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

SS 24
5. 7. 2024

Functional Data Structures

Exercise Sheet 11

Exercise 11.1 Insert for Leftist Heap

- Define a function to directly insert an element into a leftist heap. Do not construct an intermediate heap like insert via merge does!
- Show that your function is correct
- Define a timing function for your insert function, and show that it is linearly bounded by the minimum height of the tree.

fun *lh_insert* :: *"'a::ord ⇒ 'a lheap ⇒ 'a lheap"*

lemma *set_lh_insert*: *"set_tree (lh_insert x t) = set_tree t ∪ {x}"*
lemma *heap_lh_insert*: *"heap t ⟹ heap (lh_insert x t)"*
lemma *ltree_lh_insert*: *"ltree t ⟹ ltree (lh_insert x t)"*
time_fun *lh_insert*

lemma *"ltree t ⟹ T_lh_insert x t ≤ min_height t + 1"*

Exercise 11.2 Bootstrapping a Priority Queue

Given a generic priority queue implementation with $O(1)$ *empty*, *is_empty* operations, $O(f_1\ n)$ insert, and $O(f_2\ n)$ *get_min* and *del_min* operations.
Derive an implementation with $O(1)$ *get_min*, and the asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing about f_1 and f_2, you must not use *get_min*/*del_min* in your new *insert* operation, and vice versa!

For technical reasons, you have to define the new implementations type outside the locale!

datatype *('a,'s) bs_pq* =

locale *Bs_Priority_Queue* =
  orig: *Priority_Queue* **where**

```
    empty = orig_empty and
    is_empty = orig_is_empty and
    insert = orig_insert and
    get_min = orig_get_min and
    del_min = orig_del_min and
    invar = orig_invar and
    mset = orig_mset
  for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
  and orig_mset :: "'s ⇒ 'a::linorder multiset"
begin
```

In here, the original implementation is available with the prefix *orig*, e.g.

```
  term orig_empty term orig_invar
  thm orig.invar_empty

  definition empty :: "('a,'s) bs_pq"
  fun is_empty :: "('a,'s) bs_pq ⇒ bool"
  fun insert :: "'a ⇒ ('a,'s) bs_pq ⇒ ('a,'s) bs_pq"
  fun get_min :: "('a,'s) bs_pq ⇒ 'a"
  fun del_min :: "('a,'s) bs_pq ⇒ ('a,'s) bs_pq"
  fun invar :: "('a,'s) bs_pq ⇒ bool"
  fun mset :: "('a,'s) bs_pq ⇒ 'a multiset"
  lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
    orig.mset_insert orig.mset_empty
    orig.invar_empty orig.invar_insert orig.invar_del_min
```

Show that your new implementation satisfies the priority queue interface!

```
  sublocale Priority_Queue
    where empty = empty
    and is_empty = is_empty
    and insert = insert
    and get_min = get_min
    and del_min = del_min
    and invar = invar
    and mset = mset
    apply unfold_locales
  proof goal_cases
```

Homework 11.1 Be Creative!

Submission until Thursday, July 11, 23:59pm.

Develop a nice Isabelle formalisation yourself!

- You may develop a formalisation from all areas, not only functional data structures. Creative topics are encouraged!
- Document your solution well, such that it is clear what you have formalised and what your main theorems state!

- Set yourself a time frame and some intermediate/minimal goals. Your formalisation needs not be universal and complete.
- You are encouraged to discuss the realisability of your project with us!
- In total, the homework will yield 15 points (for minimal solutions). Additionally, bonus points may be awarded for particularly nice/original/etc solutions.
- Finish your project this week. It does not need to be polished or completely finished, but the main points should be there!
- To submit, use the submission system if you have a single file. **Submitting is sufficient, ignore any errors that the submission system may raise when checking the submission.** If the project is more than one file, send an archive by e-mail.