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

11. November 2019

Algorithms & Data Structures — Exercise sheet 8 — HS 19

Exercise Class (Room & TA): ________________

Submitted by: ________________

Peer Feedback by: ________________

Points: ____

Submission: On Monday, 18 November 2019, 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.*

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

b) Delete key 4 in the above tree, and afterwards key 7 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) Delete key 7 in the above tree, and afterwards key 8 in the resulting tree.

***Exercise 8.2** *Maximum Depth Difference of two Leaves.*

Consider an AVL tree of height h. What is the maximum possible difference of the depths of two leaves? Describe which structure such trees need to have, and draw examples of corresponding trees for every $h \in \{2, 3, 4\}$. Derive a recursive formula (depending on h), solve it and use induction to prove the correctness of your solution. Provide a detailed explanation of your considerations.

Hint: *For the proof the principle of complete induction can be used. Let $\mathcal{A}(n)$ be a statement for a number $n \in \mathbb{N}$. If, for every $n \in \mathbb{N}$, the validity of all statements $\mathcal{A}(m)$ for $m \in \{1, \dots, n-1\}$ implies the validity of $\mathcal{A}(n)$, then $\mathcal{A}(n)$ is true for every $n \in \mathbb{N}$. Thus, complete induction allows multiple base cases and inductive hypotheses.*

***Exercise 8.3** *Fibonacci numbers.*

The *Fibonacci numbers* are defined by $F(0) := 0$, $F(1) := 1$, and $F(n) := F(n-1) + F(n-2)$ for $n \geq 2$. Prove that for all $n \geq 0$,

$$F_n = \frac{1}{\sqrt{5}} \left(\phi^n - \psi^n\right),$$

where $\phi = (1 + \sqrt{5})/2$ and $\psi = (1 - \sqrt{5})/2$. Show that this formula implies $F(n) = \Theta(\phi^n)$.

Hint: *You can solve the problem inductively by direct (and nasty) calculation. If you want to understand where the formula comes from, you should proceed by the following steps.*

1. *Show that ϕ and ψ are the roots of the equation $x^2 = x + 1$.*
2. *Show that if a real number x satisfies $x^2 = x + 1$, then for all $n \geq 2$ it satisfies $x^n = x^{n-1} + x^{n-2}$.* Hint: This is trivial!
3. *Consider the set $V := \{f : \mathbb{N}_0 \to \mathbb{R} \mid f(n) = f(n-1) + f(n-2)$ for all $n \geq 2\}$. Show that V forms a vector space, and that the two functions $G(n) := \phi^n$ and $H(n) := \psi^n$ are in V. In fact, it is easy to see (you don't need to formally prove this, but give an intuitive argument) that V has dimension 2, and that G and H are independent. So they form a basis.*
4. *Conclude that there must be two constants $a, b \in \mathbb{R}$ such that $F = a \cdot G + b \cdot H$.*
5. *Compute a and b by looking at the special cases $F(0) = a \cdot G(0) + b \cdot H(0)$ and $F(1) = a \cdot G(1) + b \cdot H(1)$.*
6. *Conclude that $F_n = \frac{1}{\sqrt{5}}(\phi^n - \psi^n)$ for all $n \geq 0$.*

Exercise 8.4 *Online supermarket* **(1 point)**.

Assume that you work in a large online supermarket that offers different types of goods. At every moment you have to know the number of goods of each type that the supermarket currently offers. Let us denote the number of goods of type t by S_t. At any moment S_t can either be decreased (if someone has bought some goods of type t) or increased (if some goods of type t have been delivered from the manufacturer). Also your boss can ask you how many goods of type t does the supermarket currently offer. So you can receive three types of queries: to decrease S_t by $0 < x \leq S_t$, to increase S_t by $x > 0$ or to return S_t.

Assume that at each moment number of different types of goods that the supermarket offers at that moment is bounded by $n > 0$, but the number of types of goods that the supermarket can potentially offer might be much larger than n. Consider the following example: $n = 3$, at 14:00 the supermarket can offer 5 balls, 1 doll and 4 phones and at 14:15 it can offer 6 balls, 3 chairs and 12 pencils.

Provide an algorithm that can handle each query in time $\mathcal{O}(\log n)$. You may assume that initially all S_t are zero.

Exercise 8.5 *Nim game* **(2 points)**.

Consider the following game in which two players move alternately: at the beginning there are 2 piles of stones with n_1 and n_2 stones, respectively. During a move a player chooses a non-empty pile and an integer $k > 0$, and removes $s = k^2$ stones from the chose pile. (Obviously, s cannot be greater than the number of stones in the chosen pile). Player 1 makes the first move. A player who cannot move loses. For example, if all piles are empty at the very beginning, Player 1 loses.

a) Provide a *dynamic programming* algorithm that, given n_1 and n_2, determines which player wins the game if both play optimally. You can assume that arithmetic operations take unit time. You can obtain full points with a dynamic programming with a DP table of dimension 2.

   Address the following aspects in your solution:

   (a) *Dimension of the DP table: What is the dimension of the table $DP[\ldots]$?* What range do you have in each dimension?

   (b) *Definition of the DP table*: What is the meaning of each entry?

(c) *Computation of an entry*: How can an entry be computed from the values of other entries? Specify the base cases, i.e., the entries that do not depend on others.

(d) *Calculation order*: In which order can entries be computed so that values needed for each entry have been determined in previous steps?

(e) *Extracting the solution*: How can the final solution be extracted once the table has been filled?

(f) *Running time*: What is the runtime of your solution?[1] Does the algorithm have polynomial runtime?

Specifically, you can use the following scheme:

Dimensions of the table:

Meaning of a table entry (in words):

Computation of an entry (initialization and recursion):

Order of computation:

Computing the output:

Running time in $\mathcal{O}$-notation in terms of n_1 and n_2:

b) Finally, assume you have already filled out the DP table, and that the current player (say, it is Player 1) is in a winning position. Describe how you can find a winning strategy, i.e., how you can determine a move such that after her turn, Player 2 is in a *losing* position.

[1]By this formulation, we expect you to specify the worst case runtime in Θ-notation.