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

SS 2022
15. 7. 2022

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 rank 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 t ⟹ heap (lh_insert x t)"*
lemma *"ltree t ⟹ ltree (lh_insert x t)"*

fun *T_lh_insert* :: *"'a::ord ⇒ 'a lheap ⇒ nat"*

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

Exercise 11.2 Sparse Binary Numbers

Implement operations carry, inc, and add on sparse binary numbers, analogously to the operations ins_tree, insert, and merge on binomial heaps.

type_synonym *rank* = *nat*
type_synonym *snat* = *"rank list"*

abbreviation *invar* :: *"snat ⇒ bool"* **where** *"invar s ≡ sorted_wrt (<) s"*
definition *α* :: *"snat ⇒ nat"* **where** *"α s = sum_list (map ((^) 2) s)"*

lemmas [*simp*] = *sorted_wrt_append*

fun *carry* :: *"rank ⇒ snat ⇒ snat"*
definition *inc* :: *"snat ⇒ snat"*
fun *add* :: *"snat ⇒ snat ⇒ snat"*

Show correctness! Start with basic properties of the abstraction function, then copy the proofs from *HOL−Data_Structures.Binomial_Heap* and adapt them.

Now show that the operations have logarithmic worst-case complexity. For simplicity, user power of two instead of log.

Homework 11.1 Be Original!

Submission until Thursday, 21. 7. 2022, 23:59pm.
Develop a nice Isabelle formalisation yourself!

- Finish your formalization this week.
- The homework will yield 15 points (for minimal solutions). Additionally, up to 10 bonus points may be awarded for particularly nice/original/etc solutions.
- You may develop a formalisation from all areas, not only functional data structures.
- 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!
- If you can't think of a topic on your own, here are a few suggestions (though they will score lower on creativity): 1-2 trees (deletion, height), sparse matrices, skew binary numbers, arbitrary precision arithmetic (on lists of bits), interval lists (extension to those of the tutorial), spatial data structures (quad-trees, oct-trees), Fibonacci heaps, etc.
- Use the submission system to submit your result as a single Isabelle theory, under the same task as before. This submission does not need to (and in fact, won't be able to) get the status "passed". If you need multiple files (e.g., for generated code), send the tutor an e-mail instead.