Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

SS 23
18. 7. 2023

Functional Data Structures

Exercise Sheet 13

The following are old exam questions!

Exercise 13.1 Amortized Complexity

A "stack with multipop" is a list with the following two interface functions:

fun *push* :: *"'a ⇒ 'a list ⇒ 'a list"* **where**
"push x xs = x # xs"

fun *pop* :: *"nat ⇒ 'a list ⇒ 'a list"* **where**
"pop n xs = drop n xs"

You may assume

definition *T_push* :: *"'a ⇒ 'a list ⇒ nat"* **where**
"T_push x xs = 1"

definition *T_pop* :: *"nat ⇒ 'a list ⇒ nat"* **where**
"T_pop n xs = min n (length xs)"

Use the potential method to show that the amortized complexity of *push* and *pop* is constant.

If you need any properties of the auxiliary functions *length*, *drop* and *min*, you should state them but you do not need to prove them.

Exercise 13.2 Converting List for Balanced Insert

Recall the standard insertion function for unbalanced binary search trees.

```
fun insert :: "'a::linorder ⇒ 'a tree ⇒ 'a tree" where
"insert x Leaf = Node Leaf x Leaf" |
"insert x (Node l a r) =
  (case cmp x a of
     LT ⇒ Node (insert x l) a r |
     EQ ⇒ Node l a r |
     GT ⇒ Node l a (insert x r))"
```

We define the function *from_list*, which inserts the elements of a list into an initially empty search tree:

definition *from_list* :: *“'a::linorder list ⇒ 'a tree”* **where**
“from_list l = fold insert l Leaf”

Your task is to specify a function *preprocess*::*'a*, that preprocesses the list such that the resulting tree is almost complete.

You may assume that the list is sorted, distinct, and has exactly *2^k − 1* elements for some *k*. That is, your *preprocess* function must satisfy:

fun *preprocess* :: *“'a list ⇒ 'a list”*

lemma
assumes *“sorted l”*
and *“distinct l”*
and *“length l = 2^k−1”*
shows *“set (preprocess l) = set l”* **and** *“acomplete (from_list (preprocess l))”*

Note: No proofs required, only a specification of the *preprocess* function!