

School of Information Technology

## WRITTEN EXAMINATION

Course Algorithms and Data Structures G1F

Course code IT405G

Credits for written examination 6

Date 2025-02-28

Examination time 14:15-19:30

Examination responsible Yacine Atif

Aid at the exam/appendices Swedish-English / English-Swedish dictionary

Other

- Instructions
- ☐ Take a new sheet of paper for each teacher.
  - ☒ Take a new sheet of paper when starting a new question.
  - ☒ Write only on one side of the paper.
  - ☒ Write your name and personal ID No. on all pages you hand in.
  - ☒ Use page numbering.
  - ☒ Don't use a red pen.
  - ☒ Mark answered questions with a cross on the cover sheet.

**Examination results should be made public within 18 working days**

*Good luck!*

Total number of pages: 11

**QUESTION 1 [Course Goal 1: Complexity Analysis]**

- a) Consider the following pseudo-codes and describe the running time in Big-O notation in terms of the problem size  $n$ . Formulate your solution in the form of explanatory steps.

```
i. for (i = 0; i < n; i++){
    i = 2 * i;
    print(i);
}
for (i = 0; i < n; i++){
    for (j = 0, j < n; j++){
        print(i*j);
    }
}
```

```
ii. for (i = 0; i < m; i++)
    for (j = 0; j < n ; j++)
        for (k = 0; k < 50; k++){
            print(i*j*k)
        }
```

```
iii. for (int i = 0; i < n + 100; ++i) {

    for (int j = 0; j < i * n ; ++j){
        sum = sum + j;
    }

    for (int k = 0; k < n + n + n; ++k){
        c[k] = c[k] + sum;
    }

}
```

b) Consider the following functions:

```
i. int foo(n, m, k) {  
    U = 0;  
    if k == 0 {  
        return n+m;  
    } else {  
        U = foo(n-1, m, k-1) + foo(n, m-1, k-1)  
        return U;  
    }  
}
```

- What is the returned value from `foo(2, 2, 2)` and `foo(3, 2, 3)`. Show your steps.
- Analyze with explanation the time complexity of this function.

```
ii. void foo(int n, int m) {  
    if (m > n) return;  
    print(m);  
    foo(n, m+2);  
}
```

- What is the output of `foo(2, 0)` and `foo(4, 0)`. Show your steps.
- Analyze with explanation the time complexity of this function.

```
iii. void foo(int n, int x, int y) {  
    for (int i = 0; i < n; ++i) {  
        if (x < y) {  
            for (int j = 0; j < n * n; ++j)  
                print(j);  
        } else  
            print(i);  
    }  
}
```

- Analyze with explanation the time complexity of this function.

EXAMINATION OF ALGORITHMS AND DATA STRUCTURES, G1F  
2025-02-28

- c) A function `findChars(chars)`, takes a list of characters (eg. `chars = ['m', 'i', 'z', 't']`) and returns a list of 26 elements representing the 26 English alphabet, where the position of the character is one (`alphabet_position[x] = 1`) if the  $x^{\text{th}}$  element exists in the `chars[]` and zero otherwise. For example, 'm', which exists in `chars[]` is the 13<sup>th</sup> element, therefore `alphabet_position[13] = 1`, while the character 'n' which is in position 14 doesn't exist in `chars[]`, therefore `alphabet_position[14] = 0`.
- i. Write a pseudocode for the function `findChars(chars)` that takes a list of characters as an input and returns a list of `alphabet_position`. You can assume that a list of alphabet characters `char alphabet_list[26]` is already initialized with the alphabet characters. `alphabet_position[26]` is initialized with zeros. All characters are lowercase, and no other characters are expected.
  - ii. Assume the `char` value of a character returns the ASCII code for that character. For example `int a = int('a')`, assigns the value **97** to the variable `a`, which is the ASCII code for the character 'a'. The lowercase alphabet in the ASCII table goes from 97 for 'a' to 122 for 'z' following the alphabet order. Write a pseudocode for the function `findChars(chars)` that uses the above ASCII code conversion to solve the same problem as in i.
  - iii. Analyze and compare the time complexity of the above two algorithms (in i and ii).

**QUESTION 2 [Course Goal 2: Data Structures]**

- a) Consider a **Stack** `s`, **Queue** `q` and an **array** `a` with a capacity of 5 elements, and the following methods:

```
void insert (int x){
    s.push( x );
    q.enqueue( x );
    a.insert( x , a.last() )
}
```

```
void remove(){
    s.pop(); // removes the element from the stack
    q.dequeue(); // removes the element from the queue
    a.delete(a.first());
}
```

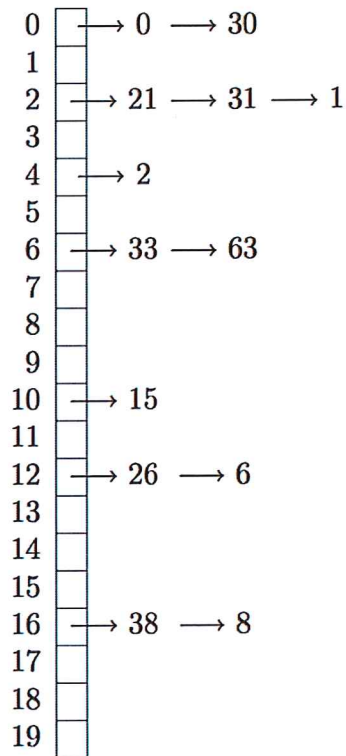
EXAMINATION OF ALGORITHMS AND DATA STRUCTURES, G1F  
2025-02-28

Show the status of each container after running the indicated methods in sequence (**use your answer sheets only**):

Method	Stack s	Queue q	Array a
insert(2)			
insert(4)			
insert(6)			
remove()			
remove()			
insert(1)			
insert(3)			
remove()			
insert(5)			

b) Given the hash table that shows:

- Table size: 20 slots (0-19)
- Current entries and their chaining sequences:



- Why it appears to be inefficient?
- What could be the cause the above inefficiency?
- How do you suggest to improve it?

c) Binary Tree

- Define the AVL balance property and explain how it is calculated for any node in the tree?
- Starting with an empty AVL tree, insert the following keys in the given order: 4, 10, 3, 8, 5, 6, 25. Circle the final tree, after drawing intermediate trees.
- Provide **with explanation** the *worst-case* running-time of an insert operation in an *AVL tree* of size  $n$

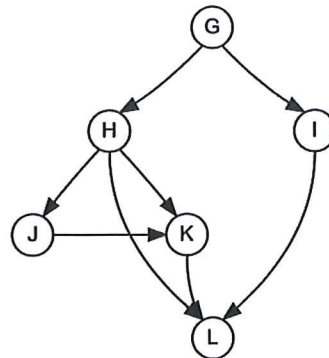
**QUESTION 3 [Course Goal 3: Sorting]**

- a) There are several algorithms that sort a given list.
- Describe one sorting algorithm that has worst-case complexity of  $O(n^2)$ .
  - Describe one sorting algorithm that has worst-case complexity of  $O(n \log n)$
  - Besides worst-case time complexity, explain an additional factor that influences your choice of sorting algorithms.
- b) Sorting with a pivot.
- Which of the following sorting algorithms uses a pivot for sorting:
    - Insertion sort
    - Bubble sort
    - Quick sort
    - Merge sort
    - Selection sort
  - This algorithm makes comparisons between the pivot and each element, which are then pushed either to the left or right side of the pivot. This process is then repeated recursively. Describe why it is important to select a good pivot element.
  - Describe how the worst and best case look like for the above algorithm (in ii).
- c) Heap Sort
- Indicate with explanation the properties of Heap Sort
  - Provide Heap Sort worst case complexity with an explanatory justification
  - Determine the speed ratio (in terms of execution time) of heap sort over insertion sort, when sorting 1 million items? Note:  $\log_2(1,000,000) = 20$ .



**QUESTION 4 [Course Goal 4: Graphs]**

- a) Consider the following graph. If there is ever a decision between multiple neighbor nodes, assume we **always choose the smallest letter in alphabet order**.



- i. The graph is (select TRUE or FALSE):

	TRUE	FALSE
A. Complete		
B. Directed		
C. Weighted		
D. Connected		

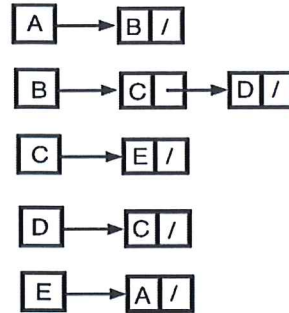
- ii. What is the order of visited nodes using BFS algorithm (starting from Node G)?

- iii. What is the order of visited nodes using DFS algorithm (starting from Node G)?

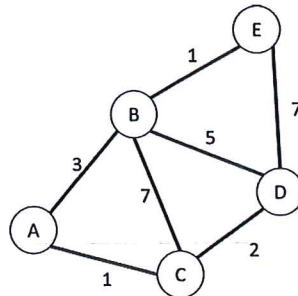


b) Graph Representation

The adjacency list representation of a graph with five vertices A, B, C, D, E is given below.



- Provide the adjacency matrix.
  - Draw the graph.
  - Discuss the adjacency list and matrix representation of the graphs in terms of complexity analysis.
- c) Consider the following graph with source Node A and destination Node E:



- Create Dijkstra table showing the step-by-step execution of Dijkstra's algorithm. Your table should include all iterations until the shortest path to Node E is found.
- What is the maximum number of items in the priority queue used to implement Dijkstra algorithm?
- Explain why Dijkstra's algorithm does not work if some of the link costs are negative.

**QUESTION 5 [Course Goal 5: Algorithm Design]**

a) Algorithm Design

Consider a function  $f$  where  $n$  is an integer ( $n \geq 1$ ):

$$f(n) = f(n-1) + 2 \cdot f(n-3) \quad \text{for } n > 3$$

$$f(1) = f(2) = f(3) = 1$$

- i. Write a recursive pseudo-code to compute the function  $f$  without using dynamic programming.
- ii. Using the recursive method, compute  $f(5)$  and  $f(6)$ .
- iii. Describe any relevant observations regarding the **time complexity** of the recursive computation and explain why it may be inefficient.

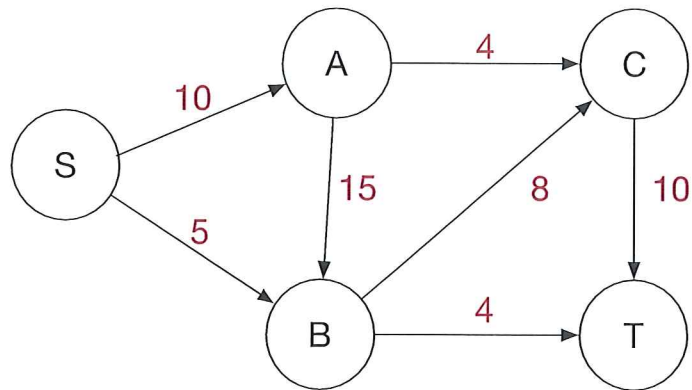
b) Dynamic programming

- i. Using the "**Bottom-Up**" method, resolve the above problem (a) and compute  $f(5)$  and  $f(6)$ . Explain your solution approach.
- ii. Using the "**Top-Down**" method with memoization, resolve the above problem (a) and compute  $f(5)$  and  $f(6)$ . Describe your approach..
- iii. Compare the "Bottom-Up" and "Top-Down" methods in terms of complexity and ease of implementation.

c) MaxFlow

In the context of network flow problems, consider a **directed graph** representing a traffic network where each edge has a capacity indicating the maximum number of vehicles that can pass through per unit time.

- i. Given the following directed graph with capacities on the edges, use the Ford-Fulkerson algorithm to find the maximum flow from  $S$  to  $T$ . Show the flow in each step and the residual capacities.



- ii. Calculate the total flow from S to T after applying the Ford-Fulkerson algorithm.
- iii. Explain the concept of an augmenting path in the context of the Ford-Fulkerson algorithm and its significance in determining the maximum flow.