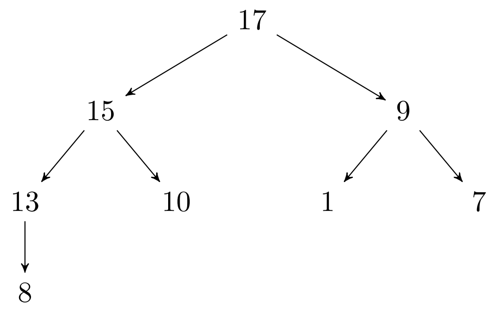
You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total.

Claim	**true**	**false**
A connected graph is called a tree.	☐	☐
If a tree contains at least two vertices, then any longest path in this tree has leaves as endpoints.	☐	☐
A bipartite graph can be coloured with two colours.	☐	☐
Any directed acyclic graph has a vertex with in-degree 0.	☐	☐

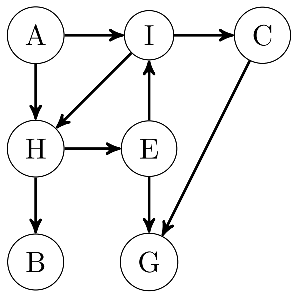
/ 2 P c) *Max-Heaps:*

i) Draw a Max-Heap that contains the keys $3, 6, 1, 5, 7, 2$.

ii) Draw the Max-Heap obtained from the following Max-Heap by performing the operation `DELETE-MAX` once.



/ 2 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 A. When processing the neighbors of a vertex, process them in alphabetical order.

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

/ 3 P

e) *Shortest paths:* Given is a weighted directed connected graph $G = (V, E, w)$ (edge weights w might be negative), where $V = \{1, \ldots, n\}$. The goal is to determine whether G contains a cycle of negative weight, and if not, you need to fill a table of size $n \times n$ such that the entry (i, j) contains the length of the shortest path between i and j in G (for all $i, j \in V$).

i) Name an algorithm that was discussed in the lecture which solves this problem. If several such algorithms exist, it is enough to name one of them.

ii) State the running time of this algorithm in $\mathcal{O}$-notation in terms of $|V|$ and $|E|$ (as tight and simplified as possible).

iii) Briefly describe how this algorithm reads out whether G contains a cycle of negative weight.

/ 5 P

f) *Graph data structures:*

Consider the following two data structures for storing a graph G with n vertices and m edges:

i) Adjacency matrix.

ii) Adjacency list.

For each of the above data structures, what is the required memory in terms of n and m (in $\mathcal{O}$-notation, as tight and simplified as possible)?

Consider the following tasks:

Task 1: Given is a vertex $v \in V$, find $\deg(v)$.

Task 2: Given is a vertex $v \in V$, find a neighbour of v (if a neighbour exists).

Task 3: Given are two vertices $u, v \in V$, determine whether u and v are adjacent.

Task 4: Given are two vertices $u, v \in V$ with $u \neq v$, insert an edge $\{u, v\}$ into the graph if it does not exist yet. Otherwise do nothing.

Fill the following table with the running times that are required to solve these tasks. Please state your answers in $\mathcal{O}$-notation in terms of n, m and vertex degrees ($\deg(v)$ for $v \in V$), as tight and simplified as possible.

	Adjacency matrix	Adjacency list
Task 1		
Task 2		
Task 3		
Task 4		

/ 5 P

g) *Computing Powers:*

Provide an algorithm that takes two positive integers a and n as input, and outputs a^n.

You can use addition, subtraction, multiplication and division of integers as basic operations without further explanation. You do not need to proof correctness or analyse the running time of your algorithm. However, for full points the number of basic operations that your algorithm uses needs to be $\mathcal{O}(\log n)$.

Theory Task T2. / 12 P

In this part, you should **justify your answers briefly**, e.g., by sketching the derivation.

/ 4 P

a) *Counting loop iterations:* For the following code snippet count how many times the function f is called. Report the number of calls in $\mathcal{O}$-notation (as tight and simplified as possible).

i) Snippet 1:

Algorithm 1

```
for $j = 1, \dots, n$ do
    for $k = j, \dots, n$ do
        $f()$
```

ii) Snippet 2:

Algorithm 2

```
for $j = 1, \dots, n$ do
    for $k = 1, 2, 3, 4, 5, \dots, 2^j$ do
        for $l = 1, \dots, 42$ do
            $f()$
```

/ 2 P

b) *Recurrence relations:* Recall the 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 $n = 2^k$,*

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

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

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

Consider the following recursive function that takes as an input a natural number m that is a power of two (that is, $m = 2^k$ for some nonnegative integer k).

```
**Algorithm 3** $g(m)$
if $m > 1$ then
    $g(m/2)$
    $g(m/2)$
    $g(m/2)$
    $g(m/2)$
    for $i = 1, \dots, m^2$ do
        $f()$
else
    $f()$
```

Let $T(m)$ be the number of calls of the function f in $g(m)$.

i) Give a recursive formula for $T(m)$.

ii) Write $T(m)$ in $\mathcal{O}$-notation in terms of m (as tight and simplified as possible).

/ 2 P c) *Minimum spanning trees:* Given is a weighted undirected connected graph $G = (V, E, w)$ with positive and pairwise different edge weights $w\colon E \to \mathbb{R}^+$. Suppose that there exists a vertex $v \in V$ of degree 4 with incident edges e_1, e_2, e_3, e_4 such that $w(e_1) = 2, w(e_2) = 3, w(e_3) = 4, w(e_4) = 7$. Further, suppose that the weight of the minimum spanning tree of G is 72.

What are the possible weights of a minimum spanning tree if we decrease the weight of the edge e_1 from 2 to 1? The weights of all other edges remain unchanged. Justify your answer.

/ 4 P d) *Induction:* Prove by mathematical induction that for any positive integer n,

$$\sum_{k=1}^{n} \frac{1}{k^2} \leq 2 - \frac{1}{n}\,.$$

Theory Task T3.

/ 1+5+1+2 = 9 P

Given is a weighted directed acyclic graph $G = (V, E, w)$ with positive edge weights $w\colon E \to \mathbb{R}^+$, where $V = \{1, \dots, n\}$. The goal is to find the length of a **longest** path in G.

Let's fix some topological ordering of G and consider the array top$[1, \dots, n]$ such that top$[i]$ is the i-th vertex in this topological ordering.

Consider the following pseudocode:

```
procedure FIND-LENGTH-OF-LONGEST-PATH($G$, top)
    $L[1], L[2], \dots, L[n] \leftarrow 0, 0, \dots, 0$
    for $1 \le i \le n$ do
        $v \leftarrow \text{top}[i]$
        $L[v] \leftarrow \max_{(u,v) \in E} \{L[u] + w(u, v)\}$
    return $\max_{1 \le i \le n} L[i]$
```

Here we assume that the maximum over the empty set is 0.

Show that the pseudocode above satisfies the following loop invariant INV(k) for $1 \le k \le n$: After k iterations of the for-loop, $L[\text{top}[j]]$ contains the length of a longest path that ends with top$[j]$ for all $1 \le j \le k$.

Specifically, prove the following 3 assertions.

i) INV(1) holds.

ii) If INV(k) holds, then INV($k + 1$) holds (for all $1 \le k < n$).

iii) INV(n) implies that the algorithm correctly computes the length of a longest path.

Finally, state the running time of the algorithm described above in Θ-notation in terms of $|V|$ and $|E|$ (as simplified as possible). Justify your answer.

Proof of i).

Proof of ii).

Proof of iii).

Running time with justification:

Theory Task T4. **/ 2 + 2 + 9 = 13 P**

Suppose that there are n airports in the country Examistan. Between some of them there are direct flights. For each airport there exists at least one direct flight from this airport to some other airport. Totally there are m different direct flights between the airports of Examistan.

For each direct flight you know its cost, which is a positive integer.

You can assume that each airport is represented by its number, i.e., the set of airports is $\{1, \ldots, n\}$.

/ 2 P a) Model these airports, direct flights and their costs as a directed graph: give a precise description of the vertices, the edges and the weights of the edges of the graph $G = (V, E, w)$ involved (if possible, in words and not formal).

In points b) and c) you can assume that the directed graph is represented by a data structure that allows you to traverse the direct predecessors and direct successors 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.

/ 2 P

b) Suppose that you are at the airport S and you want to fill the array d of minimal travelling costs to each airport. That is, for each airport A, $d[A]$ is a minimal cost that you must pay to travel from S to A.

Name the most efficient algorithm that was discussed in lectures which solves the corresponding graph problem. If several such algorithms were described in lectures (with the same running time), it is enough to name one of them. State the running time of this algorithm in $\mathcal{O}$-notation in terms of $|V|$ and $|E|$ (as tight and simplified as possible).

/ 9 P

c) Now you want to know *how many* optimal routes there are from airport S to a given airport T. In other words, if $c_{\min}$ is the minimal cost to travel from S to T then you want to compute *the number of routes from S to T of cost* $c_{\min}$.

Assume that the array d from b) is already filled. Provide an as efficient as possible *dynamic programming* algorithm that takes as input the graph G from task a), the array d from point b) and the airports S and T, and outputs the number of routes from S to T of minimal cost.

Address the following aspects in your solution and state the running time of your algorithm:

1) *Definition of the DP table: What are the dimensions of the table $DP[\dots]$?* What is the meaning of each entry?
2) *Computation of an entry*: How can an entry be computed from the values of other entries? Specify the base cases, i.e., the entries that do not depend on others.
3) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
4) *Extracting the solution*: How can the final solution be extracted once the table has been filled?

Size of the DP table / Number of entries: ____________________

Meaning of a table entry:

$DP[\dots\dots\dots\dots]$: ____

____

Computation of an entry (initialization and recursion):

Scheme continues on the next page.

Order of computation:

Extracting the result:

Running time in concise Θ-notation in terms of $|V|$ and $|E|$. Justify your answer.

Hint: *Note that the array d is a part of the input, so you don't need to include the time that is required to fill this array to the running time here.*

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.

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.