UNIVERSITY OF BIRMINGHAM

School of Computer Science

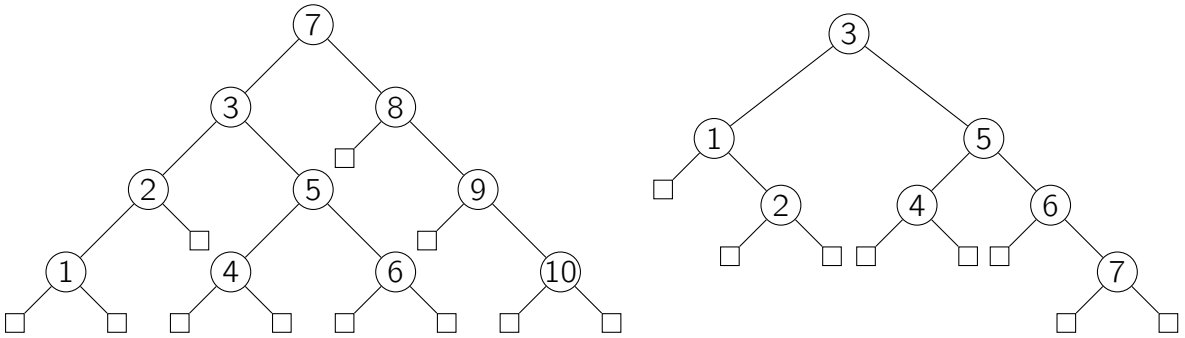
Data Structures, Algorithms and Databases

Class Test #1 [2022-23]

23th Feb, 2023

Data Structures, Algorithms and Databases

Question 1



Part 1 Consider the two trees above and answer the following two questions. Briefly motivate your answers.

(a) Determine which of these two trees is perfectly balanced. **[2 marks]**

(b) Determine which of these two trees is an AVL tree. **[2 marks]**

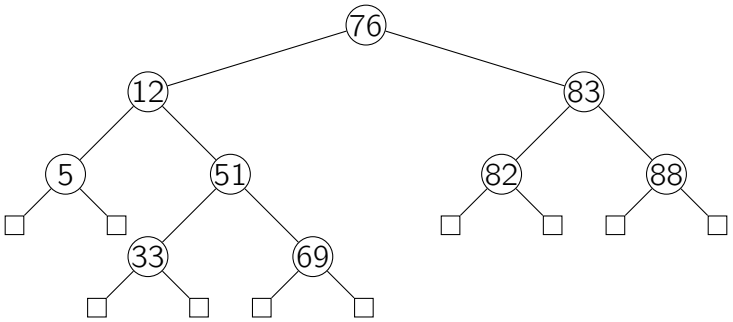
Part 2 Consider a general variant of binary search trees that only admits nodes with balance -2, -1, 0, 1, or 2. Answer the following questions about the size of such trees.

(c) What is the maximum possible number of nodes of a tree with height 6? **[2 marks]**

(d) What is the minimum possible number of nodes of a tree with height 6? **[2 marks]**

Hint: think about how to construct the largest and the smallest tree with this property; it is not required to construct the entire tree to answer this question.

Question 2



(a) Consider the AVL tree above and delete the element with value 76. **[2 marks]**

(b) Consider the AVL tree above and insert an element with value 40 **[2 marks]**

Note that the two questions above are independent of one another. Each of these two operations should be performed on the original tree above. Show the tree resulting from the operation first, and then the tree resulting from the rotations steps (when necessary) afterwards.

Question 3

Consider a procedure that visits the elements of a tree and constructs a list with these elements. Let us define a function `visit`:

$$\texttt{visit}(\square) = [\,]$$
$$\texttt{visit}(\text{Fork}(x, l, r)) = \texttt{visit}(r) + [x] + \texttt{visit}(l)$$

where $[\ldots]$ denotes a list of elements. In particular, the special case $[\,]$ denotes the empty list and operation $+$ denotes concatenation of lists. For example, $[3, 4, 1] + [2, 3, 5]$ results in $[3, 4, 1, 2, 3, 5]$. Consider a function named `doSomething` that takes an array of integers a with n elements (that is, `a.length` $= n$) and first inserts these elements in a binary search tree and then constructs a list using the function `visit`.

```
List doSomething(int [] a) {
  Tree T = □;
  for (int i = 0; i < a.length; i++) {
    assert(#nodes(T) == i); // invariant
    T = insert(T, A[i]);
  }
  List L = visit(T);
  assert(length(L) == a.length); // postcondition
  return L;
}
```

(Note that function `#nodes(T)` returns the number of nodes in tree T and function `length(L)` returns the number of elements in list L).

(a) Briefly describe the task this algorithm performs. Be precise about the content of the output list `L` in relation to the input array `a`. **[2 marks]**

(b) The invariant and the postconditon given in this program do **not always** hold true. Under what condition would they hold true? **[3 marks]**

(c) Give the wost case time complexity of `doSomething` in term of the input size n = `a.length`. Be precise about the kind of binary search tree you refer to when motivating your answer. **[3 marks]**