Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler, Ulysse Schaller

9. November 2020

Algorithms & Data Structures — Exercise sheet 8 — HS 20

Exercise Class (Room & TA): _____

Submitted by: _____

Peer Feedback by: _____

Points: ____

Submission: On Monday, 16 November 2020, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

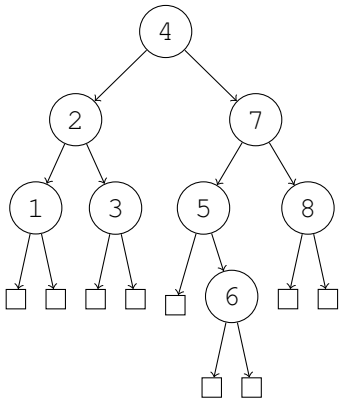
Exercise 8.1 *Search Trees* **(1 point)**.

a) Draw the resulting tree when the keys 1, 3, 6, 5, 4, 8, 7, 2 in this order are inserted into an initially empty binary (natural) search tree.

b) Delete key 8 in the above tree, and afterwards key 3 in the resulting tree.

c) Draw the resulting tree when the keys are inserted into an initially empty AVL tree. Give also the intermediate states before and after each rotation that is performed during the process.

d) Consider the following AVL tree:



Delete key 8 in this tree, and afterwards key 2 in the resulting tree. Give also the intermediate states before and after each rotation is performed during the process.

Exercise 8.2 *Finding an invariant (exercise 6.5. revisited).*

In this exercise we revisit exercise 5 from sheet 6, where we had a runner who wants to run from A to B in Fig. 1. There are two lanes available. One is represented by the first row and the other by the second row. On some sections, the first lane is faster than the second lane, and vice versa. The runner

A	1	3	2	1
3	2	1	1	B

Figure 1: Runner problem for a cost array of size 2×5.

can change lanes at any time, but this costs 1 minute every time. In this exercise, you are supposed to provide a dynamic programming algorithm that computes the optimal track.

Formally, the problem is defined in terms of a cost array $c \in \mathbb{N}^{2\times n}$. In Fig. 1 $n = 5$. Now, the runner starts at position $(1, 1)$ and wants to run to $(2, n)$. Running along a lane from field (i, j) to the field $(i, j + 1)$ requires $c_{i,j+1}$ minutes. Changing lanes from field $(1, j)$ to $(2, j)$ requires $1 + c_{2,j}$ minutes, and from field $(2, j)$ to $(1, j)$ requires $1 + c_{1,j}$ minutes.

Consider the following implementation of our DP solution for computing the minimal time required to get from A to B (note that in this implementation `previous` and `DP` are both one dimensional arrays with 2 elements):

Algorithm 1 `optimal_time`(c)

```
previous ← [0, 1 + c_{2,1}]
DP ← [0, 0]
for j = 2, ..., n do
    DP[1] = min(previous[1] + c_{1,j}, previous[2] + c_{2,j} + c_{1,j} + 1)
    DP[2] = min(previous[2] + c_{2,j}, previous[1] + c_{1,j} + c_{2,j} + 1)
    previous ← DP
    // Your invariant from a) must hold here.
return previous[2]
```

This implementation is more memory-efficient than the solution of 6.5. It only requires $O(1)$ additional memory. (It also has disadvantages: it only computes the optimal time,not the corresponding track.) Your task is to prove that this algorithm is correct.

a) Formulate an invariant $INV(j)$ that holds after the iteration with the running index j of the for loop.

b) Prove the correctness of the algorithm `optimal_time` by induction using your invariant.

Exercise 8.3 *Exponential bounds for a sequence defined inductively.*

Consider the sequence $(a_n)_{n\in\mathbb{N}}$ defined by

$$\begin{aligned} a_0 &= 1, \\ a_1 &= 1, \\ a_2 &= 2, \\ a_i &= a_{i-1} + 2a_{i-2} + a_{i-3} \quad \forall i \geq 3. \end{aligned}$$

The goal of this exercise is to find exponential lower and upper bounds for a_n.

a) Find a constant $C > 1$ such that $a_n \leq \mathcal{O}(C^n)$ and prove your statement.

b) Find a constant $c > 1$ such that $a_n \geq \Omega(c^n)$ and prove your statement.

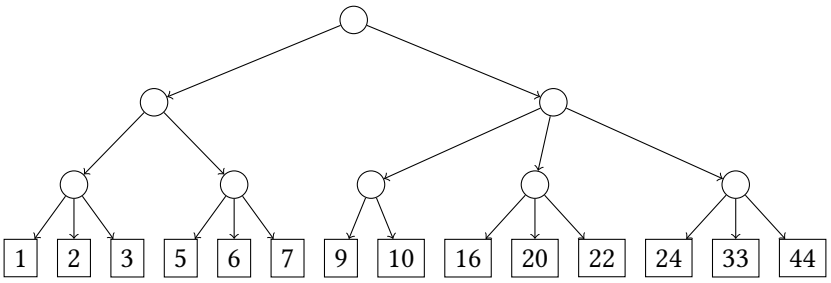
Remark. The solution of 6.2 has been updated, and contains now some intution on how to come up with a good inductive statement, which mayhelp for this exercise.

Exercise 8.4 *The (2, 3)-tree datastructure* **(2 points)**.

A (2,3)-tree is a tree in which each inner node has between 2 and 3 children. Additionally, all leaves are at the same depth. There are two different types of inner nodes:

1. **2-nodes**: A 2-node is a node with two children l and r of the same height.
2. **3-nodes**: A 3-node is a node with three children l, m and r of the same height.

The keys are stored in the leafs and ordered from left to right.



a) Come up with a way to augment the inner nodes with information that enables efficient search for keys. Provide pseudocode of your search method.

b) Find a way to insert a new key at the correct position. Your method should run in time $\mathcal{O}(h)$, where h is the height of the (2,3)-tree. Provide pseudocode of your insertion method. You may ignore the information in the inner nodes for this task, i.e., you don't need to describe how to update this information. Also, you don't need to discuss how exactly you would store the nodes. You can assume that from a node, you can access parent, siblings and children in constant time.

Hint: *When the parent of the new leaf has more than 3 leafs after the insertion, split it into two. Be cautious, what happens to the 'grandparent'?*

c) Insert the key 8 into the example above and draw the result. Now, insert the key 11 into the tree from the previous step and draw the result. Finally, insert the key 4 into the tree from the previous step and draw the result.

d) Prove asymptotically matching upper and lower bounds for the height h of a (2,3)-tree with n leaves. "Asymptotically matching" means that they are of the form $h = \mathcal{O}(f(n))$ and $h = \Omega(f(n))$ for the same function $f(n)$.