Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich

SS 2018
29. 6. 2018

Functional Data Structures

Exercise Sheet 12

Exercise 12.1 Sparse Binary Numbers

Implement operations carry, inc, and add on sparse binary numbers, analogously to the operations link, ins, and meld on binomial heaps.

Show that the operations have logarithmic worst-case complexity.

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

abbreviation *invar* :: *"snat ⇒ bool"* **where** *"invar s ≡ sorted_wrt op< s"*
definition α :: *"snat ⇒ nat"* **where** "$\alpha\ s = (\sum i{\leftarrow}s.\ 2\hat{\ }i)$"

lemmas [*simp*] = *sorted_wrt_Cons sorted_wrt_append*

fun *carry* :: *"rank ⇒ snat ⇒ snat"*
lemma *carry_invar*[*simp*]:
lemma *carry_*α:

definition *inc* :: *"snat ⇒ snat"*
lemma *inc_invar*[*simp*]: *"invar rs ⟹ invar (inc rs)"*
lemma *inc_*α[*simp*]: "*invar rs* ⟹ α (*inc rs*) = *Suc* (α *rs*)"
fun *add* :: *"snat ⇒ snat ⇒ snat"*
lemma *add_invar*[*simp*]:
  assumes "*invar* rs_1"
  assumes "*invar* rs_2"
  shows "*invar* (*add* rs_1 rs_2)"
lemma *add_*α[*simp*]:
  assumes "*invar* rs_1"
  assumes "*invar* rs_2"
  shows "α (*add* rs_1 rs_2) = α rs_1 + α rs_2"

lemma *size_snat*:
  assumes *"invar rs"*
  shows "$2\hat{\ }$*length rs* ≤ α *rs* + *1*"
fun *t_carry* :: *"rank ⇒ snat ⇒ nat"*
definition *t_inc* :: *"snat ⇒ nat"*
lemma *t_inc_bound*:

assumes *"invar rs"*
shows *"t_inc rs* $\leq$ *log 2* $(\alpha$ *rs* $+$ *1)* $+$ *1"*
fun *t_add* :: *"snat* $\Rightarrow$ *snat* $\Rightarrow$ *nat"*
lemma *t_add_bound*:
fixes rs_1 rs_2
defines "$n_1 \equiv \alpha\ rs_1$"
defines "$n_2 \equiv \alpha\ rs_2$"
assumes *INVARS*: *"invar* rs_1*"* *"invar* rs_2*"*
shows *"t_add* rs_1 rs_2 $\leq$ *4**log 2* $(n_1 + n_2 + 1) + 2$*"*

Homework 12 Explicit Priorities

Submission until Friday, 6.7.2018, 11:59am.

Modify the priority queue interface to handle multisets of pairs of data and priority, i.e., the new *mset* function has the signature $mset::'q \Rightarrow ('d \times 'a::linorder)$ *multiset*.

Next, implement the new interface using leftist heaps. No complexity proofs are required.

Hints:

- Start with the existing theories, and modify them!
- Be careful to design a good specification for *get_min*!

Homework 12.1 Be Creative!

Submission until Friday, 13. 7. 2017, 11:59am.

Develop a nice Isabelle formalization yourself!

- This homework goes in parallel to other homeworks for the rest of the lecture period.
- This homework will yield 15 points (for minimal solutions). Additionally, up to 15 bonus points may be awarded for particularly nice/original/etc solutions.
- You may develop a formalization from all areas, not only functional data structures.
- Document your solution, such that it is clear what you have formalized and what your main theorems state!
- Set yourself a time frame and some intermediate/minimal goals. Your formalization needs not be universal and complete after 3 weeks.
- You are welcome to discuss the realizability of your project with the tutor!
- In case you should need inspiration to find a project: Sparse matrices, skew binary numbers, arbitrary precision arithmetic (on lists of bits), interval data structures (e.g. interval lists), spatial data structures (quad-trees, oct-trees), Fibonacci heaps, prefix tries/arrays and BWT, etc.