Stockholm University
Department of Mathematics
Lecturer: Marc Hellmuth
Tutors: Anna Lindeberg, Patricia Alexandra Ebert, David Sermoneta, Marcus Hultvall Hultin

3. Exercise "Data Structures and Algorithms (DA 4006)"

Problem 1: 8+2+3p=13p

Given is the following pseudo-code of a divide-and-conquer approach to find the maximum element in an array A of integers (the first element in A is $A[1]$).

```
function find_max(array of integers A)
  IF (A.length = 1) RETURN A[1]
  k = ⌊A.length/2⌋
  print(A[1..k])
  left = find_max(A[1..k])
  print(A[k+1..A.length])
  right = find_max(A[k+1..A.length])
  print("LR:", left, right)
  RETURN max{left, right}
```

You can assume that all basic operations like comparisons, mathematical operations, `RETURN` and `print`("LR:", *left*, *right*) take $\Theta(1)$ time. The `print`($A[p..q]$) command produces the following output:

$$A[p] \dots A[q],$$

which represents all the elements in $A[p..q]$ printed step-by-step in a single line, beginning with entry $A[p]$, followed by $A[p+1]$, and continuing until entry $A[q]$. The command `print`("LR:", *left*, *right*) outputs "LR:"followed by the current values of *left* and *right*.

(a) Apply this algorithms on the array $A = [5, 9, 3, 8, 2]$ and provide the output of the `print` command in each of the single recursive calls in the order they appear.

(b) Use the Master Theorem to determine the runtime $T(n)$ of `function` **find_max**(A) for an array A of length n. In particular, specify a, b, and d (explain how you come up with these values!) as well as the function f such that $T(n) \in \Theta(f)$.

(c) Shortly explain, if there are parts that can be removed from the `function` **find_max**(A) such that the resulting algorithm still finds the maximum element in an array A of length n and such that $T(n) \in \Theta(n)$?

Problem 2: 3+2=5p

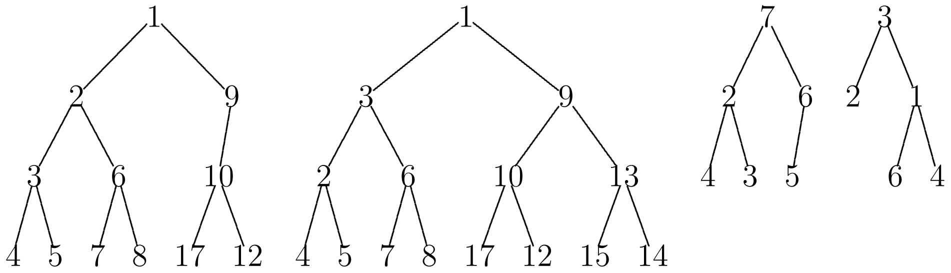
(a) Draw a binary tree T with 6 vertices whose inorder and preorder traversal yields the following sequence of vertices

Inorder:	D	B	E	A	F	C
Preorder:	A	B	D	E	C	F

(b) Explain if this tree T is uniquely determined when only the preorder traversal in (a) is provided.

Problem 3: 10p

Given are the following binary trees:



(i) (ii) (iii) (iv)

For each of the trees (i)-(iv), specify its height, the depth of vertex 3 and indicate (yes/no) whether it has the following properties: fully-binary, nearly-complete, complete. Use the definitions from the lecture.

Problem 4: 6p

Given is the following array

$$A = [6, 2, 3]$$

where the first entry is $A[1]$. Apply `MERGE_SORT`$(A, 1, 3)$ according to the lecture. In the order it appears, provide for each call of `MERGE_SORT`(A, m, m') the values m and m' (in the lecture, we had $(m, m') = (p, q)$ or $(m, m') = (q+1, r)$) and for each call of `MERGE`(A, p, q, r) the values p, q, r as well as the resulting sub-array $A[p..r]$ after `MERGE`(A, p, q, r) has been called.

Problem 5: 1+4.5+2.5 = 7p

Given is the following array

$$A = [1, 2, 3, 4, 5]$$

where the first entry is $A[1]$.

(a) Draw the array A as a binary heap tree.

(b) Use `Build-Max-Heap(`A`)` according to the lecture to transform A into a max-heap. Specify the indices i and keys $A[i]$ for which the max-heap property is violated within the respective calls of `Max-Heapify`(..). Additionally, provide the heap (binary tree) after each adjustment (e.g. as a drawing).

(c) Sort A using `Heapsort`(A) according to the lecture. Provide the max-heap after each iteration of the `FOR`-loop in `Heapsort` (e.g. as a drawing of the respective tree).

Problem 6: Bonus Exercise 4p

Fake-Coin Detection revisited. Cool! Instead of a digital scale as in Ex1(5), this time you have a fashionable spring scale. Moreover, you have $n > 1$ identical-looking coins: $n - 1$ of them are genuine with a known weight w, and one of them – of an unknown weight different from w – is counterfeit. Design an algorithm that finds the fake coin using a minimum number of weighings on a spring scale. Explain how and why your algorithm works. You can assume that the spring scale indicates the exact weight of the coins being weighed and that you can place any number between $0, 1, \ldots, n$ of coins on the scale.

Bonus problems can be used to earn a certain amount of extra points counted for the final exam.

Deadline: Thursday - April 30, 2026