MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Marc Hellmuth

Tentamensskrivning i
Datastrukturer och algoritmer
7.5 hp
2024-05-23

**General Information**

- Time 8:00 - 12:00
- No books, internet, smartphones, notebooks are any extra material are allowed to be used.
- In total, you can get .. points and you need to get at least ..(half-of).. points to pass the exam.
- There are statements "Prove *in detail* [..]".

  *In detail* means that arguments "as shown in the lecture" are not sufficient and that a rigorous proof is required.
- 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 problems.

**********

The following problems serve just as an idea for the potential exam structure.

Examples are in many case in extrem simplified form to give you an idea.

The exam problems can be expected to be more difficult than stated it here, but many of them follow to some extent the ideas as provided below.

Additional exam question not provided here might be asked.

**********

**Problem** (*Basic Questions*)

▶ A couple of basic questions will be asked.

**Example** (Examples might be provided in a very simplified form to give you an idea)

- What is an instance of a problem?
- What is a stack?
- What is a spanning tree of a graph?

**Problem** (*Algorithm Design*)

▶ Here pseudocode for a simple task should be provided and examined on a give example.

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Provide pseudocode for an recursive algorithm that takes as input an integer $n$ and that computes the sum $\sum_{i=1}^{n}$. Use only the basic arithmetic operation substraction and addition.

  Exemplify how your algorithm works step-by-step by using as input the integer $n = 7$ and provide the value of each of your used variables in each step of your executed algorithm.

**Problem** ($O, \Omega,$ *and* $\Theta$*-Notation*)

▶ Here you should be able to prove or indicate if for a given function $f$ it holds that $f \in O(g)$, $f \in \Omega(g)$ or $f \in \Theta(g)$ (it might be an extra task to determine $g$).

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Prove in detail that $T(n) = (3n + 2)^4 \in O(n^4)$
- Indicate with *yes* or *no* if $\frac{1}{3}2^n \in O(3n^2)$
- Determine an asymptotically tight bound $g(n)$ such that $f(n) = \frac{1}{3}2^n \in \Theta(g(n))$.

**Problem** (*Time and Space Complexity and Master Theorem*)

▶ Here you should be able to determine for a given algorithm its space and time complexity. This might also include application of the Master Theorem.

**Example:** (Examples might be provided in a very simplified form to give you an idea)

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

```
Sum(int n)
1 total_sum := 0
2 FOR (i = 1 to n) DO
3   total_sum := total_sum + i
4 PRINT total_sum
```

- Given is the following pseudo-code of a divide-and-conquer approach.

```
Some_Rec(n)
1 IF (n > 1) THEN
2     print("hello")
3     Some_Rec(n/2) + Some_Rec(n/2)
```

  Call `Some_Rec`(10) and provide the output of the `print` command in each of the single recursive calls in the order they appear.

  Use the Master Theorem to determine the runtime $T(n)$ of `Some_Rec`$(n)$, assuming a unit-cost model. In particular, specify $a$, $b$, and $d$ as well as the function $f$ such that $T(n) \in \Theta(f)$.

- What complexity $T(n) \in \Theta(..)$ would result from the Master theorem for the recurrence equation $T(n) = 2T(n/4) + \Theta(n^{10})$?

**Problem** (*Sorting*)

▶ Here you should be able to apply any of the sorting algorithms as provided in the lecture.

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Given is the array $A = [2, 3, 1, 5, 4]$. Apply `counting_sort` as provided in the lecture to sort $A$. For each of the steps in `counting_sort` provide the auxilary array $B$ used to sort $B$ as well as the array $C$ used to count the occurences of keys (here $k = 5$).

**Problem** (*Searching*)

▶ Here you should be able to apply any of the searching algorithms as provided in the lecture.

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Given is the array $A = [2, 3, 1, 5, 4]$. Use `binary_search` as provided in the lecture and provide the search steps for searching for element 5. 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.

**Problem** (*Graphs and Trees*)

▶ Here you should be able to apply any of the results we established for graphs and tree, determine structural properties provide proofs of certain statements

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Given is the following rooted tree $T$.


  Specify the height of $T$ and if $T$ is nearly complete.

  Write the keys in the order they are visited in $T$ using post-order traversal

- Prove in detail the handshake-lemma.

**Problem** (*Binary Search Trees / AVL trees / Red-Black Trees*)

▶ Here you should be able to apply any of the results we established for binary search trees (including their subclasses)

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- For the given sequence of integers $1, 3, 2$ build the binary search tree, that is, insert the integers one after another as keys into an initially empty binary tree and draw the tree after each insertion.

- Given is the AVL tree $T$


  Insert 5 into the tree $T$ and draw the resulting tree "$T + 5$" including the balance factors next to each vertex. Specify the rotation(s) for rebalancing and draw the resulting tree after rebalancing it.

**Problem** (*Hashing*)

▶ Here you should be able to apply any of the results we established in the section hashing (**e.g. chaining, probing, bloom filters**)

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Given is a hash table $H$ of size $m = 17$.

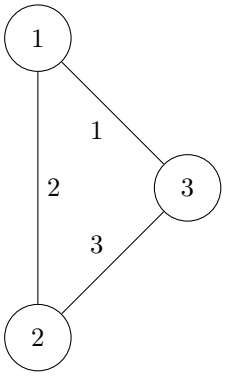
  Insert the keys `16,17` in this order into the initially empty hash table $H$ using the hash function $h(k) = k \bmod m$.

**Problem** (*Elementary Graph Algorithms*)

▶ Here you should be able to apply any of the results we established in the section Elementary Graph Algorithms (**e.g. BFS, DFS, Kruskal**)

**Example:** (Examples might be provided in a very simplified form to give you an idea)

- Given is the following undirected weighted graph $G$.




  Use `BFS`(1) and provide the order in which the vertices are visited (if several choices are possible prioritize the vertices from small to large). How many spanning tree does $G$ have? How many of them are minimum spanning trees?