DATA STRUCTURES AND ALGORITHMS

Extra reading 13

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș - Bolyai University
Computer Science and Mathematics Faculty

2023 - 2024

Content

- This document contains a few sample exam questions.
- It is not a sample exam subject, it just contains several problems to show you possible problem types.
- I encourage you to try to solve the problems (alone or in teams). The solutions (and some more details about the exam) will be discussed in Lecture 14.

Quiz questions I

- Consider a hash table with m positions, collision resolution with open addressing, which currently contains n elements. Can the value of n be greater than the value of m?

  a) No, it is not possible, no matter how the hash function is defined.
  b) Yes, it is possible, no matter how the hash function is defined.
  c) If we use double hashing it is possible.
  d) If m is a power of two and we use quadratic probing, it is possible.

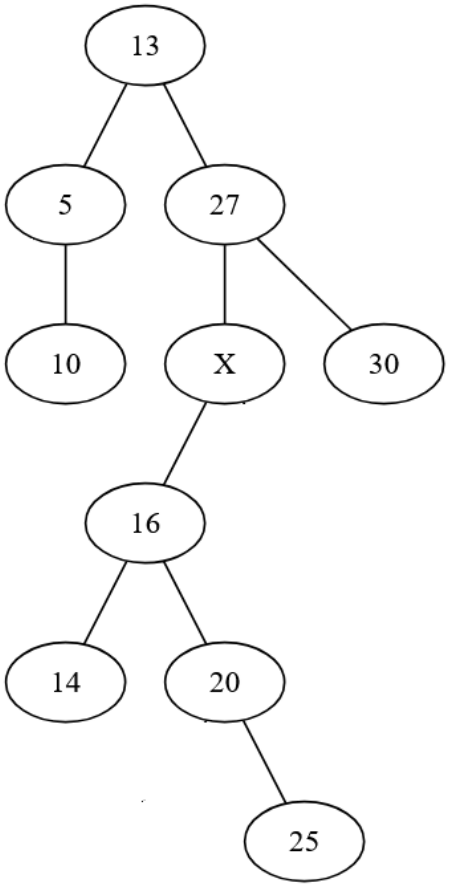
Quiz questions II

- Consider a hash table with m positions, collision resolution with open addressing, which currently contains n elements. If we do a search for a given element e, is it possible that during the search more than n positions are checked?

  a) No, it is not possible.
  b) It is possible only if e is not in the hash table.
  c) It is possible, no matter whether e is in the hash table or not.
  d) For a search is not possible, but it we had to count how many times e occurs in the hash table, then we could check more than n positions.

Quiz questions III

- Consider the BST below. What value could be in the node marked with X?



a) 21
b) 24
c) 22
d) 26
e) 28

Quiz questions IV

- Starting from an initially empty stack, we push into it, in the given order, the following elements: 1, 2, 3, 4, 5, 6. Then we pop an element and push it into another, initially empty stack. We do this three more times (so we pop a total of 4 times). After this, we pop an element from the second stack. What value is now on the top of the second stack?

a) 1
b) 2
c) 3
d) 4
e) 5
f) 6

Quiz questions V

- What is the result of evaluating the following expression made of single digit numbers and the regular operators?
  4 6 7 + 1 - 2 5 * + +

a) A value below -15
b) A value between -15 and -5
c) A value between -5 and 5
d) A value between 5 and 15
e) A value above 15.

Quiz questions VI

- What is the difference between the minimum and maximum height that we can have for a binary tree with 20 nodes?

a) 19
b) 15
c) 11
d) 8
e) 5

Quiz questions VII

- What data structure is the most suitable representation for ADT List, if we will frequently want to get an element from a position p (*getElement* operation will be called frequently)

a) dynamic array
b) singly linked list
c) doubly linked list
d) skip list
e) binary heap
f) hash table

Quiz questions VIII

- What complexity class does the following expression belong to: $12 * n^3 + n * log_2 n + 52$?

a) $O(n^4)$
b) $O(n^2)$
c) $\Theta(n * log_2 n)$
d) $\Omega(n^3)$
e) $\Omega(1)$
f) $\Theta(n^4)$

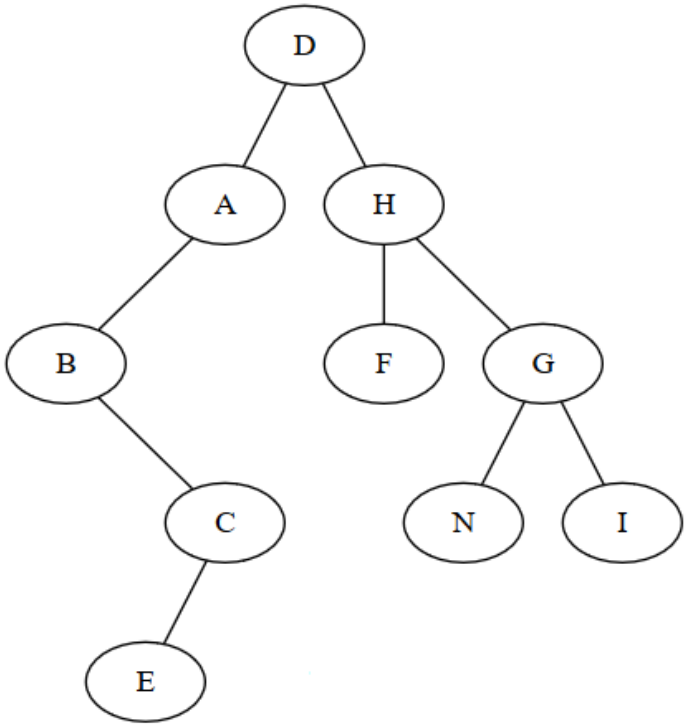
Quiz questions IX

- How many binomial trees are in a binomial heap with 27 elements?

a) 1
b) 2
c) 3
d) 4
e) 5
f) 6
g) 27

Drawing problems I

- For the Binary tree below, specify the preorder, postorder, inorder and level order traversal.



Drawing problems II

- Consider a hash table with $m = 8$ positions in which the division method is used as hash function and separate chaining is used as collision resolution method, but every position of the table contains the root of an AVL tree (instead of containing a linked list).
- Show how the following elements can be added into this hash table (which is initially empty): 8, 99, 40, 19, 56, 16, 17, 28, 62.
- It is enough to draw the final hash table, but show how you computed the position for every element.
- Specify the load factor of the table.

Drawing problems III

- Starting from an initially empty binary search tree (built with the regular $\leq$ relation), instert into it, in the given order, the following elements: 41, 54, 60, 23, 73, 68, 98, 16, 13, 19, 36, 100, 76.
- It is enough to draw the final tree.
- Show the two possible trees after the removal of 41.

Drawing problems IV

- Is the following array a binary heap? If not, transform it into a binary heap by swapping two elements:

$$41, 54, 13, 73, 68, 98, 63, 100, 76$$

.

- In the (possibly modified) heap add the following elements (in this order): 19, 18, 59. After adding these 3 elements, remove an element from the heap. Draw the heap after every operation (4 drawings in total).

Complexity computation

- Compute the complexity of subalgorithm *main*:

```
subalgorithm operation(n, m) is:
//n, m are integer numbers
        if n > 1 then
                k ← [n/2]
                i ← 3
                while n > 0 execute
                        i ← i + n
                        n ← n - 1
                end-while
                operation(k, m)
        else
                for i ←1, m * m execute
                        print i
                end-for
        end-if
end-subalgorithm

subalgorithm main(n) is:
        operation(n, n)
end-subalgorithm
```

Implementation problems I

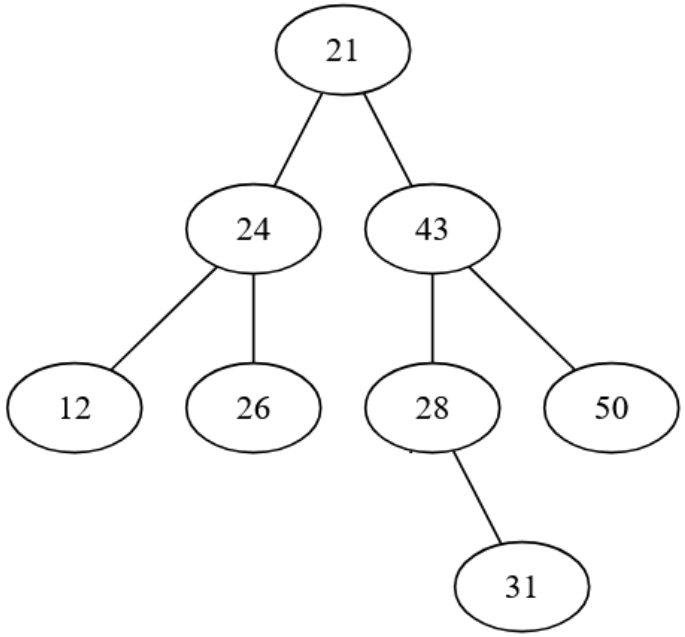
- Consider ADT TwoStacks, which contains two stacks together. It should support the following operations, all in $\Theta(1)$ (amortized) complexity:
  - push(ts, elem, stackNr) - If stackNr $= 1$, pushes an element to the first stack. If stackNr $= 2$, pushes an element to the second stack. Otherwise it throws an exception.
  - pop(ts, stackNr) - If stackNr $= 1$, pops and returns an element from the first stack. If stackNr $= 2$, pops and returns an element from the second stack. If stackNr has another value or the chosen stack is empty, it throws an exception.
  - top(ts, stackNr) - If stackNr $= 1$, returns the top element from the first stack. If stackNr $= 2$, returns the top element from the second stack. If stackNr has another value or the chosen stack is empty, it throws an exception.
  - isEmpty(ts, stackNr) - If stackNr $= 1$, returns whether the first stack is empty. If stackNr $= 2$, returns whether the second stack is empty. If stackNr has another value, it throws an exception.

Implementation problems II

- Describe how would you implement ADT TwoStacks using only one array. Describe the implementation of every operation and explain why it matches the required complexity.
- Give the representation of ADT TwoStacks and implement operation *push*.

Implementation problems III

- Write a subalgorithm that finds the maximum value in a binary tree containing integer numbers.
- Give the representation of the binary tree (do not memorize the parent of a node).
- Implement the subalgorithm.
- Specify and explain the complexity of the operation.
- If you use auxiliary containers specify the used operations.
- For example, for the following binary tree, the maximum value is 50.



Implementation problems IV

- Consider ADT Quartiler, which contains integer numbers and has the following operations (with the provided complexity requirements):
  - init(q) - creates a new, empty Quartiler - $\Theta(1)$ - total complexity
  - add(q, elem) - add a new element to the Quartiler q - $O(log_2 n) - amortized$
  - getTopQuartile(q) - returns the element closest to the 75th percentile (explanations below). If there is no such element, the operations throws an exception - $\Theta(1)$ - total complexity
  - deleteTopQuartile(q) - removes the element closest to the 75th percentile. If there is no such element, it throws ans exception - $O(log_2 n)$ - total complexity

Implementation problems V

- **Explanation:** the 75th percentile (called 3rd quartile as well) of a sequence is a value from the sequence, the one below which 75% of the sequence's values are found, if the sequence is sorted. So, if you have the values from 1 to 100 (in any order) the 75th percentile is the value 75. If you have the values 3, 1,2,4, the 75th percentile is 3. In case of a tie, any value can be returned, for example if you have the values 1, 2, 3, 4, 5, 6 either 5 or 6 can be returned.

Implementation problems VI

1. Which data structure (out of the ones discussed during the lectures) would you use as a representation for the Quartiler and how?
2. Explain in short how would you implement each operation od the Quartiler and why the implementation fits the complexity requirement.
3. Give the representation of the Quartiler and implement in pseudocode the deleteTopQuartile operation.

- **Obs:**
  1. You can ignore memory deallocation altogether when computing the complexity of an operation. You can assume that you can deallocate anything (including a linked list with dynamic allocation, for example) in $\Theta(1)$ complexity.
  2. You do not have to implement resize.
  3. You need to implement every function you need for question 3, you cannot assume that you already have operations implemented (unless specified differently). But you can define (and implement) auxiliary functions.