Algorithms and Data Structures (L.EIC011) 2024/2025

Exam Sample Questions

Student ID: | | | | | | | | | |

Full name: _____

Group 1 - Correctness Analysis and Asymptotic Complexity

1.1. For the following function pairs, write **Θ**, $\mathcal{O}$, or **Ω** in the blank box to make the statement true. Note that you must **write the symbol that best characterizes the relationship**.

$n^4 = \square\ (4^n)$ $\quad 2^n = \square\ (2^{n+2})$ $\quad 42 = \square\ 24$ $\quad \log(\log n^2) = \square\ (\log n^2)$ $\quad \sqrt{n} = \square\ (\log_4 n)$

1.2. Describe the worst-case running time for the following code snippets, using $\mathcal{O}$ notation, as a function of n, which is an integer (`int`). Use the tightest possible limit (e.g. indicating all as being $\mathcal{O}(n!)$ will result in 0 points).

a) _____

```
for (int i=0; i<n; i++)
   for (int j=0; j<i; j++)
         count++;
```

b) _____

```
int f1(int n) {
   if (n <= 0) return 0;
   return f1(n/2) + 1;
}
```

c) _____

```
int f2(std::vector<int> & v, int k) {
   int n = v.size();
   int m = 0, q = k, j = 0, s;
   while(q <= n) { // assume 1 <= k < n
       s = 0;
       while(j < q) {
          s += v[j];
          j++;
       }
       if (s > m) m = s;
       q += k;
   }
   return m;
}
```

1.3. Justify your answer to question **1.2.b)**. Use the recursion tree and you can assume that n **is a power of** 2.

1.4. Consider function `f2` defined in question **1.2.c)**, with $1 \leq k < n = v.size()$, and `v[j] > 0`, for all `j`.

a) Justify your answer to question **1.2.c)**.

b) We would like to prove that `f2` returns the maximum of $\{ \sum_{t=k(p-1)}^{kp-1} x[t] \mid 1 \leq p \leq \lfloor n/k \rfloor \}$. State the **invariant** of the `while(q<n)` loop that implies that. Give a brief **explanation** why that invariant is true (by checking the properties we refer to as Initialization, Maintenance, Progress, Termination) and how that conclusion follows from it.

1.5. Predict Running Time.
Complete the table using the information already given in each line. For times, use a prediction based on the time already filled in, taking into account the ratio between n_1 and n_2 (ms = milliseconds).

Program	Complexity	Usual name	Time for $n_1 = 10$	Time for $n_2 = 20$
A		constant	10ms	
B		linear		50ms
C	$\Theta(n^2)$		20ms	
D		cubic		40ms
E	$\Theta(2^n)$		10ms	

1.6. Discuss the truthfulness or falsity of the sentence: "If `bool func(std::vector<int> & v, int k)` is any function whose asymptotic time complexity $O(1)$ then, **for the same cost model**, the execution times of `func(x,p)` and of `func(x,q)` are equal, for all values `p` and `q`, and vectors `x`, such that `x.size() >= 1`.

Student ID | | | | | | | | | | Name:

Group 2 - Searching and Sorting

2.1. Suppose you have an implementation (in C++) for sets of words that uses as container an (unordered) array of type `std::vector<string>`, where the words of the set are stored (one for each position). Assume that all **words have less than 100 characters**.

a) Using notation $\mathcal{O}$ or Θ, characterize the **asymptotic time complexity**, in the worst case, of optimal algorithms for the following methods, when the set has n words. Justify your answers briefly.

Obtain the number of words in the set:

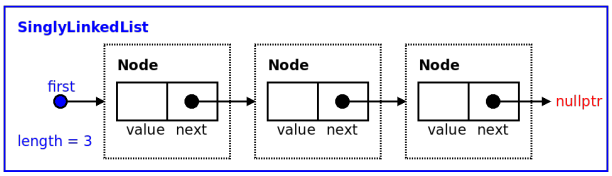
Check if some word is in the set:

b) If the vector was sorted in lexicographic increasing order, could the worst case time complexity of checking if a given word w is in the set be improved? How?

c) We need to check whether K words belong to a given set of n words. Explain briefly why if $K \in \Omega(\log n)$ it can be worthwhile start sorting the vector by a comparative sorting algorithm. How large should K be if we use **SelectionSort** for sorting in that preprocessing phase?

Group 3 - Lists, Stacks and Queues

3.1. Consider a template class `SinglyLinkedList<T>` representing a generic simply linked list similar to what was done on classes.



Recall that the methods `getNext`, `setNext` and `getValue()` of the class `Node<T>` were defined by

```
Node<T> *getNext() { return next; }
void setNext(Node<T> *n) { next = n; }
T & getValue { return value;  }
```

a) Explain why the following method removes the element that is in position `k`, if `k` is a valid position, assuming zero-based numbering, that is, the first element of the list is in position 0. Which is the loop invariant that allows us to conclude that the function is correct?

```
void remove(int k) {
    if (k >= length || k < 0) return;
    Node<T> *prev = nullptr;
    Node<T> *victim = first;
    while (k > 0) {
       prev = victim;
       victim = victim -> getNext();
       k--;
    }
    if (prev == nullptr)
      first = victim -> getNext();
    else prev -> setNext(victim->getNext());
    delete victim;
}
```

b) Using either Θ, $\mathcal{O}$, or Ω, characterize the time complexity of `remove(k)` as a function of `k`, in the worst case [ ] and best case [ ].

c) Write in C++ a method `int count(const T & x, int a, int b)` that counts the number of occurrences of `x` between positions `a` and `b`, inclusive, if `a` and `b` are valid (otherwise, returns 0). Present its time complexity in the worst case [ ] and in the best case [ ], using `a` and `b` and either Θ, $\mathcal{O}$, or Ω.

3.2. State a situation where it makes more sense to use a linked list instead of a simple array. Explain briefly.

3.3. Explain the difference between a stack, a queue and a priority queue.

3.4. Recall the algorithm Graham scan (based on rotational sweep) and the incremental algorithm (by Edelsbrunner) described in class for computing the convex hull of a set of n points in the plane.

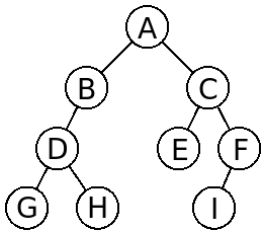
a) Illustrate how the algorithm Graham Scan works.

b) Which step dominates the time complexity of Graham Scan? Explain why.

c) In the incremental algorithm we used a doubly linked circular list to represent the convex hull whereas in the algoritm by Graham, we had a stack. Explain why.

Group 4 - Binary Trees

4.1. Considering the **tree in the following figure**, answer to the following questions:

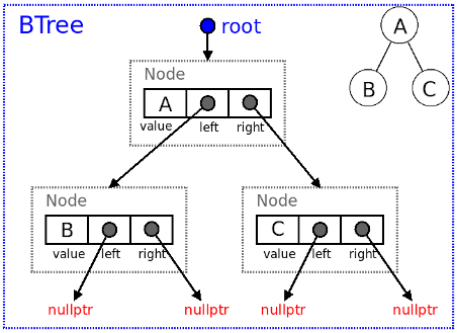


a) The nodes of the tree in **preorder:**

b) The nodes of the tree in **inorder:**

c) The nodes in a **breadth-first order:**

4.2. Consider a template class `BTree<T>` representing a generic binary tree similar to what was done on classes.



Recall that the nodes are structs:

```
struct Node {
  T value;              // The value stored on the node
  Node *left, *right; // Pointers to left and right child
};
```

And that the root is a pointer to a node:

```
Node *root;             // Pointer to the root node
```

You can assume that the tree is already created.

a) Implement a **method** `height()` for the class `BTree<T>` that returns the height of the tree. You can access the root and the attributes of a node, but you cannot use any other class method. Please indicate the **time complexity** of the method you wrote.

b) Implement a **method** `level(k)` for the class `BTree<T>` that returns the number of nodes at depth k. You can access the root and the attributes of a node, but you cannot use any other class method. Please indicate the **time complexity** of the method you wrote

Group 5 - Binary Search Trees

5.1. Suppose you insert the following numbers on a binary search search tree: 42, 20, 24, 50, 53, 11.
For the following 3 questions **draw the resulting tree** (after the corresponding operation is complete):

a) What is the **original tree** after the 6 numbers are inserted?

b) How does the (original) tree look like after we **add 51**?

c) How does the (original) tree look like after we **remove 42**?

5.2. In what node is **smallest element** on a binary search tree? Give a brief justification.

5.3. Suppose you want to insert the 15 integers from 1 to 15 on a binary search tree. What is the **minimum height possible** of the resulting tree? In **what order should you insert the elements** to guarantee that minimal height?

Minimal Height: | | Insertion Order: | | | | | | | | | | | | | | | |

5.4. Explain (in words) an algorithm for **searching for an element on a binary search tree**. What is the **time complexity** of the method you described assuming the tree is balanced? Briefly justify.

5.5. Explain the **key ideas of an AVL tree**, indicating what restrictions it imposes, how it maintains these properties and what its temporal complexities are for an insertion and removal.

5.6. Name **one advantage and one disadvantage** of AVL trees compared to Red-Black trees.

Group 6 - Graphs

6.1. Explain what the **diameter** of a graph is and draw a graph with diameter 3 where all the nodes have degree > 1.

6.2. Graph Representation

Complete each of the blank boxes with the words **LIST** or **MATRIX** to indicate whether the answer should be an adjacency list or an adjacency matrix. Assume we are dealing with **sparse graphs**.

takes up less memory

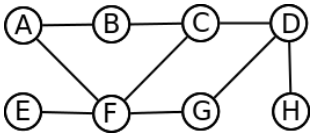
is better for checking if there is a link between a pair of nodes

is better to remove a single edge

is better for deleting all edges connected to a given node

6.3. Graph Traversal Order

a) Consider the graph in the following figure. Imagine that you **start a search at vertex** (G) and that the neighbor nodes are always **traversed in alphabetical order**. Indicate the **order** in which the nodes would be visited in a:



Depth-First Search (DFS):

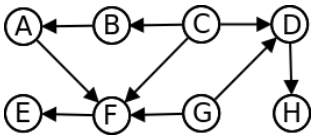
Breadth-First Search (BFS):

b) Fill in the blank spaces assuming $G = (V, E)$ and using V and E as the variables and using $\mathcal{O}$ notation.

A DFS or a BFS using an **adjacency matrix** has time complexity

A DFS or a BFS using an **adjacency list** has time complexity

6.4. Topological Sorting. Indicate two possible topological sorting node orders for the following graph:

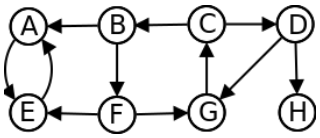


Two different topological sortings:

1)

2)

6.5. Strongly Connected Components. Indicate the strongly connected components (SCCs) of the following graph:



Nr SCCs:

Nodes in each SCC:

6.6. Code for DFS. Write (in code or pseudo-code) a function `dfs(G,v)` to make a depth-first search starting from node v in graph G. **The function should return the number nodes that were visited**.

6.7. BFS Application.

Explain how BFS can be used for finding the **shortest distance** from a node u to a node v in **unweighted** graphs and indicate the time complexity of finding that shortest distance. Briefly explain why BFS would not produce the correct answer for a **weighted** graph.

Group 7 - Hash Tables

7.1. Give a brief explanation of what a **hash function** is.

7.2. Explain why and in what types of positive integers would the following hash function be a **bad choice**.

```
int hash(int n) {
  int h = 0;
  while (n>0) {
    h += n % 10;
    n /= 10;
  }
  return h;
}
```

7.3. Imagine you have an initially empty hash table with an **open addressing** strategy, **linear probing** and capacity for 5 elements. Suppose you use the **hash function from 7.2** (with **modular hashing** for fitting in the table).

a) Draw the state of the hash table after the following consecutive insertions (assume a 0-based index):

insert(4) | | | | | |

insert(9) | | | | | |

insert(1231) | | | | | |

insert(1001) | | | | | |

b) What is the **load factor** after the insertions?

c) Explain of advantage and one disadvantage of **open addressing** when compared with **separate chaining**.

Group 8 - Priority Queues and Heaps

8.1. Imagine a **maxHeap** described by the following array:

9	8	7	4	5	6

For the following 3 questions **draw the tree** (the heap) always with the invariant restored.

a) What is the heap represented by the **original array**?

b) What does (original) heap look like if we **add a 10**?

c) What does (original) heap look like if we **remove the max**?

8.2. Suppose you read a set of n numbers, inserting them one by one into a priority queue implemented with a *maxHeap*. Then you removes them one by one from the heap and print the values as they are removed.

a) In what **order** do the numbers appear in the output?

b) Indicate, justifying, the **temporal and spatial complexity** of the whole process.

(this is just a selection of sample exam questions aimed towards showing the type of questions you might encounter)

(the number and difficulty of questions of the real exam will be calibrated for its duration)

(you will be able to (pre)choose a portuguese or an english version of the exam)