MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Marc Hellmuth

Tentamensskrivning i
Datastrukturer och algoritmer
7.5 hp
2025-06-03

General Information

- Time 8:00 - 12:00
- No books, internet, smartphones, notebooks or any extra material are allowed to be used.
- In total, you can get 63 points and you need to get at least 31.5 points to pass the exam.
- Many problems just require simple answer like 'yes' or 'no'; or $T(n) \in \Theta(f(n))$; or something similar. To answer these problems, **do not** waste your time by writing more than required to save time for answering other exercises!
- Do not provide ambiguous (or multiple) solutions. Do not change the problem statement to fit your solution.
- Make sure not to overlook any of the **16** problems.
- On average, you have 15 minutes for each of the 16 problems. However, many of them can be solved much faster. The level of difficulty or time needed to solve the respective problem is indicated using 1, 2, or 3 stars "$\star$":

  easy / quick = ($\star$)
  medium / moderate time = ($\star\star$)
  difficult / more time-consuming = ($\star\star\star$)

Problem 1 (*Basic Questions* ($\star$)) 1+1+1+1=4p

Answer the following questions.

(a) Can an algorithm whose runtime is $\Theta(n)$ (where n is the size of the input) be a *comparison sort* algorithm?

(b) What does it mean that a sorting algorithm is *in place*?

(c) How many byte long is the following binary string: 0110001001001001

(d) Sort the following algorithms in terms of their worst-case runtime, starting with fastest one:
`Insertion Sort`, `Heapsort`, `Quicksort`.

Problem 2 (*Algorithm Design* ($\star\star\star$)) 5p

Design an algorithm that takes as input a binary number B of length n, given as an array $B[0..n-1]$, where $B[i]$ represents the bit corresponding to 2^i. The algorithm should return true precisely if the following two conditions are satisfied:

1. The binary number B represents an even integer (base_10 number).
2. The binary number B is a palindrome - that is, it reads the same forwards and backwards. For example, 11011 or 000000 are both palindroms.

In all other cases, your algorithm should return false. You are allowed to use any type of variable assignments and the return command. In addition, only comparison-operarors "$<$", "$>$", "$=$" as well as `while`-loops, `if-else`-conditions and the arithmetic operations "$+$" and "$-$" are allowed.

Demonstrate your algorithm step-by-step using the two inputs: $B = [1, 0, 1, 0, 1]$ and $B = [0, 0, 1, 0, 0]$. A correctness proof is *not* required.

Problem 3 (*O-, Ω- and Θ-Notation* ($\star\star$)) 3+6=9p

(a) Prove in detail that $T(n) = \frac{n^2-1}{n-1} \in \Theta(n)$.
In detail means that arguments "as shown in the lecture" are not sufficient and that a rigorous mathematical proof is required.

(b) For two given functions f and g, indicate whether $f \in O(g)$, whether $f \in \Omega(g)$, and whether $f \in \Theta(g)$.
Here *yes/no* answers are sufficient. Provide your solutions in a table as indicated right-below and where the entries are filled with either *yes* or *no*.

Nr	$f(n)$	$g(n)$
(1)	$n^3 + 5n + \cos(3n^4)$	$(\frac{2}{n^4} + n)^3$
(2)	$\frac{6n^5+n^2-8}{n^3+8}$	$5n^3 - 6\sqrt{n} - 1$
(3)	$n^{\frac{n}{3}}$	10^n
(4)	$n \log_2(n)$	$n \log_{10}(\frac{n}{2} + 1)$

Nr	$f \in O(g)$	$f \in \Omega(g)$	$f \in \Theta(g)$
(1)			
(2)			
(3)			
(4)			

Problem 4 (*Time and Space Complexity* (⋆⋆)) 1.5+1.5=3p

Provide the exact bounds for the time complexity $T(n)$ and space complexity $S(n)$ of the following program `NW(...)` in Θ-notation, assuming a unit-cost model.

Here is some more information about used commands.

- The input size is determined by the size n of the arrays a and b with $n =$`length(`a`)=length(`b`)`.
- *init empty array* $A[0..n-1]$ results in an empty array A of size n whose first entry is at position 0 and whose last entry is at position $n-1$.

Remark: You do **not** need to understand what the program computes in order to answer this question.

`NW`(array a, array b)

```
1  int n := length(a) //Comment: length(a) = length(b),
2  init empty array f[0..n − 1]
4  FOR (i := 0; i < n; i := i + 1) DO
3      init empty array g_i[0..n − 1]
3      f[i] = g_i
5  int d := 1
6  FOR (i := 0; i < n; i := i + 1) DO f[i][0] := d · i
7  FOR (j := 0; j < n; j := j + 1) DO f[0][j] := d · j
8  FOR (i := 1; i < n; i := i + 1) DO
9      FOR (j := 1; j < n; j := j + 1) DO
10         int match := 0
11         IF (a[i] = b[j]) THEN match := f[i − 1][j − 1]
12         ELSE match := f[i − 1][j − 1] + d
13         int del := f[i − 1][j] + d
14         int ins := f[i][j − 1] + d
15         f[i][j] := min(match, del, ins)
16 return f
```

Answers of the form $T(n) \in \Theta(..)$ and $S(n) \in \Theta(..)$ are sufficient and proofs are not needed. In other words, you only need to answer:

(a) Time complexity: Provide $f(n)$ such that $T(n) \in \Theta(f(n))$.

(b) Space complexity: Provide $g(n)$ such that $S(n) \in \Theta(g(n))$.

Problem 5 (*Master Theorem* ($\star\star\star$)) 2+1+5 = 8p

Recall, the Master Theorem specifies the runtime $T(n) \in \Theta(..)$ for certain algorithms and is based on the input size n and

a = number of subproblems in the recursion

n/b = size of a single subproblem

d for the overhead $g(n) \in \Theta(n^d)$

(a) Provide the values a, b and d for the algorithm `MergeSort` for sorting the elements in an array of length n as provided in the lecture.

What is the solution $T(n) \in \Theta(f(n))$ according to the Master theorem for `MergeSort`?

Simplify $f(n)$ as much as possible.

(b) What complexity $T(n) \in \Theta(f(n))$ would result from the Master theorem for the recurrence equation

$$T(n) = T(n/2) + \Theta(n^2)?$$

(c) Given is the following pseudocode of a divide-and-conquer approach to compute $\sum_{i=0}^{n-1} A[i]$ for a given array of size n.

`sumTernary`(array A, int left, int right)

```
1  IF (left = right) THEN return A[left]
2  int len := right − left + 1
3  int oneThird := ⌊len/3⌋
4  int m1 := left + oneThird − 1
5  int m2 := m1 + oneThird
6  s1 := sumTernary(A, left, m1)
7  print(s1)
8  s2 := sumTernary(A, m1 + 1, m2)
9  print(s2)
10 s3 := sumTernary(A, m2 + 1, right)
11 print(s3)
12 return s1 + s2 + s3
```

(i) Call `sumTernary`(A,0,4) for $A = [1, 5, 10, 100, 4]$ where the first entry of A is $A[0] = 1$ and provide the output of the `print` command in each of the single recursive calls in the order they appear.

(ii) Use the Master Theorem to determine the runtime $T(n)$ of `sumTernary` that takes as input an arbitrary array A of size n, assuming a unit-cost model. In particular, specify a, b, and d as well as the function $f(n)$ such that $T(n) \in \Theta(f(n))$.

Simplify $f(n)$ as much as possible.

Problem 6 (*Heap Sort* ($\star\star$)) 1.5+1.5= 3p

Given is the array $A = [4, 1, 3, 2, 16, 9, 10, 14, 8, 7]$

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

(b) Provide a drawing of A after transforming A into a max-heap using `Build-Max-Heap(`A`)` according to the lecture.

Problem 7 (*Counting Sort* ($\star$)) 3p

Given is the array $A = [2, 0]$. Consider the algorithm `COUNTING-SORT`$(A, 2, 2)$ as provided in the lecture. The array C is used to count certain elements in A and is initialized as an array whose entries are all 0.
Provide the array C after *all* steps in which the array C differs from the array C in the previous step.

Problem 8 (*Searching* ($\star\star$)) 2.5+2.5 = 5p

Given is the following sequence of numbers stored in an array L, sorted in ascending order:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$L[i]$	10	28	33	57	62	79	80	86	88	89	91	92	94	117	133	138

i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$L[i]$	157	167	179	212	223	233	236	239	258	265	271	283	286	287	295	300

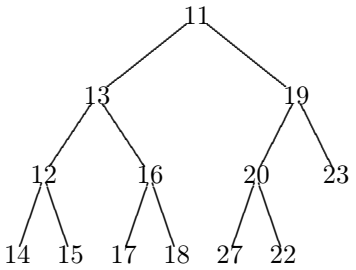
(a) Use `binary_search` as provided in the lecture and provide the search steps for searching for element 94. Specify for each step, in which part of the array you are searching and shortly state how this decision has been derived in each step.

(b) Use `Jump_Search` with optimal jump-width m as provided in the lecture for searching for element 94.

Provide the value of m. If you do not now how to compute m, proceed with $m = 4$. Specify for each step, in which part of the array you jump and shortly state how this decision has been derived in each step. In the final part, to locate 94 you can use linear search.

Problem 9 (*Properties of Trees* ($\star$)) 3+1 = 4p

(a) Given is the following rooted tree T.



Specify:

- the height of T
- the depth of vertex 13

Indicate (yes/no) whether T has the following properties:

- binary search tree
- fully-binary
- nearly-complete
- complete

(b) Recall, trees are graphs $G = (V, E)$ with the following two properties:

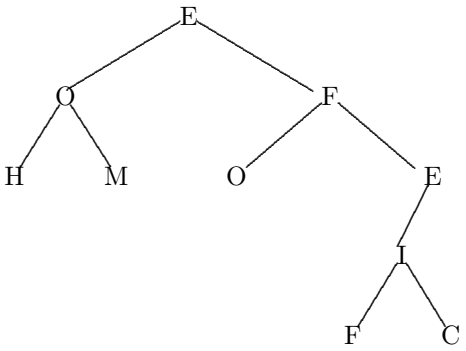
(i) G is connected and
(ii) G has $|E| = |V| - 1$ edges.

Prove or disprove the following statement:

All graphs $G = (V, E)$ with $|E| = |V| - 1$ edges are trees.

Problem 10 (*Traversing Trees* ($\star$)) 1+1=2p

Given is the following rooted tree T.



Write the keys in the order they are visited in T using

(a) post-order

(b) in-order

Problem 11 (*Binary Search Trees* ($\star\star$)) 2+2 = 4p

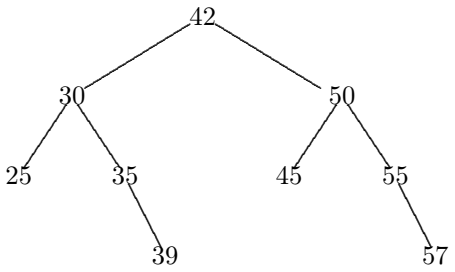
Given is the following sequences of keys `(1, 10, 8, 15, 16, 12, 11)` ordered from left to right.

(a) Build the binary search tree according to the lecture for this sequence. That is, insert the keys in the given order one after another into an initially empty binary tree. Provide a simple drawing of the tree after the last insertion.

(b) Now take your tree and delete the key `10` according to the lecture and draw the resulting tree.

Problem 12 (*AVL tree* ($\star\star$)) 1+2 =3p

Given is the following AVL tree T.



(a) Insert 37 into the tree T and draw the resulting tree "$T + 37$" including the balance factors next to each inner vertex.

(b) If "$T+37$" is not an AVL tree, rebalance it. If double rotations are needed, represent them as two single rotations. For required rotations, specify whether it is a left rotation (LR) or right rotation (RR), and specify the vertex where the rotation takes place.

Specify the rotation(s) for rebalancing and draw the resulting tree after each rotation.

Problem 13 (*Hashing* ($\star$)) 2p

Given is a hash table H of size $m = 7$. Access to H is done using Linear Probing.
The hash functions for keys k is here defined as follows:

$$h(k, i) = (k + 2 \cdot i) \bmod m \qquad (i = 0, 1, 2, \ldots) \; .$$

Insert the keys `11,14,7,9` in this order into the initially empty hash table H; indicate the table and the respective value of i after each single insertion of a key into the hash table H.

Problem 14 (*Bloom Filter* ($\star$)) 1p

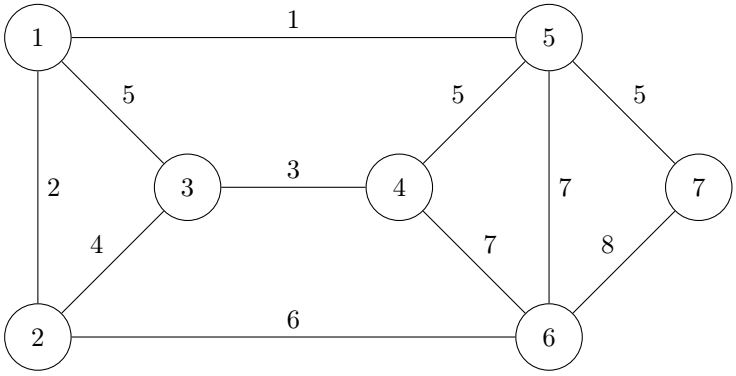
Given is the following bloom filter $(B, \mathcal{H})$ where the array $B = [0..8]$ consists of $m = 9$ bits, all initially set to 0 and $\mathcal{H} = \{h_1, h_2, h_3\}$ is the set of the hash functions

$$h_i(k) = (k + i) \bmod m; \text{ where } i \in \{1, 2, 3\}.$$

Based on this, provide the bloom filter that represents the set $S = \{5, 11, 19\}$.

Problem 15 (*Elementary Graph Algorithms* ($\star$)) 3+1 = 4p

(a) Given is the following undirected weighted graph G.



Find a minimum spanning tree of G using `Kruskal`'s algorithm. Provide the edges in the order in which they are inserted into the spanning tree. Edges that are **not** included in the spanning tree do not need to be specified.

(b) Provide a sketch of a graph $G = (V, E)$ with n vertices for which `DFS`(G, s) and `BFS`(G, s) would traverse the vertices of G in the same order for some (but not necessarily every) vertex $s \in V$.

Problem 16 (*A final puzzle* ($\star\star$)) 3p

The puzzle keeper stands as the guardian at the border, ready to challenge any who seek passage to the land of happiness and completed exams. With a knowing smile, the puzzle keeper speaks to you:

"I have a secret number taken from the set $\{0, 1, 2, 3, 4, 5, 6, 7\}$*. I will allow you to ask me two questions that can be answered with either 'yes' or 'no'. If you can tell me the exact number I have in mind after two such questions, you can pass. By the way, my number is even."*

Design your two questions and explain how, using the answers, you can uniquely determine the secret number.