ALGORITHMEN & DATENSTRUKTUREN - ÜBUNGSSTUNDE 6

Exercise 5.2 *Guessing an interval.*

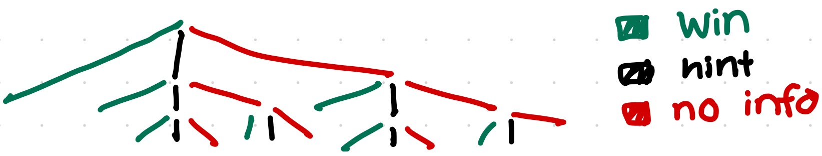
Alice and Bob play the following game:

- Alice selects two integers $1 \leq a < b \leq 200$, which she keeps secret.
- Then, Alice and Bob repeat the following:
  - Bob chooses two integers $0 \leq a' < b' \leq 201$.
  - If $a = a'$ and $b = b'$, Bob wins.
  - If $a' < a$ and $b < b'$, Alice tells Bob 'my numbers are strictly between your numbers!'.
  - Otherwise, Alice does not give any clue to Bob.

Bob claims that he has a strategy to win this game in 12 attempts at most. Prove that such a strategy cannot exist.

Hint: *Represent Bob's strategy as a decision tree. Each edge of the decision tree corresponds to one of Alice's answers, while each leaf corresponds to a win for Bob.*

Hint: *After defining the decision tree, you can show that there is at most one leaf for every non-leaf node and the number of non-leaf nodes is at most $2^n - 1$ for a tree of depth n for $n \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$.*



#choices for Alice = $200 \cdot 201/2 - 200$

$$\sum_{i=1}^{200} \sum_{j=i+1}^{200} 1$$

winning leaves for Bob in 12 guesses

$$\sum_{i=0}^{11} 2^i = 2^{12} - 1 = 4095$$

Exercise 5.4 *Bubble sort invariant* **(1 point).**

Consider the pseudocode of the bubble sort algorithm on an integer array $A[1,\dots,n]$:

Algorithm 3 BUBBLESORT(A)

---

for $1 \le j < n$ **do**
    for $1 \le i < n$ **do**
        if $A[i] > A[i+1]$ **then**
            $t \leftarrow A[i]$
            $A[i] \leftarrow A[i+1]$
            $A[i+1] \leftarrow t$
return A

(a) Formulate an invariant INV(j) that holds at the end of the j-th iteration of the outer for-loop.

(b) Using the invariant from part (a), prove the correctness of the algorithm. Specifically, prove the following three assertions:

(1) INV(1) holds.

(2) If INV(j) holds, then INV$(j+1)$ holds (for all $1 \le j < n$).

(3) INV(n) implies that BUBBLESORT(A) correctly sorts the array A.

Hint: For the induction step in part (2), suppose INV(j) holds, then observe how the largest element in $A[1,\dots,n-j]$ (at the end of the jth iteration) moves throughout the $(j+1)$th iteration of the outer loop. Where would this element need to be to located in the array in order to satisfy INV$(j+1)$?

a) INV(j): After j iterations of the outer for Loop, the subarray $A[n-j+1,\dots,n]$ is sorted and each element from $A[1,\dots,n-j]$ is not greater than each element from $A[n-j+1,\dots,n]$

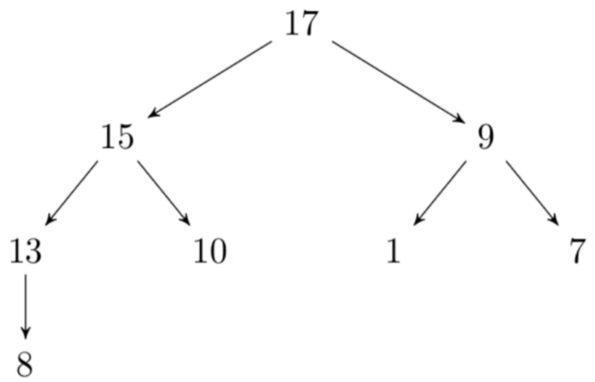
b) (1): INV(1) holds: after first iteration largest element is at position n. Assume largest element at position j in the beginning. In the j'th iteration $A[j] > A[j+1]$ will become true. we therefore swap the 2 elements. For all future iterations of the inner loop the condition is true and the largest element gets bubbled up to position n.

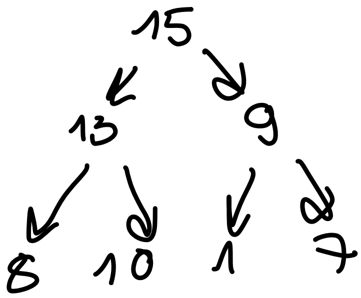
(2): INV$(j) \Rightarrow$ INV$(j+1)$, $\forall\, 1 \le j \le n$: Assume INV(j) holds we now use the same reasoning as for (1) that if the "$j+1$ largest element" is at position $x \ne A[n-(j+1)+1]$ we swap it until it is there in the inner loop

(3): INV$(n) \Rightarrow$ Array gets sorted correctly; INV(n) means $A[\underbrace{n-n+1}_{=1},\dots,n]$ is sorted correctly
$\Rightarrow$ whole array is sorted correctly

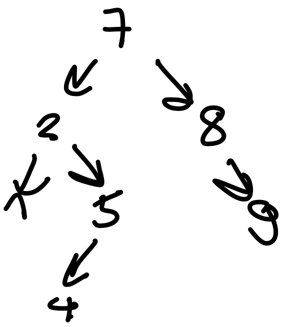
ALGORITHMEN & DATENSTRUKTUREN - ÜBUNGSSTUNDE 6

ii) Draw the Max-Heap obtained from the following Max-Heap by performing the operation `DELETE-MAX` once.

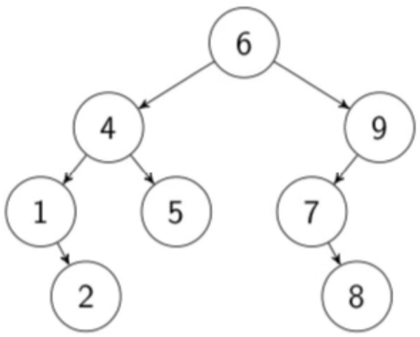


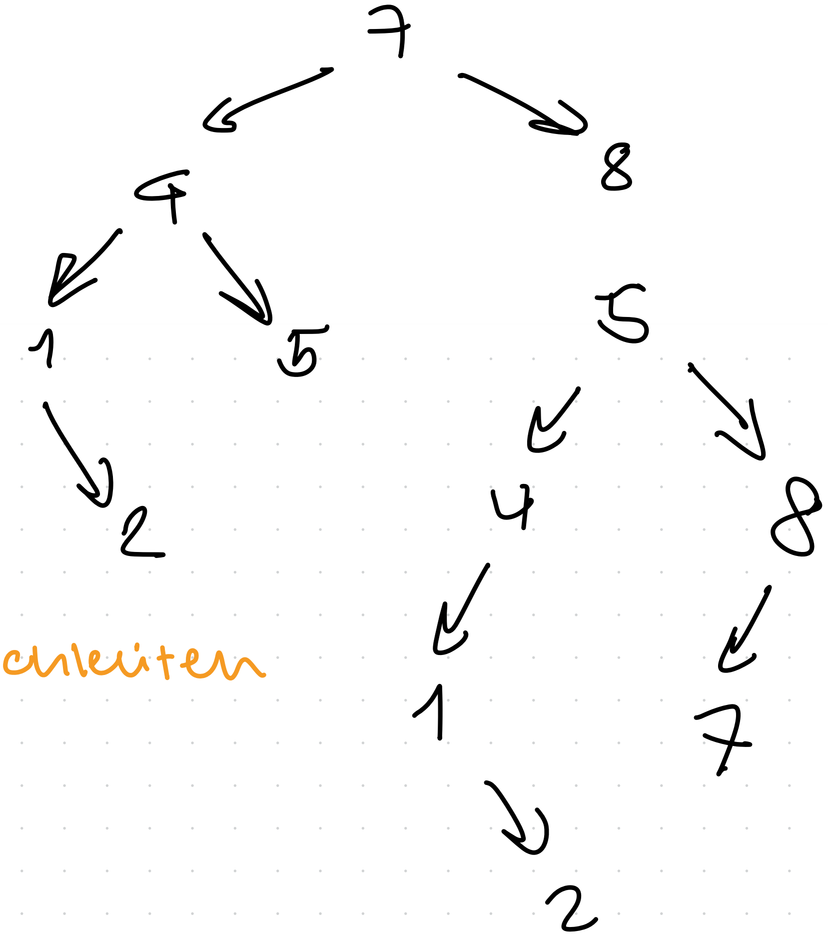


g) *Binary search trees*: Draw the binary search tree that is obtained when inserting into an empty tree the keys 7, 2, 1, 5, 4, 8, 9 in this order.



h) *Binary search trees*: Draw the resulting binary search tree obtained by deleting the keys 6 and 9 in this order from the following binary search tree.





mehrere möglichkeiten
richtig