Technical University of Munich

Esolution
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Fundamentals of Algorithms and Data Structures

Exam:	/ Bonus test 2	**Date:**	Friday 27^{th} June, 2025
Examiner:		**Time:**	10:00 – 11:00

Working instructions

- This exam consists of **10 pages** with a total of **10 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 100 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - None
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.
- *Mark correct answers with a cross* ☒
  To undo a cross, completely fill out the answer option ■
  To re-mark an option, use a human-readable marking ✕■

Left room from ________ to ________ / Early submission at ________

Problem 1 Complexity of Pseudocode (10 credits)

a) Problems that can be solved by “dynamic programming” exhibit two properties. What are the two properties?

- [x] Overlapping sub-problems
- [ ] Greedy-choice property
- [x] Optimal substructure
- [ ] Monotone substructure
- [ ] Independence of sub-problems
- [ ] Incremental sub-problems

b) The rod-cutting problem has which of the following properties?

- [ ] Greedy-choice property
- [x] Overlapping sub-problems
- [ ] Monotone substructure

c) The matrix-chain multiplication problem has which of the following properties?

- [x] Optimal substructure
- [ ] Monotone substructure
- [ ] Independence of sub-problems

Problem 2 LCS (10 credits)

Recall the dynamic programming algorithm for computing the Longest Common Subsequence (LCS) of two sequences X and Y.

0 1 2 3 4

a) Write down the recursive formula for $c[i,j]$, the length of the LCS of X_i and Y_j:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1 & \text{if } X_i = Y_j \\ \max(c[i-1,j], c[i,j-1]) & \text{if } X_i \neq Y_j \end{cases}$$

0 1 2 3

b) Consider the two sequences $X = TCGA$ and $Y = CGGTA$. Fill in the table $c[i,j]$ below with the appropriate values:

	C	*G*	*G*	*T*	*A*
T	0	0	0	1	1
C	1	1	1	1	1
G	1	2	2	2	2
A	1	2	2	2	3

0 1 2 3

c) Show how the longest common subsequence of X and Y can be extracted from the table. That is, add arrows in each cell and trace the path that gives us the LCS.

	C	*G*	*G*	*T*	*A*
T	↑	↑	↑	↖	←
C	↖	←	←	↑	↑
G	↑	↖	↖	←	←
A	↑	↑	↑	↑	↖

Tracing the path following the arrows starting in the bottom right corner gives the LCS "*CGA*".

Problem 3 Fibbish (10 credits)

Considder Fibbish, a language with a very peculiar frequency distribution of the letters in its alphabet. There are 8 letters in the alphabet and in a random sample of a Fibbish text with 54 characters “a” and “b” occur once, “c” occurs twice, “d” three times, “e” five times, “f” eight times, “g” thirteen times, and “h” twenty-one times.

a) What would be an optimal prefix encoding for the Fibbish alphabet?

0 1 2 3 4 5

b) In a set of parallel universes, Fibbish has $9, 10, \ldots, n$ characters. The same frequency pattern (the Fibonacci sequence) that holds for the 8-letter Fibbish, holds for these larger alphabets. What would be the optimal prefix encoding for the alphabet given some $n > 8$?

0 1 2 3 4 5

Problem 4 Rod Cutting (10 credits)

a) Recall the rod-cutting problem. Let $c(n)$ be the number of different ways we can cut a rod of length n into pieces of size $1, 2, \ldots, n$. What are the values of $c(1), c(2), c(3), c(4)$?

0 1 2 3 4 5

b) Write down the recursive formula for $c(n)$.

0 1 2 3 4 5

Problem 5 BST (10 credits)

Algorithm 1 The input is x, a node in a BST

```
if left[x] ≠ NIL then
    return BSTMax(left[x])
end if
y ← p[x]
while y ≠ NIL and x ≠ right[y] do
    x ← y
    y ← p[y]
end while
return y
```

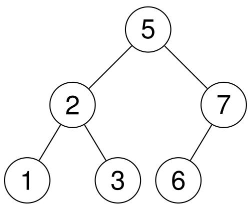
a) Describe in words what the pseudocode does.

b) What is its worst-case runtime?

- [ ] $O(\lg n)$
- [x] $O(n)$
- [ ] $O(n \lg n)$
- [ ] $O(n^2)$

c) Under what conditions does the `while` loop execute at least once in the given algorithm?

d) Given the following tree, what would be the output of this algorithm on the input $x = 6$?



- [ ] 2
- [ ] 3
- [x] 5
- [ ] 7

Problem 6 Open-Addressing Hash Table (9 credits)

Insert the keys $23, 12, 5, 71, 29, 44$ into a hash table of length $m = 9$ using open addressing and the hash function $h(k) = k \bmod m$. Illustrate the result of inserting the keys using linear probing, quadratic probing with $c_1 = 2$ and $c_2 = 5$, double hashing with $h_2(k) = 7 - (k \bmod 7)$

index	linear probing	quadratic probing	double hashing
0	44	29	-
1	-	-	-
2	29	5	29
3	12	12	12
4	-	-	44
5	23	23	23
6	5	44	-
7	-	-	5
8	71	71	71

Problem 7 Complexity (11 credits)

Match the problems/algorithms below with the worst-case upper bounds on the most efficient deterministic algorithms that we have studied for them, where n is the size of the input.

a) FIND in a BST

☐ $O(\lg n)$ ☒ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

b) FIND in a RB-Tree

☒ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

c) HUFFMAN'S Algorithm

☐ $O(\lg n)$ ☐ $O(n)$ ☒ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

d) Optimal Activity Selection, given that the input is already sorted by finish time

☐ $O(\lg n)$ ☒ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

e) Convex Hull

☐ $O(\lg n)$ ☐ $O(n)$ ☒ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

f) Optimal Rod Cutting

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☒ $O(n^2)$ ☐ $O(n^3)$ ☐ $O(2^n)$

g) Optimal Polygon Triangulation

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☒ $O(n^3)$ ☐ $O(2^n)$

h) Multiplying two $n \times n$ matrices

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☒ $O(n^3)$ ☐ $O(2^n)$

i) Optimal Parenthesization

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☒ $O(n^3)$ ☐ $O(2^n)$

j) Optimal Matrix Chain Multiplication

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☒ $O(n^3)$ ☐ $O(2^n)$

k) Enumerating all Rod Cutting solutions

☐ $O(\lg n)$ ☐ $O(n)$ ☐ $(n \lg n)$ ☐ $O(n^2)$ ☐ $O(n^3)$ ☒ $O(2^n)$

Problem 8 True or False (10 credits)

Check the correctness of the claims below. (If you need to make additional assumptions, write them down.)

a) The greedy algorithm for computing change in US currency (25,10,5,1) is optimal.

☒ True ☐ False

b) Any binary search tree storing n numbers can be transformed in any other BST with these n numbers using rotations.

☒ True ☐ False

c) It is possible that a leaf in a red-black tree is twice as deep as another leaf in the same tree.

☒ True ☐ False

d) It is possible that a leaf in a red-black tree is three times as deep as another leaf in the same tree.

☐ True ☒ False

e) The lower-bound for computing Convex Hull of a set of n points is $\Omega(n \lg n)$.

☒ True ☐ False

f) Huffman's algorithm creates an optimal prefix code.

☒ True ☐ False

g) The Shannon-Fano algorithm creates an optimal prefix code.

☐ True ☒ False

h) The longest path problem exhibits the optimal substructure property.

☐ True ☒ False

i) The longest common subsequence problem exhibits the greedy choice property.

☐ True ☒ False

j) Every optimization problem can be solved optimally with dynamic programming.

☐ True ☒ False

Problem 9 Hashing (8 credits)

Let $h(x)$ be a hash function mapping elements to integers $1, \ldots, n$. The hash function is uniform across its domain, that is, any given element is equally likely to be assigned any given integer in $[1, n]$.

a) What is the likelihood a new element x is assigned a particular index i?

☒ $\frac{1}{n}$ ☐ $\frac{1}{\binom{n}{2}}$ ☐ n ☐ $\frac{1}{i}$

b) How many elements must be added to the hash table to ensure a collision?

☐ $2n$ ☒ $n+1$ ☐ $2n+1$ ☐ n

c) If there are k unique elements already in the hash table with no collisions, what is the likelihood the next element inserted causes a collision?

☐ $\frac{1}{n}$ ☐ $\frac{n}{k}$ ☐ $\binom{k}{2}$ ☒ $\frac{k}{n}$

Problem 10 Knapsack Problem (12 credits)

A salesperson is packing their van for a special event. They have n items to choose from. Each item i has a size, s_i, and a value, v_i. The van has a size limit of S (all positive integers).

The salesperson must choose which items to take in order to maximize the total value without exceeding the van's capacity. Let $a_i \in \{0, 1\}$ indicate whether item i is selected. Then the goal is to:

$$\max \sum_{i=1}^{n} a_i v_i \quad \text{subject to} \quad \sum_{i=1}^{n} a_i s_i \leq S$$

Design and analyze a dynamic program for this problem in the following steps.

a) Write the recurrence relation that characterizes the optimal substructure of the problem. Express it using a dynamic programming table. Clearly define what each table entry represents (the meaning of the indices), specify any base cases, and state where the final answer can be found. A description of each case is required for full credit.

b) Describe a polynomial-time algorithm that computes an optimal solution, using the recurrence relation above. Your algorithm should find both the value of an optimal solution, as well as the set of items that achieve it. Pseudocode is not required, but may help illustrate your solution.

c) Give a correctness argument and complexity analysis of your above solution.

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.