Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Simon Roßkopf

SS 24
19. 7. 2024

Functional Data Structures

Exercise Sheet 13

Exercise 13.1 Amortized Complexity

Consider the following (very abstract) model of a dynamic table. We represent it as two natural numbers, one representing the capacity, one how many slots are filled. Of course, the capacity always has to be greater or equal to the number of elements in there.

```
type_synonym table = "nat × nat"

fun inv :: "table ⇒ bool" where
  "inv (n,c) ⟷ n ≤ c"
```

There is only one operation, insertion. It increases the number of filled slots, and, if capacity is reached, doubles the capacity:

```
fun ins :: "table ⇒ table" where
  "ins (n,c) = (n+1, if n < c then c else 2∗c)"
```

Give (by hand) a suitable timing function for this operation and (informally) justify its choice:

```
fun C_ins :: "table ⇒ nat"
```

Give a potential function Φ, so that *ins* runs in constant amortized time:

```
fun Φ :: "table ⇒ nat"
```

The following is an old exam question!

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!