Script generated by TTT

Title: FDS (26.07.2019)

Date: Fri Jul 26 08:33:23 CEST 2019

Duration: 97:40 min

Pages: 43

20 Amortized Complexity

21 Skew Heap

22 Splay Tree

23 Pairing Heap

24 More Verified Data Structures and Algorithms (in Isabelle/HOL)

Implementation type

datatype $'a\ heap = Empty \mid Hp\ 'a\ ('a\ heap\ list)$

Implementation type

datatype $'a\ heap = Empty \mid Hp\ 'a\ ('a\ heap\ list)$

Heap invariant:

$pheap\ Empty = True$
$pheap\ (Hp\ x\ hs) =$
$(\forall\, h{\in}set\ hs.\ (\forall\, y{\in}set\_heap\ h.\ x \leq y) \wedge pheap\ h)$

Implementation type

datatype $'a\ heap = Empty \mid Hp\ 'a\ ('a\ heap\ list)$

Heap invariant:

$pheap\ Empty = True$
$pheap\ (Hp\ x\ hs) =$
$(\forall\, h{\in}set\ hs.\ (\forall\, y{\in}set\_heap\ h.\ x \leq y) \wedge pheap\ h)$

Also: *Empty* must only occur at the root

insert

$insert\ x\ h = merge\ (Hp\ x\ [])\ h$

insert

$insert\ x\ h = merge\ (Hp\ x\ [])\ h$

$merge\ h\ Empty = h$
$merge\ Empty\ h = h$

insert

$insert\ x\ h = merge\ (Hp\ x\ [])\ h$

$merge\ h\ Empty = h$
$merge\ Empty\ h = h$
$merge\ (Hp\ x\ lx =: hx)\ (Hp\ y\ ly =: hy) =$
(if $x < y$ then $Hp\ x\ (hy\ \#\ lx)$ else $Hp\ y\ (hx\ \#\ ly)$)

insert

$insert\ x\ h = merge\ (Hp\ x\ [])\ h$

$merge\ h\ Empty = h$
$merge\ Empty\ h = h$
$merge\ (Hp\ x\ lx =: hx)\ (Hp\ y\ ly =: hy) =$
$(\text{if } x < y \text{ then } Hp\ x\ (hy\ \#\ lx) \text{ else } Hp\ y\ (hx\ \#\ ly))$

Like function *link* for binomial heaps

del_min

$del\_min\ Empty = Empty$
$del\_min\ (Hp\ x\ hs) = pass_2\ (pass_1\ hs)$

del_min

$del\_min\ Empty = Empty$
$del\_min\ (Hp\ x\ hs) = pass_2\ (pass_1\ hs)$

$pass_1\ [] = []$
$pass_1\ [h] = [h]$
$pass_1\ (h_1\ \#\ h_2\ \#\ hs) = merge\ h_1\ h_2\ \#\ pass_1\ hs$

del_min

$del\_min\ Empty = Empty$
$del\_min\ (Hp\ x\ hs) = pass_2\ (pass_1\ hs)$

$pass_1\ [] = []$
$pass_1\ [h] = [h]$
$pass_1\ (h_1\ \#\ h_2\ \#\ hs) = merge\ h_1\ h_2\ \#\ pass_1\ hs$

$pass_2\ [] = Empty$
$pass_2\ (h\ \#\ hs) = merge\ h\ (pass_2\ hs)$

Fusing $pass_2 \circ pass_1$

$merge\_pairs\ [] = Empty$
$merge\_pairs\ [h] = h$
$merge\_pairs\ (h_1\ \#\ h_2\ \#\ hs) =$
$merge\ (merge\ h_1\ h_2)\ (merge\_pairs\ hs)$

Functional correctness proofs

Straightforward

Analysis

Analysis easier (more uniform) if a pairing heap is viewed as a binary tree:

Analysis

Analysis easier (more uniform) if a pairing heap is viewed as a binary tree:

$homs :: 'a\ heap\ list \Rightarrow 'a\ tree$
$homs\ [] = \langle\rangle$
$homs\ (Hp\ x\ hs_1\ \#\ hs_2) = \langle homs\ hs_1,\ x,\ homs\ hs_2\rangle$

Analysis

Analysis easier (more uniform) if a pairing heap is viewed as a binary tree:

$$homs :: 'a\ heap\ list \Rightarrow 'a\ tree$$
$$homs\ [] = \langle\rangle$$
$$homs\ (Hp\ x\ hs_1\ \#\ hs_2) = \langle homs\ hs_1,\ x,\ homs\ hs_2\rangle$$

$$hom :: 'a\ heap \Rightarrow 'a\ tree$$
$$hom\ Empty = \langle\rangle$$
$$hom\ (Hp\ x\ hs) = \langle homs\ hs,\ x,\ \langle\rangle\rangle$$

Analysis

Analysis easier (more uniform) if a pairing heap is viewed as a binary tree:

$$homs :: 'a\ heap\ list \Rightarrow 'a\ tree$$
$$homs\ [] = \langle\rangle$$
$$homs\ (Hp\ x\ hs_1\ \#\ hs_2) = \langle homs\ hs_1,\ x,\ homs\ hs_2\rangle$$

$$hom :: 'a\ heap \Rightarrow 'a\ tree$$
$$hom\ Empty = \langle\rangle$$
$$hom\ (Hp\ x\ hs) = \langle homs\ hs,\ x,\ \langle\rangle\rangle$$

Potential function same as for splay trees

Analysis

Analysis easier (more uniform) if a pairing heap is viewed as a binary tree:

$$homs :: 'a\ heap\ list \Rightarrow 'a\ tree$$
$$homs\ [] = \langle\rangle$$
$$homs\ (Hp\ x\ hs_1\ \#\ hs_2) = \langle homs\ hs_1,\ x,\ homs\ hs_2\rangle$$

Analysis of *splay*

Theorem

$$[\![bst\ t;\ \langle l,\ a,\ r\rangle \in subtrees\ t]\!]$$
$$\Longrightarrow a\_splay\ a\ t \leq 3 * (\varphi\ t - \varphi\ \langle l,\ a,\ r\rangle) + 1$$

Corollary

$$[\![bst\ t;\ a \in set\_tree\ t]\!]$$
$$\Longrightarrow a\_splay\ a\ t \leq 3 * (\varphi\ t - 1) + 1$$

Verified:

The functions $insert$, del_min and $merge$ all have $O(\log_2 n)$ amortized complexity.

These bounds are not tight.

Verified:

The functions $insert$, del_min and $merge$ all have $O(\log_2 n)$ amortized complexity.

These bounds are not tight.
Better amortized bounds in the literature:
$insert \in O(1)$, $del\_min \in O(\log_2 n)$, $merge \in O(1)$

297

Archive of Formal Proofs

`https://www.isa-afp.org/entries/Amortized_Complexity.shtml`

Sources

The inventors of the pairing heap:

M. Fredman, R. Sedgewick, D. Sleator and R. Tarjan.
The Pairing Heap: A New Form of Self-Adjusting Heap.
Algorithmica, 1986.

20 Amortized Complexity

21 Skew Heap

22 Splay Tree

23 Pairing Heap

24 More Verified Data Structures and Algorithms (in Isabelle/HOL)

More trees

- Huffman Trees:

More trees

- Huffman Trees:
  Huffman 1952 / Blanchette 2008

Graph algorithms

Graph algorithms

- Floyd-Warshall:
  Floyd 1962, Warshall 1962 /
  Wimmer and Lammich 2017

Graph algorithms

- Floyd-Warshall:
  Floyd 1962, Warshall 1962 /
  Wimmer and Lammich 2017
- Shortest Path:
  Dijkstra 1956 / Nordhoff and Lammich 2012

Graph algorithms

- Floyd-Warshall:
  Floyd 1962, Warshall 1962 /
  Wimmer and Lammich 2017
- Shortest Path:
  Dijkstra 1956 / Nordhoff and Lammich 2012
- Maximum Flow:
  Ford-Fulkerson 1955 / Lammich and Sefidgar 2016

Graph algorithms

- Floyd-Warshall:
  Floyd 1962, Warshall 1962 /
  Wimmer and Lammich 2017
- Shortest Path:
  Dijkstra 1956 / Nordhoff and Lammich 2012
- Maximum Flow:
  Ford-Fulkerson 1955 / Lammich and Sefidgar 2016
- Strongly Connected Components:
  Tarjan 1972 / Schimpf 2015

Graph algorithms

- Floyd-Warshall:
  Floyd 1962, Warshall 1962 /
  Wimmer and Lammich 2017
- Shortest Path:
  Dijkstra 1956 / Nordhoff and Lammich 2012
- Maximum Flow:
  Ford-Fulkerson 1955 / Lammich and Sefidgar 2016
- Strongly Connected Components:
  Tarjan 1972 / Schimpf 2015
  Gabow 2000 / Lammich 2014
- Minimum spanning tree:
  Kruskal 1956, Prim 1957 /
  Guttmann 2018, Lammich *et al.* 2019

Model Checkers

- SPIN-like LTL Model Checker:
  Esparza, Lammich, Neumann, Nipkow, Schimpf,
  Smaus 2013

Dynamic programming

- Start with recursive function

Dynamic programming

- Start with recursive function
- Automatic translation to memoized version incl. correctness theorem

Dynamic programming

- Start with recursive function
- Automatic translation to memoized version incl. correctness theorem
- Applications
  - Optimal binary search tree
  - Minimum edit distance
  - Bellman-Ford (SSSP)
  - CYK
  - ...

Infrastructure

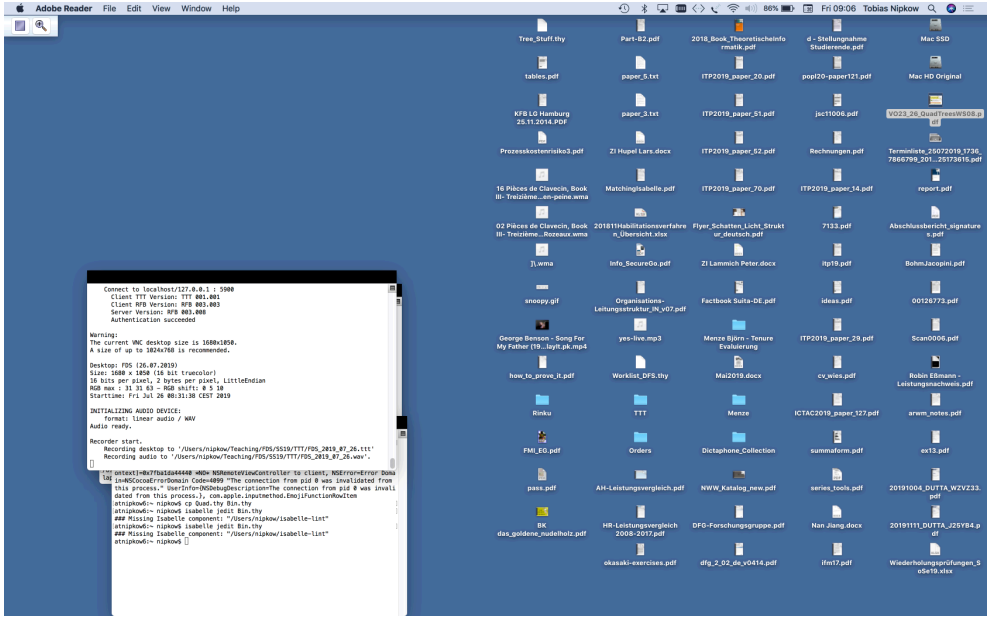
Refinement Frameworks by Lammich:

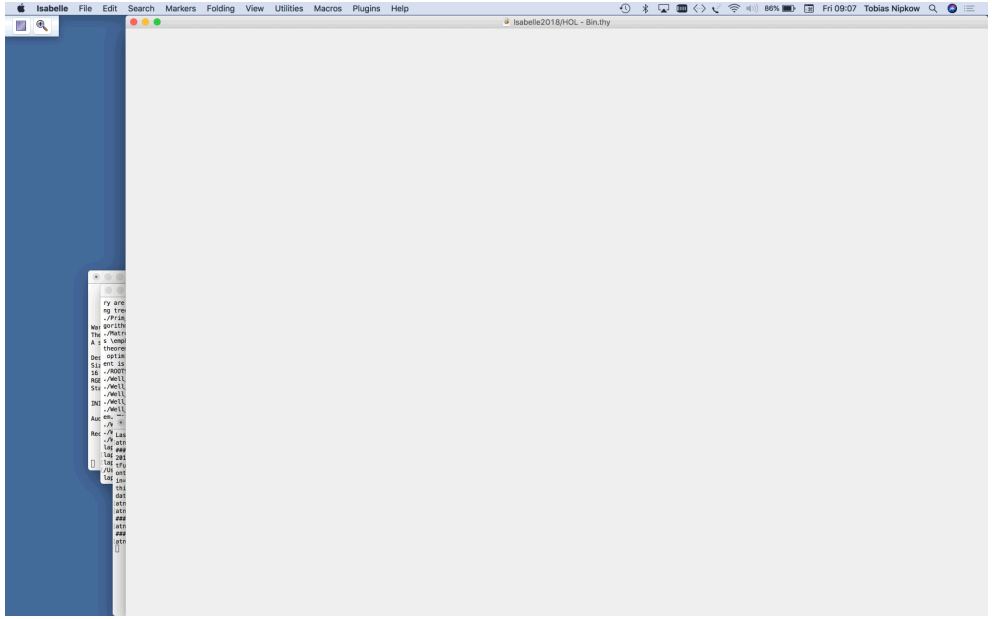
Infrastructure

Refinement Frameworks by Lammich:

Abstract specification
$\rightsquigarrow$ functional program
$\rightsquigarrow$ imperative program

Mostly in the Archive of Formal Proofs


Adobe Reader File Edit View Window Help
Fri 09:06 Tobias Nipkow
Tree_Stuff.thy
Part-B2.pdf
2018_Book_TheoretischeInformatik.pdf
d - Stellungnahme Studierende.pdf
Mac SSD
tables.pdf
paper_5.txt
ITP2019_paper_20.pdf
pop20-paper121.pdf
Mac HD Original
KFB LG Hamburg 25.11.2014.PDF
paper_3.txt
ITP2019_paper_51.pdf
jsc11006.pdf
VO23_26_QuadTreesWS08.pdf
Prozesskostenrisiko3.pdf
ZI Hupel Lars.docx
ITP2019_paper_52.pdf
Rechnungen.pdf
Terminliste_25072019_1736_7866799_201...25173615.pdf
16 Pièces de Clavecin, Book III- Treizième...en-peine.wma
Matchinglsabelle.pdf
ITP2019_paper_70.pdf
ITP2019_paper_14.pdf
report.pdf
02 Pièces de Clavecin, Book III- Treizième...Rozeaux.wma
201811Habilitationsverfahren_Übersicht.xlsx
Flyer_Schatten_Licht_Struktur_deutsch.pdf
7133.pdf
Abschlussbericht_signatures.pdf
].wma
Info_SecureGo.pdf
ZI Lammich Peter.docx
itp19.pdf
BohmJacopini.pdf
snoopy.gif
Organisations-Leitungsstruktur_IN_v07.pdf
Factbook Suita-DE.pdf
ideas.pdf
00126773.pdf
George Benson - Song For My Father (19...layit.pk.mp4
yes-live.mp3
Menze Björn - Tenure Evaluierung
ITP2019_paper_29.pdf
Scan0006.pdf
how_to_prove_it.pdf
Worklist_DFS.thy
Mai2019.docx
cv_wies.pdf
Robin EBmann - Leistungsnachweis.pdf
Rinku
TTT
Menze
ICTAC2019_paper_127.pdf
arwm_notes.pdf
FMI_EG.pdf
Orders
Dictaphone_Collection
summaform.pdf
ex13.pdf
pass.pdf
AH-Leistungsvergleich.pdf
NWW_Katalog_new.pdf
series_tools.pdf
20191004_DUTTA_WZVZ33.pdf
BK das_goldene_nudelholz.pdf
HR-Leistungsvergleich 2008-2017.pdf
DFG-Forschungsgruppe.pdf
Nan Jiang.docx
20191111_DUTTA_J25Y84.pdf
okasaki-exercises.pdf
dfg_2_02_de_v0414.pdf
ifm17.pdf
Wiederholungsprüfungen_SoSe19.xlsx
Connect to localhost/127.0.0.1 : 5900
Client TTT Version: TTT 003.001
Client RFB Version: RFB 003.003
Server Version: RFB 003.008
Authentication succeeded
Warning:
The current VNC desktop size is 1680x1050.
A size of up to 10244788 is recommended.
Desktop: FDS (26.07.2019)
Size: 1680 x 1050 (16 bit truecolor)
16 bits per pixel, 2 bytes per pixel, LittleEndian
RGB max : 31 31 63 - RGB shift: 0 5 10
Starttime: Fri Jul 26 00:31:38 CEST 2019
INITIALIZING AUDIO DEVICE:
format: linear audio / WAV
Audio ready.
Recorder start.
Recording desktop to '/Users/nipkow/Teaching/FDS/SS19/TTT/FDS_2019_07_26.ttt'
Recording audio to '/Users/nipkow/Teaching/FDS/SS19/TTT/FDS_2019_07_26.wav'.
atnipkow6:~ nipkow$ cp Quad.thy Bin.thy
atnipkow6:~ nipkow$ isabelle jedit Bin.thy
Missing Isabelle component: "/Users/nipkow/isabelle-lint"
atnipkow6:~ nipkow$ isabelle jedit Bin.thy
Missing Isabelle component: "/Users/nipkow/isabelle-lint"
atnipkow6:~ nipkow$


Isabelle File Edit Search Markers Folding View Utilities Macros Plugins Help
Fri 09:07 Tobias Nipkow
isabelle2018/HOL - Bin.thy