Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique f´ed´erale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik

David Mihnea-Stefan

Exam Preparation**

Algorithmen und Datenstrukturen

November 11, 2024

DO NOT OPEN!

Student number: ______________________

Seat number: ______________________

Good luck!

	T1 (31P)	T2 (13P)	T3 (9P)	T4 (7P)	**Σ** (60P)
Score					
Corrected by					

***The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

***The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

Theory Task T1. / 31 P

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

/ 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. Assume $n \geq 4$.

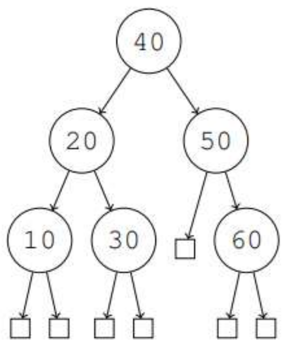
Claim	true	false
$e^n - n^{100} - 22 \geq \Omega(e^n)$	☐	☐
$\ln\left(\prod_{i=1}^{n} i^n\right) \leq O(n^{n+1})$	☐	☐
$\sum_{i=1}^{n} (n-i)^3 \leq O(n^{3.5})$	☐	☐
$\sqrt{n} \geq \Omega\left(\frac{n}{\log(n)}\right)$	☐	☐
$n * \log(n^{n^2} - e^n + 12\log(n^{n^2}) = \Theta(n^3 * \log n)$	☐	☐
$for\ T(1) = 1\ and\ T(n) = T(n-1) + n => T(n) = \Theta(\mathrm{n}^2)$	☐	☐

/ 8 P

b). Search trees

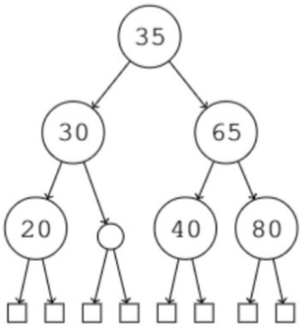
i). Draw a binary search tree of maximum possible depth that contains exactly the numbers $\{1, 4, 6, 9, 21, 14\}$. The depth of the tree is the longest path from the root to any of its leaves.

ii). Draw the binary search tree obtained from the following tree by performing the two operations INSERT(25) and INSERT(44), in that order



***The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

iii). Argue if the next binary search tree is an AVL tree or not:



iv). AVL trees – true / false

Claim	true	false
The time complexity for searching, inserting, and deleting a node in an AVL tree is always O(n) in the worst-case scenario.	☐	☐
After deleting a node in an AVL tree, rebalancing the tree may require up to two rotations to restore the AVL property.	☐	☐

/ 6 P *c). Heaps*

i). Draw the (max-) heap obtained by inserting the operations INSERT(22), INSERT(12), INSERT(20), INSERT(41), INSERT(30), INSERT(50)

ii). For the Max-Heap from i)., apply twice the DELETE_MAX operation

iii). (max-) Heap - true/ false:

Claim	true	false
A binary heap can have at most two children per node and is always a balanced binary tree.	☐	☐
*After building a **max-heap** from an unordered array using the heapify process, the resulting heap is guaranteed to be a complete binary tree*	☐	☐

/ **11P** *d). Sorting algorithms*

i). Quiz:

Claim	true	false
*QuickSort has a worst-case time complexity of O(n*log n).*	☐	☐
Bubble Sort is more efficient than Insertion Sort for nearly sorted arrays.	☐	☐
Selection Sort has a time complexity of $O(n^2)$ *for both the best and worst cases*	☐	☐
The best possible time complexity for a comparison-based sorting algorithm is O(n log n)	☐	☐
Merge Sort requires O(log n) extra space	☐	☐
Selection Sort performs fewer swaps than Bubble Sort	☐	☐
HeapSort maintains the same invariant as Insertion Sort, where the largest (or smallest) element is moved to its correct position in each iteration.	☐	☐
All comparison-based sorting algorithms have a worst-case time complexity of at most $O(n^2)$	☐	☐

ii). Number of operations - For each of the algorithms listed below, specify the number of swaps and comparisons using **Theta (Θ)** when possible, otherwise **Big-O (O)** notation.

Algorithm	Swaps	Comparison
Bubble Sort	$O(n^2)$	$\Theta(n^2)$
Selection Sort		
Insertion Sort		

***The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

Theory Task T2.

/ 13 P

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

/ 4 P

a). Explain briefly how **QuickSort** works. In your answer, consider the following:

- *What is the pivot?*
- *What property do the elements to the left and to the right of it have?*
- *Why can't we make a clear statement about its run-time?*

/ 4 P *b). Counting iterations:* For the following code snippets, derive an asymptotic bound for the number of times f is called. Simplify the expression as much as possible and state it in Θ-notation as concisely as possible.

i) Snippet 1:

Algorithm 1

for $i = 1, \ldots, n$ **do**
  for $j = i, i + 1, \ldots, n^2$ **do**
    $f()$
    $f()$

ii). Snippet 2:

Algorithm 2 T(n)

if n >= *1* **do**
  *P = 10 * n^2 + 4 * n*
  for $i = 1, \ldots, P$ **do**
    f()
  for $k = 1, \ldots, 4$ **do**
    T(n / 2)

/ 5P

b). Induction: Prove the following inequality by induction: for every integer n ≥ 2,

$$\sum_{i=1}^{n} \frac{1}{n+i} > \frac{13}{24}$$

***The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

**The material serves as a supporting resource for the preparation for the AnD exam and was created in the capacity of a Teaching Assistant for this course. The content has not been validated by the course staff team.*

Theory Task T3. / 9 P

You are given an array **A** = [a_1, a_2, ..., a_n] of non-negative integers. The array is called **complete subset sum** if for every element a_i (where $2 \leq i \leq n$), there exists a subset **S** $\subseteq \{1, 2, ..., i - 1\}$ such that a_i can be represented as:

$$a_i = \sum_{j \in S} a_j$$

In other words, each element from the second onward must be the sum of a subset of the preceding elements in the array.

For example:

- The array [3, 3, 6, 9, 12] is **complete subset sum**, because:
  - 3 = 3
  - 6 = 3 + 3
  - 9 = 3 + 6
  - 12 = 3 + 9 *(or 3 + 3 + 6)*
- The array [3, 5, 8, 14] is **not complete subset sum**, because 14 cannot be formed by a subset of {3, 5, 8}.

Your task is to design a dynamic programming algorithm that, given an array **A** of length **n**, returns **True** if the array is **complete subset sum** and **False** otherwise. Your algorithm should have a time complexity of **O(n · max(A))**.

In your solution, address the following aspects:

1. *Dimensions of the DP table*: What are the dimensions of the *DP* table?
2. *Subproblems*: What is the meaning of each entry?
3. *Recursion*: How can an entry of the table be computed from previous entries? Justify why your recurrence relation is correct. Specify the base cases of the recursion, i.e., the cases that do not depend on others.
4. *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?
5. *Extracting the solution*: How can the solution be extracted once the table has been filled?
6. *Running time*: What is the running time of your solution?

Theory Task T4.

/ 7 P

You are given an array of n integers a[1], a[2], ..., a[n]. A "balance point" in the array is an index i ($1 \leq i \leq n$) such that the sum of all elements to the left of i (i.e., a[1] + a[2] + ... + a[i-1]) is equal to the sum of all elements to the right of i (i.e., a[i+1] + a[i+2] + ... + a[n]).

For example, in the array [1, 2, 3, 11, 2, 3, 1], the balance point is at index 4, because the sum of the values on the left (1 + 2 + 3) is equal to the sum of the values on the right (2 + 3 + 1).

a). Describe an algorithm that finds the index of a balance point in the array. If no such balance point exists, return -1.

You should explain the idea behind your approach and describe how it could be implemented in O(n) time complexity.

b) Briefly explain why we cannot find an algorithm with a time complexity better than O(n)

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.

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.