Eidgenössische
Technische Hochschule
Zürich

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

Department Informatik
Markus Püschel
Johannes Lengler
Chris Wendler

David Steurer
Gleb Novikov
Ulysse Schaller

Exam

Algorithmen und Datenstrukturen

January 26, 2021

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 (20P)	T2 (19P)	T3 (9P)	T4 (12P)	Prog. (40P)	Σ (100P)
Score						
Corrected by						

Theory Task T1. / 20 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.

Assume $n \geq 2$.

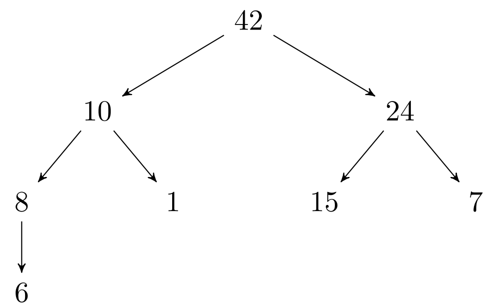
Claim	**true**	**false**
$n^2 + 10n - 23 \geq \Omega(n^{2.5})$	☐	☐
$\sqrt[3]{n} \leq \mathcal{O}(\frac{\sqrt{n}}{\log n})$	☐	☐
$\log_3(n^4) = \Theta(\log_6(n^2))$	☐	☐
$\sum_{i=1}^{n} \sqrt{i} = \Theta(n^{1.5})$	☐	☐
$\sum_{i=1}^{n} i! \geq \Omega(n \cdot n!)$	☐	☐

/ 2 P

b) *Max-Heaps:*

i) Draw a Max-Heap that contains the keys $8, 4, 2, 3, 5, 7$ (note that several solutions are possible here).

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



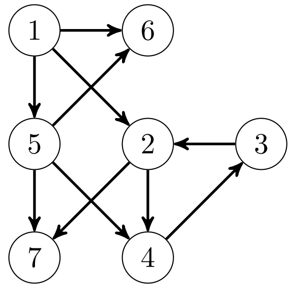
/ 5 P

c) *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.

Claim	**true**	**false**
The topological ordering of a directed acyclic graph is unique.	☐	☐
For all $n \in \mathbb{N}$, there exists a directed acyclic graph on n vertices with $\binom{n}{2}$ edges.	☐	☐
Let $v \in V$ be a vertex of an undirected graph $G = (V, E)$ with adjacency matrix A. It takes time $\Theta(1 + \deg(v))$ to compute $\deg(v)$ from A.	☐	☐
If every vertex of an undirected graph G has even degree, then G has an Eulerian walk.	☐	☐
In order to run Dijkstra's algorithm on a directed graph G, you first need to have a topological ordering of G.	☐	☐

/ 2 P

d) *Depth-first search / Breadth-first search:* Consider the following directed graph:

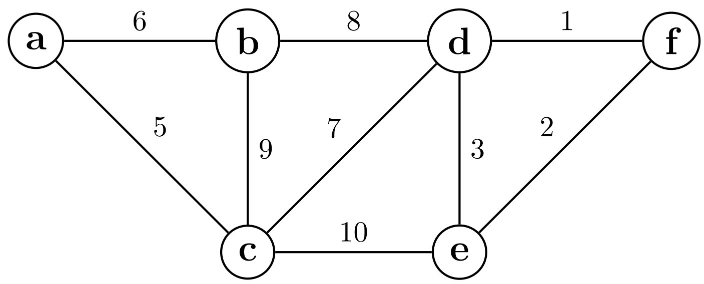


i) Draw the depth-first tree resulting from a depth-first search starting from vertex 1. Process the neighbors of a vertex in increasing order.

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

/ 1 P

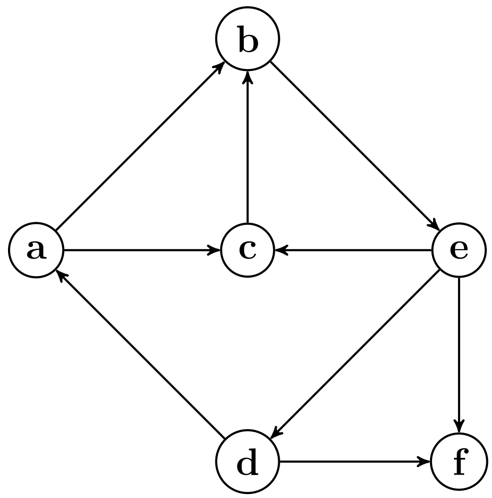
e) *Minimum Spanning Tree:* Consider the following graph:



Highlight the edges that are part of the minimum spanning tree.

/ 2 P

f) *Topological sorting:* Consider the following directed graph G:



i) Remove the smallest possible number of edges from G such that a topological ordering of its vertices exists.

ii) Compute a topological ordering of the vertices of your modified graph.

/ 3 P

g) *Sorting algorithms:*

i) Consider the sequence $6, 5, 4, 1, 2, 3$. How many swaps does `Bubble Sort` perform to sort this sequence? *Give the exact number of swaps required.*

ii) Consider the sequence $6, 5, 4, 1, 2, 3$. How many swaps does `Selection Sort` perform to sort this sequence? *Give the exact number of swaps required.*

iii) Let $n \in \mathbb{N}$ be an even number and consider the sequence with the following structure:

$$2, 1, 4, 3, 6, 5, \ldots, n, n-1.$$

How many swaps does `Insertion Sort` perform to sort this sequence? *Give the exact number, not just the asymptotics.*

Theory Task T2. / 19 P

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

/ 3 P a) *Induction:* For the following task you may use the identity

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \quad \text{for all } 1 \leq k \leq n$$

without proof. Show by mathematical induction that for any integer $n \geq 1$,

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

/ 2 P b) *Recurrence relations:*

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

Theorem 1 (Master theorem) *Let $a, C > 0$ and $b \geq 0$ be constants and $T : \mathbb{N} \to \mathbb{R}^+$ a non-decreasing function such that for all $k \in \mathbb{N}$ and $n = 2^k$,*

$$T(n) \leq aT(n/2) + Cn^b.$$

Then

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

Consider the following recursive function that takes as an input a positive integer m that is a power of two (that is, $m = 2^k$ for some integer $k \geq 0$).

Algorithm 1 $g(m)$

```
if m > 1 then
    g(m/2)
    g(m/2)
    for i = 1, ..., 6⌊√m⌋ 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 $T(m)$. Don't forget to provide the base case as well.

ii) Determine $T(m)$ in $\mathcal{O}$-notation. Your answer should be as tight as possible.

/ 4 P c) *Graph connectivity*

Recall the following two definitions from the exercises.

Definition 1 *A vertex v in a connected graph is called a* cut vertex *if the subgraph obtained by removing v (and all its incident edges) is disconnected.*

Definition 2 *An edge e in a connected graph is called a* cut edge *if the subgraph obtained by removing e (but keeping all the vertices) is disconnected.*

In the following, we always assume that the original graph is connected. Prove or find a counterexample to the following statements:

i) If a vertex v is part of a cycle, then it is not a cut vertex.

ii) If a vertex v is not a cut vertex, then v must be part of a cycle.

iii) If an edge e is part of a cycle (that is, e connects two consecutive vertices in a cycle), then it is not a cut edge.

iv) If v is a cut vertex and e is an edge incident to v, then e is a cut edge.

/ 5 P d) *Insertion sort invariant*

Let $A[0, \ldots, n-1]$ be an integer array of size n. Consider the following implementation of insertion sort:

Algorithm 2 `InsertionSort`(A)

for $i = 1 \ldots n-1$ **do**
  Find the smallest index $j \in \{0, \ldots, i\}$ such that $A[i] \leq A[j]$.
  Shift the subarray $A[j, \ldots, i-1]$ by one to the right, and move the element $A[i]$ to position j.

i) Formulate an invariant $INV(i)$ that holds after the ith iteration of the `for`-loop (the iteration with $i = 1$ is the first iteration).

ii) Use this invariant to prove correctness of the algorithm `InsertionSort`.

1. Show that the invariant holds at the beginning (base case).

2. Let $1 \leq i \leq n-2$. Show that if $INV(i)$ holds after the ith iteration of the `for`-loop, then $INV(i+1)$ holds after the $(i+1)$st iteration (induction step).

3. Show that if $INV(n-1)$ holds at the end of the algorithm, then the array A is sorted.

/ 5 P

e) *Finding a cheap cycle*

Let $G = (V, E)$ be a weighted undirected graph, where all edge weights are positive. Provide an efficient algorithm that, given an edge $e \in E$, outputs the weight of the cheapest cycle (that is, the cycle of smallest total weight) that contains e, and outputs ∞ if e is not contained in any cycle. Give the running time of your algorithm in terms of $|V|$ and $|E|$. In order to get full points, your algorithm should run in time $\mathcal{O}((|V| + |E|) \log |V|)$

You do not need to write a proof of correctness or a runtime analysis. If you use algorithms known from the lecture as sub-routines, you do not need to re-discuss how they work.

Theory Task T3. / 9 P

You are given an array of n natural numbers $a_1, \ldots, a_n \in \mathbb{N}$ summing to $A := \sum_{i=1}^{n} a_i$, which is a multiple of 3. You want to determine whether it is possible to partition $\{1, \ldots, n\}$ into three disjoint subsets I, J, K such that the corresponding elements of the array yield the same sum, i.e.

$$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_k = \frac{A}{3}.$$

Note that I, J, K form a partition of $\{1, \ldots, n\}$ if and only if $I \cap J = I \cap K = J \cap K = \emptyset$ and $I \cup J \cup K = \{1, \ldots, n\}$.

For example, the answer for the input $[2, 4, 8, 1, 4, 5, 3]$ is *yes*, because there is the partition $\{3, 4\}$, $\{2, 6\}$, $\{1, 5, 7\}$ (corresponding to the subarrays $[8, 1]$, $[4, 5]$, $[2, 4, 3]$, which are all summing to 9). On the other hand, the answer for the input $[3, 2, 5, 2]$ is *no*.

Provide a *dynamic programming* algorithm that determines whether such a partition exists. Your algorithm should have an $\mathcal{O}(nA^2)$ runtime to get full points. Address the following aspects in your solution:

1) *Definition of the DP table*: What are the dimensions of the table $DP[\ldots]$? 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?
5) *Running time*: What is the running time of your algorithm? Provide it in Θ-notation in terms of n and A, and justify your answer.

Size of the DP table / Number of entries: ______________________

Meaning of a table entry:

Scheme continues on the next page.

Computation of an entry (initialization and recursion):

Order of computation:

Extracting the result:

Running time:

Theory Task T4. / 12 P

/ 3 P

a) Consider the following problem. The Swiss government is negotiating a deal with Elon Musk to build a tunnel system between all major Swiss cities. They put their faith into you and consult you. They present you with a map of Switzerland. For each pair of cities it depicts the cost of building a bidirectional tunnel between them. The Swiss government asks you to determine the cheapest possible tunnel system such that every city is reachable from every other city using the tunnel network (possibly by a tour that visits other cities on the way).

i) Model the problem as a graph problem. Describe the set of vertices, the set of edges and the weights in words. What is the corresponding graph problem?

ii) Use an algorithm from the lecture to solve the graph problem. State the name of the algorithm and its running time in terms of $|V|$ and $|E|$ in Θ-notation.

/ 9 P

b) Now, the Swiss tunneling society contacts the government and proposes to build the tunnel between Basel and Geneva for half of Musk's cost. Thus, the government contacts you again. They want you to solve the following problem: Given the solution of the old problem in a) and an edge for which the cost is divided by two, design an algorithm that updates the solution such that the new edge cost is taken into account. *In order to achieve full points, your algorithm must run in time $\mathcal{O}(|V|)$.*

Hint: You are only allowed to use the *solution* from a), i.e. the set of tunnels in the chosen tunnel system. You are not allowed to use any intermediate computation results from your algorithm in a).

i) Describe your algorithm (for example, via pseudocode). A high-level description is enough.

ii) Prove the correctness of your algorithm and show that it runs in time $\mathcal{O}(|V|)$.

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.