DATA STRUCTURES AND ALGORITHMS

LECTURE 14

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș - Bolyai University
Computer Science and Mathematics Faculty

2019 - 2020

In Lecture 13...

- AVL Trees
- Problems/Applications

Today

- Problems/Applications
- Conclusions
- Exam rules
- Mock-up exam solutions

Think about it

- Assume you have a binary tree, but you do not know how it looks like, but you have the *preorder* and *inorder* traversal of the tree. Give an algorithm for building the tree based on these two traversals.

- For example:
  - Preorder: A B F G H E L M
  - Inorder: B G F H A L E M

Think about it

- Can you rebuild the tree if you have the *postorder* and the *inorder* traversal?
- Can you rebuild the tree if you have the *preorder* and the *postorder* traversal?

Think about it

- For a Binary Tree we have seen that in some situations, if we have two tree traversals, we can rebuild the tree based on them.
- Can we do the same for a Binary Search Tree from one single traversal? Which one(s)?

Huffman coding

Huffman coding

- The *Huffman coding* can be used to encode characters (from an alphabet) using variable length codes.
- In order to reduce the total number of bits needed to encode a message, characters that appear more frequently have shorter codes.
- Since we use variable length code for each character, *no code can be the prefix of any other code* (if we encode letter E with 01 and letter X with 010011, during decoding, when we find a 01, we will not know whether it is E or the beginning of X).

Huffman coding

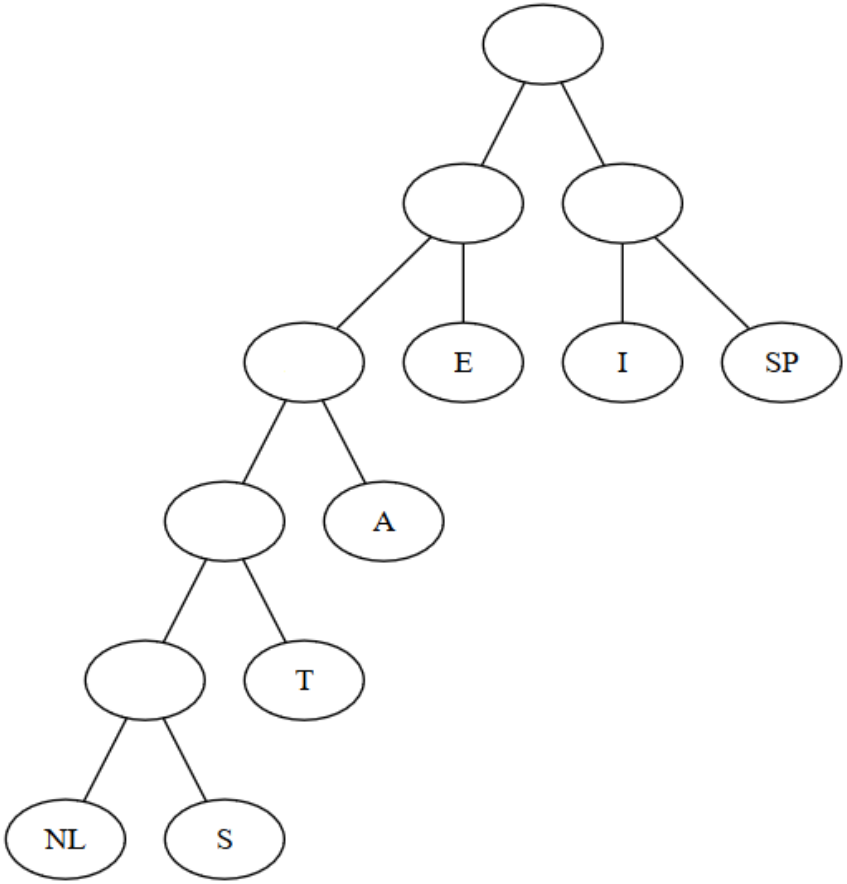
- When building the Huffman encoding for a message, we first have to compute the frequency of every character from the message, because we are going to define the codes based on the frequencies.
- Assume that we have a message with the following letters and frequencies

Character	a	e	i	s	t	space	newline
Frequency	10	15	12	3	4	13	1

Huffman coding

- For defining the Huffman code a binary tree is build in the following way:
  - Start with trees containing only a root node, one for every character. Each tree has a weight, which is frequency of the character.
  - Get the two trees with the least weight (if there is a tie, choose randomly), combine them into one tree which has as weight the sum of the two weights.
  - Repeat until we get have only one tree.

Huffman coding



Huffman coding

- Code for each character can be read from the tree in the following way: start from the root and go towards the corresponding leaf node. Every time we go left add the bit 0 to encoding and when we go right add bit 1.
- Code for the characters:
  - NL - 00000
  - S - 00001
  - T - 0001
  - A - 001
  - E - 01
  - I - 10
  - SP - 11
- In order to encode a message, just replace each character with the corresponding code

Huffman coding

- Assume we have the following code and we want to decode it: 01101100010001001110010000 0
- We do not know where the code of each character ends, but we can use the previously built tree to decode it.
- Start parsing the code and iterate through the tree in the following way:
  - Start from the root
  - If the current bit from the code is 0 go to the left child, otherwise go to the right child
  - If we are at a leaf node we have decoded a character and have to start over from the root
- The decoded message: E I SP T T A SP I E NL

Conclusions

- During the semester we have talked about the most important containers (ADT) and their main properties and operations
  - Bag, Set, Map, Multimap, List, Stack, Queue and their sorted versions
- We have also talked about the most important data structures that can be used to implement these containers
  - Dynamic array, Linked lists, Binary heap, Hash table, Binary Search Tree, AVL Tree
- We have mentioned some of the more advanced data structures as well
  - Skip Lists, Binomial Heaps, Cuckoo hashing, Perfect Hashing

Conclusions

- You should be able to identify the most suitable container for solving a given problem:

Conclusions

- You should be able to identify the most suitable container for solving a given problem:
- Example: *You have a type Student which has a name and a city. Write a function which takes as input a list of students and prints for each city all the students that are from that city. Each city should be printed only once and in any order.*
- How would you solve the problem? What container would you use?

Conclusions

- When you use containers existing in different programming languages, you should have an idea of how they are implemented and what is the complexity of their operations:

Conclusions

- When you use containers existing in different programming languages, you should have an idea of how they are implemented and what is the complexity of their operations:
- Consider the following algorithm (written in Python):

```
def testContainer(container, l):
 '''
container is a container with integer numbers
l is a list with integer numbers
'''
    count = 0
    for elem in l:
        if elem in container:
            count += 1
    return count
```

- The above function counts how many elements from the list *l* can be found in the container. What is the complexity of *testContainer*?

Conclusions

- Consider the following problem: *We want to model the content of a wallet, by using a list of integer numbers, in which every value denotes a bill. For example, a list with values [5, 1, 50, 1, 5] means that we have 62 RON in our wallet.*
  Obviously, we are not allowed to have any numbers in our list, only numbers corresponding to actual bills (we cannot have a value of 8 in the list, because there is no 8 RON bill).
  We need to implement a functionality to pay a given amount of sum and to receive rest of necessary.
  There are many optimal algorithms for this, but we go for a very simple (and non-optimal): keep removing bills of the wallet until the sum of removed bills is greater than or equal to the sum you want to pay.
  If we need to receive a rest, we will receive it in 1 RON bills.

Conclusions

- For example, if the wallet contains the values [5, 1, 50, 1, 5] and we need to pay 43 RON, we might remove the first 3 bills (a total of 56) and receive the 13 RON rest in 13 bills of 1.

Conclusions

- For example, if the wallet contains the values [5, 1, 50, 1, 5] and we need to pay 43 RON, we might remove the first 3 bills (a total of 56) and receive the 13 RON rest in 13 bills of 1.
- This is an implementation provided by a student. What is wrong with it?

```
public void spendMoney(ArrayList<Integer> wallet, Integer amount) {
    Integer spent = 0;
    while (spent < amount) {
        Integer bill = wallet.remove(0); //removes element from position 0
        spent += bill;
    }
    Integer rest = spent - amount;
    while (rest > 0) {
        wallet.add(0, 1);
        rest--;
    }
}
```

- Thank you for your participation at the mock-up exam!

Exam Rules

- Seminar attendance - at least 5 (marked in the attendance document) and Lab attendance (at least 6)
- There will be several Tencent meetings (more information and the exact link will besent before the exam)
- Make sure that you have your full name on Tencent (you have an option to rename yourself)
- Everyone will have to turn on their camera and be present in the Tencent call (otherwise their attempt will be removed)
- Questions (if something is not clear) will be asked only on Moodle (not in Tencent chat)

Exam Rules II

- For the "long problems" you will have to solve the problems on paper, take a picture of your work and upload it to Moodle.
- Every paper that you upload has to contain your name (handwritten). It might be easier if before the exam you write your name on every paper you intend to use.
- If you upload text files, screenshots, pictures of papers not containing your name, they will be ignored.
- For "short problems" you can upload at most 1 picture, for long ones you can upload at most 3 pictures. Each picture can have at most 1MB.
- All files need to be uploaded before the time runs out.

Exam Rules III

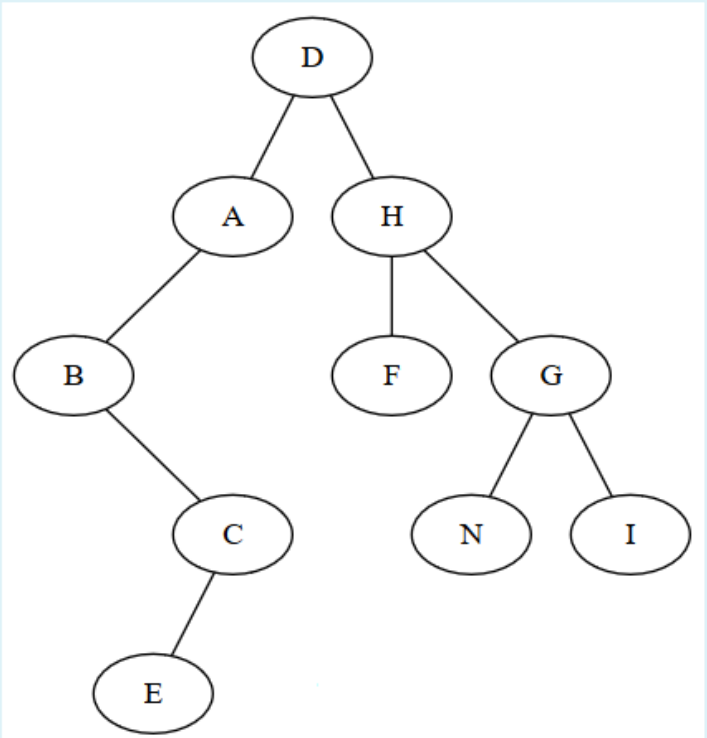
- The exam will be open book, meaning that you can consult the lecture notes during the exam.
- You are not allowed to communicate with other people during the exam. If you do, your exam grade will be 1!
- While this is an open book exam, I expect you to study for it and know things: the quantity of questions and problems is set in such a way that if you need to search for the answer for every questions you will not have time to solve many problems.
- The exact number of questions you will find out a the beginning of the exam.
- You will have only 1 attempt for this exam and you cannot return to a previous question.

Exam Rules IV

- If your lab grade is not at least 5, but you have the attendances, you can only participate at the exam in the retake session.
- In the retake session you will also have to present your labs, more information about it will be given at the end of the exam session.

Question 1

Specify the postorder traversal of the following binary tree:

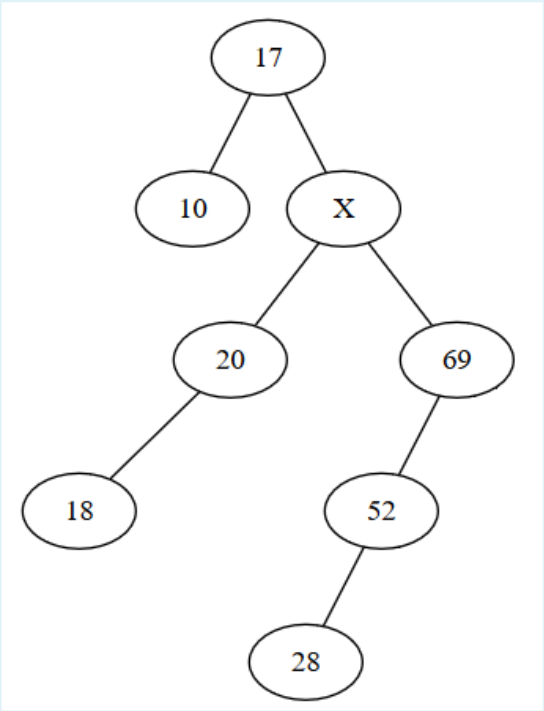


B H E C N I F A D G

- 71 correct answers (out of 119)
- Was it complicated to realize that it is drag and drop?

Question 2

Consider the binary search tree from the figure below. What of the following values could be in the node with X?



Select one or more:

- ☐ 23
- ☐ 40
- ☐ 15
- ☐ 19
- ☐ 30
- ☐ 26

- 60 correct answers

Question 3

Starting from an 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?

Select one or more:

- ☐ 5
- ☐ 6
- ☐ 2
- ☐ 1
- ☐ 3
- ☐ 4

- 92 correct answers

Question 4

What is the result of the following expression in postfix form made of single digit numbers and the regular operators (+, -, *, /):

4 6 7 + 1 - 2 5 * + +

- 66 correct answers

Question 5

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

- 72 correct answers

Question 6

What data structure is the most suitable representation for a List, where we will frequently want to get an element from a position p (getElement operation)?

Select one or more:

- ☐ dynamic array
- ☐ hash table
- ☐ doubly linked list
- ☐ binary heap
- ☐ singly linked list

- 75 correct answers

Question 7

To which complexity class does the following expression belong to: $12*n^3 + n*\log_2 n + 52$?

Select one or more:

- ☐ $O(n^4)$
- ☐ $O(n^2)$
- ☐ $\Theta(n*\log_2 n)$
- ☐ $\Omega(n^3)$
- ☐ $\Omega(1)$
- ☐ $\Theta(n^4)$

- 35 correct answers

Question 8

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

Answer: ✖

- 57 correct answers

Question 9

We have a hashtable with m = 8 positions in which separate chaining is used as a collision resolution method, but every position contains the root of an AVL tree.

Show how the following elements can be added into this hashtable (which is initially empty): 8, 99, 40, 19, 56, 16, 17, 28, 62. It is enough to draw the final hashtable, but show how you computed the position for every element.

Specify the load factor of the table.

- 56 uploaded files

Question 10

Starting from an initially empty binary search tree (built with the regular "≤" relation), insert 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.

Justify your answers.

- 70 uploaded files

Question 11

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).

Justify your answers.

- 46 uploaded files

Question 12

Write a function with $\Theta(n*\log_2 n)$ complexity. Explain why it has the required complexity.

Obs. Having a for loop going from 1 to the required complexity value is not going to get you any points.

- 46 uploaded files

Question 13

Deduce the complexity of the following function, called as p(x, 1, n).

```
subalgorithm p(x,s,d) is:
    if s < d then
        m ← [(s+d)/2]
        for i ← s, d-1, execute
            @elementary operation
        end-for
        for i ← 1,2 execute
            p(x, s, m)
        end-for
    end-if
end-subalgorithm
```

- 42 uploaded files

Question 14

Show how a stack can be implemented using two queues. You can assume that the Queue ADT is already implemented (we don't know how and we don't care how). You will have to implement the Stack ADT using the operations of the Queue.

Give the specifications for the operations of the Queue.

Give the representation of the Stack.

Implement ALL Stack operations.

Specify the complexity for every stack operation, assuming $\Theta(1)$ complexity for the queue operations.

- 23 uploaded files

Question 15

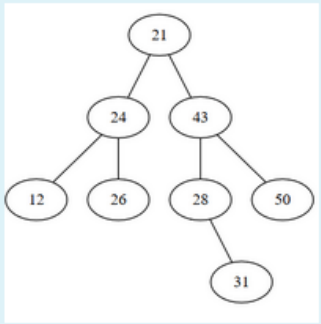
Write a subalgorithm that finds the maximum value in a binary tree containing integer numbers.

Give the representation of the binary tree

Implement the subalgorithm.

Specify and explain the complexity of the operation.

For example, in the binary tree below, the maximum value is 50.



- 33 uploaded files
- 16 did not read the question carefully

Question 16

Consider the following problem: we want to implement ADT Quartiler, which contains integer numbers and has the following operations (with the specified complexity requirements).

- init(q) - creates a new, empty Quartiler - $\Theta(1)$ - total complexity

- add(q, elem) - adds a new element to the Quartiler q - $O(\log_2 n)$ - amortized complexity

- getTopQuartile(q) - returns the element closest to the 75^{th} percentile (explanations below). If there is no such element, throws an exception - $\Theta(1)$ - total complexity

- deleteTopQuartile(q) - removes the element closest to the 75^{th} percentile. If there is no such element, throws an exception - $O(\log_2 n)$ - total complexity

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

1. Which data structure (out of the ones discussed during the lectures) would you use as a representation for the Quartiler?

2. Explain in short how would you implement each operation of 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.

- 11 uploaded files
- 5 Heap (1 perfect), 2 BST, 1 AVL, (3 Misc)