Algorithms and Data Structures
Jacobs University Bremen
Dr. Florian Rabe

Homework 7
given: 2017-03-21

Homework 7

You have to submit your solutions as announced in the lecture.
Unless mentioned otherwise, all problems are due 2017-03-30, 11:00.
There will be no deadline extensions unless mentioned otherwise in the lecture.

Problem 7.1 *Heap-Backed Priority Queue*

Points: 8

Implement $Heap[int, \geq]$ in any programming language.

Use it to implement $PriorityQueue[int]$ where the priority of each element is the element itself.

Depending on your programming language, this may look like

class $MaxHeap()$
    private $elements :=$ the underlying data structure backing the heap, e.g., a binary tree

    fun $insert(x : int) : unit =$
        ...
    fun $extract() : Option[int] =$
        ...
    fun $find() : Option[A] =$
        ...

class $IntPriorityQueue()$
    private $elements :=$ **new** $MaxHeap()$ the underlying heap backing the queue
    fun $enqueue(x : int) : unit =$
        $elements.insert(x)$
    fun $dequeue() : Option[int] =$
        $elements.extract()$

Write a test program that

- creates a new priority queue
- enqueues some values into it
- dequeues all values and prints them

Problem 7.2 *Trees, DFS/BFS*

Points: 3+6+6

Implement a data structure for $Tree[A]$.

Implement two functions $Tree[A] \to Iterator[A]$ that return the nodes of a tree in

1. DFS order
2. BFS order

One way to do this is to use the functions from the lecture notes to build the list of all nodes in the tree and then to turn the list into an iterator. That is bad because it forces traversing the entire tree. The right wy to do it is to build the iterator in such a way that the next node is only visited when *next* is called on the iterator. The functions given in the lecture can be adapted in this way.

Problem 7.3

Points: 5

Prove the theorem in the lecture notes about the number of nodes in a perfect binary tree