UNIVERSITY
OF SKÖVDE

School of Information Technology

WRITTEN EXAMINATION

Course Algorithms and Data Structures G1F

Course code IT405G Credits for written examination 6

Date 2026-03-06 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: **10**

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.

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'th** element exists in the **chars[]** and zero otherwise. For example, 'm', which exists in **chars[]** is the 13th 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) You are given a collection of elements that must support the following operations:

- Insert an element
- Remove an element
- Check whether an element is present

i. Name **two** different data structures that can be used to store the elements. Briefly describe how each data structure supports the three operations above.

ii. For each data structure you named in (i), state the worst-case time complexity of the following operations (state any implementation-specific assumptions):

- `insertion`,
- `removal`,
- `membership_check`.

iii. Briefly explain one advantage and one disadvantage of each of the data structures you named in (i), in practice.

b) Queue, Stack, Hash and Heap:

i. Stack and Queue

Stacks and queues are container ADTs that obey different ordering principles. Show the output produced by the following program, where a **stack** and a **queue** store integers.

```
void In(int item, queue<int>& q, stack<int>& s)
{
    s.push(item);
    q.push(item);
}

void Out(queue<int>& q, stack<int>& s)
{
    s.pop();
    q.pop();
}
```

// The following utility functions display the contents of the queue and the stack:

```
void showQueue(queue<int> mq)
{
    while (!mq.empty())
    {
        cout << "\t" << mq.front();
        mq.pop();
    }
    cout << endl;
}

void showStack(stack<int> ms)
{
    while (!ms.empty())
    {
        cout << "\t" << ms.top();
        ms.pop();
    }
    cout << endl;
}
```

// Consider the following `main()` function:

```
int main()
{
    queue<int> q;
    stack<int> s;

    In(1, q, s);
    In(2, q, s);
    In(3, q, s);
```

```
    In(4, q, s);

    Out(q, s);
    Out(q, s);

    In(5, q, s);
    Out(q, s);
    In(6, q, s);

    cout << "The queue contains:";
    showQueue(q);

    cout << "The stack contains:";
    showStack(s);

    return 0;
}
```

ii. Consider a hash table of size **7** that uses the hash function:

$$h(k) = k \bmod 7$$

Show the contents of the hash table after inserting the following keys (State your collision resolution approach using a method **shown in class**):

$$\{10, 24, 31, 18, 7\}$$

iii. Construct the resulting heap after inserting all the following sequence of integers in the given order, into a priority queue that is implemented as a min-heap:

$$\{18, 4, 12, 25, 7, 20\}$$

Show all steps including the final heap tree.

c) Binary Tree

i) Define the AVL balance property and explain how it is calculated for any node in the tree?

ii) 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.

iii) 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.

i. Describe one sorting algorithm that has worst-case complexity of $O(n^2)$.
ii. Describe one sorting algorithm that has worst-case complexity of O(n log n)
iii. Besides worst-case time complexity, explain an additional factor that influences your choice of sorting algorithms.

b) Sorting with a pivot.

i. Which of the following sorting algorithms uses a pivot for sorting:

- Insertion sort
- Bubble sort
- Quick sort
- Merge sort
- Selection sort

ii. 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.

iii. Describe how the worst and best case look like for the above algorithm (in ii).

c) Heap Sort

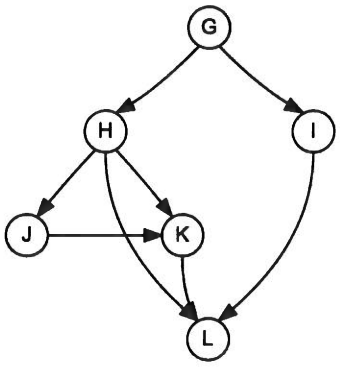
i. Indicate with explanation the properties of Heap Sort

ii. Provide Heap Sort worst case complexity with an explanatory justification

iii. 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):

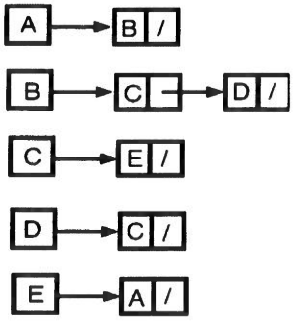
	TRUE	FALSE
A. Complete		
B. Directed		
C. Weighted		
D. Connected		

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.

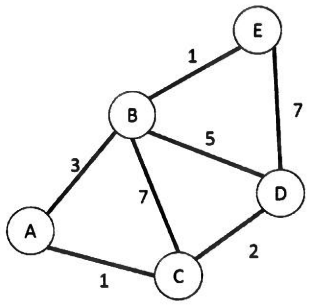


i. Provide the adjacency matrix.

ii. Draw the graph.

iii. 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:



i. Create Dijkstra table 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.

ii. What is the maximum number of items in the priority queue used to implement Dijkstra algorithm?

iii. 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 ff 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.

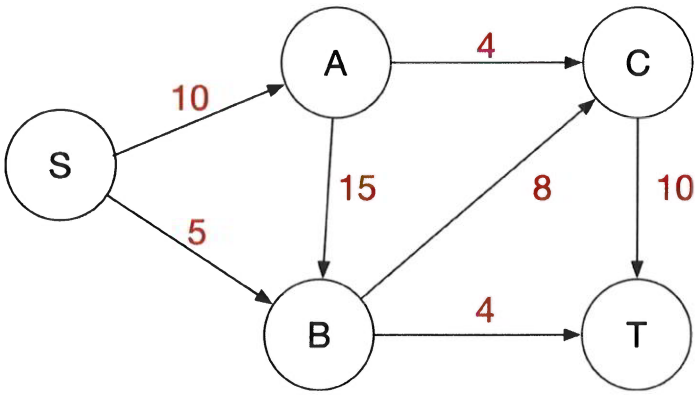
ii. Using the **"Top-Down"** method with memoization, resolve the above problem (a) and compute f(5) and f(6). Describe your approach..

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.