

Script generated by TTT

Title: Lammich: FDS (29.06.2018)

Date: Fri Jun 29 08:35:57 CEST 2018

Duration: 86:22 min

Pages: 60

15 Priority Queues

16 Leftist Heap

17 Priority Queue via Braun Tree

18 Binomial Heap

19 Skew Binomial Heap

Chapter 9

Priority Queues

Implementation type

datatype

'a lheap = Leaf | Node nat ('a tree) 'a ('a tree)

Archive of Formal Proofs

https://www.isa-afp.org/entries/Priority_Queue_Braun.shtml

195

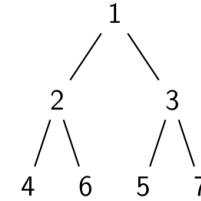
Idea of invariant maintenance

$braun \langle \rangle = True$
 $braun \langle l, x, r \rangle =$
 $(|r| \leq |l| \wedge |l| \leq |r| + 1 \wedge braun \ l \wedge braun \ r)$

197

What is a Braun tree?

$braun :: 'a \ tree \Rightarrow \ bool$
 $braun \langle \rangle = True$
 $braun \langle l, x, r \rangle =$
 $(|r| \leq |l| \wedge |l| \leq |r| + 1 \wedge braun \ l \wedge braun \ r)$



196

Priority queue implementation

Implementation type: $'a \ tree$

198

del_min

del_min :: 'a tree ⇒ 'a tree

200

del_min

del_min :: 'a tree ⇒ 'a tree

del_min ⟨⟩ = ⟨⟩

del_min ⟨⟨⟩, x, r⟩ = ⟨⟩

del_min ⟨l, x, r⟩ =

(let (y, l') = *del_left* l in *sift_down* r y l')

- 1 Delete leftmost element *y*
- 2 Sift *y* from the root down

200

sift_down

sift_down :: 'a tree ⇒ 'a ⇒ 'a tree ⇒ 'a tree

sift_down ⟨⟩ a ⟨⟩ = ⟨⟨⟩, a, ⟨⟩⟩

sift_down ⟨⟨⟩, x, ⟨⟩⟩ a ⟨⟩ =

(if $a \leq x$ then ⟨⟨⟨⟩, x, ⟨⟩⟩, a, ⟨⟩⟩

else ⟨⟨⟨⟩, a, ⟨⟩⟩, x, ⟨⟩⟩)

sift_down ⟨⟨l₁, x₁, r₁⟩ =: t₁⟩ a ⟨⟨l₂, x₂, r₂⟩ =: t₂⟩ =

if $a \leq x_1 \wedge a \leq x_2$ then ⟨t₁, a, t₂⟩

else if $x_1 \leq x_2$ then ⟨*sift_down* l₁ a r₁, x₁, t₂⟩

else ⟨t₁, x₂, *sift_down* l₂ a r₂⟩

Maintains *braun*

202

Sorting with priority queue

pq [] = *empty*

pq (x#xs) = *insert* x (*pq* xs)

mins q =

(if *is_empty* q then []

else *get_min* h # *mins* (*del_min* h))

206

Sorting with priority queue

$pq [] = empty$

$pq (x\#xs) = insert\ x\ (pq\ xs)$

$mins\ q =$

(if $is_empty\ q$ then $[]$)

else $get_min\ h\ \#\ mins\ (del_min\ h)$)

$sort_pq = mins\ o\ pq$

206

HOL/Data_Structures/
Binomial_Heap.thy

208

15 Priority Queues

16 Leftist Heap

17 Priority Queue via Braun Tree

18 Binomial Heap

19 Skew Binomial Heap

207

Numerical method

Idea: only use trees t_i of size 2^i

Example

To store (in binary) 11001 elements: $[t_0, 0, 0, t_3, t_4]$

209

Numerical method

Idea: only use trees t_i of size 2^i

Example

To store (in binary) 11001 elements: $[t_0, 0, 0, t_3, t_4]$

Merge \approx addition with carry

209

Numerical method

Idea: only use trees t_i of size 2^i

Example

To store (in binary) 11001 elements: $[t_0, 0, 0, t_3, t_4]$

Merge \approx addition with carry

Needs function to combine two trees of size 2^i
into one tree of size 2^{i+1}

209

Binomial tree

datatype 'a tree =

Node (rank: nat) (root: 'a) ('a tree list)

210

Binomial tree

datatype 'a tree =

Node (rank: nat) (root: 'a) ('a tree list)

Invariant: Node of rank r has children $[t_{r-1}, \dots, t_0]$
of ranks $[r-1, \dots, 0]$

210

Binomial tree

```
datatype 'a tree =  
  Node (rank: nat) (root: 'a) ('a tree list)
```

Invariant: Node of rank r has children $[t_{r-1}, \dots, t_0]$
of ranks $[r-1, \dots, 0]$

```
invar_btree (Node r x ts) =  
(( $\forall t \in \text{set } ts. \text{invar\_btree } t$ )  $\wedge$  map rank ts = rev [0.. $r$ ])
```

210

Binomial tree

```
datatype 'a tree =  
  Node (rank: nat) (root: 'a) ('a tree list)
```

Invariant: Node of rank r has children $[t_{r-1}, \dots, t_0]$
of ranks $[r-1, \dots, 0]$

```
invar_btree (Node r x ts) =  
(( $\forall t \in \text{set } ts. \text{invar\_btree } t$ )  $\wedge$  map rank ts = rev [0.. $r$ ])
```

Lemma

```
invar_btree t  $\implies$  |t| =  $2^{\text{rank } t}$ 
```

210

Combining two trees

How to combine two trees of rank i
into one tree of rank $i+1$

```
link (Node r x1 ts1 =: t1) (Node r' x2 ts2 =: t2) =  
(if  $x_1 \leq x_2$  then Node (r + 1) x1 (t2 # ts1)  
 else Node (r + 1) x2 (t1 # ts2))
```

211

Binomial heap

Use sparse representation for binary numbers:
 $[t_0, 0, 0, t_3, t_4]$ represented as $[(0, t_0), (3, t_3), (4, t_4)]$

212

Combining two trees

How to combine two trees of rank i
into one tree of rank $i+1$

```
link (Node r x1 ts1 =: t1) (Node r' x2 ts2 =: t2) =  
(if x1 ≤ x2 then Node (r + 1) x1 (t2 # ts1)  
 else Node (r + 1) x2 (t1 # ts2))
```

211

Inserting a tree

```
ins_tree t [] = [t]  
ins_tree t1 (t2 # ts) =  
(if rank t1 < rank t2 then t1 # t2 # ts  
 else ins_tree (link t1 t2) ts)
```

213

Binomial heap

Use sparse representation for binary numbers:
[$t_0, 0, 0, t_3, t_4$] represented as [(0, t_0), (3, t_3), (4, t_4)]

type_synonym 'a heap = 'a tree list

212

merge

```
merge ts1 [] = ts1  
merge [] ts2 = ts2  
merge (t1 # ts1) (t2 # ts2) =  
(if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)  
 else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2  
 else ins_tree (link t1 t2) (merge ts1 ts2))
```

214

merge

```
merge ts1 [] = ts1
merge [] ts2 = ts2
merge (t1 # ts1) (t2 # ts2) =
  (if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)
   else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2
   else ins_tree (link t1 t2) (merge ts1 ts2))
```

Intuition: Addition of binary numbers

214

Get/delete minimum element

All trees are min-heaps.

Smallest element may be any root node:

```
ts ≠ [] ⇒ get_min ts = Min (set (map root ts))
```

Similar:

```
get_min_rest :: 'a tree list ⇒ 'a tree × 'a tree list
```

Returns tree with minimal root, and remaining trees

215

Get/delete minimum element

All trees are min-heaps.

Smallest element may be any root node:

```
ts ≠ [] ⇒ get_min ts = Min (set (map root ts))
```

215

Get/delete minimum element

All trees are min-heaps.

Smallest element may be any root node:

```
ts ≠ [] ⇒ get_min ts = Min (set (map root ts))
```

Similar:

```
get_min_rest :: 'a tree list ⇒ 'a tree × 'a tree list
```

Returns tree with minimal root, and remaining trees

```
del_min ts =
```

```
(case get_min_rest ts of
```

```
  (Node r x ts1, ts2) ⇒ merge (rev ts1) ts2)
```

Why *rev*?

215

Get/delete minimum element

All trees are min-heaps.

Smallest element may be any root node:

$ts \neq [] \implies \text{get_min } ts = \text{Min } (\text{set } (\text{map } \text{root } ts))$

Similar:

$\text{get_min_rest} :: 'a \text{ tree list} \Rightarrow 'a \text{ tree} \times 'a \text{ tree list}$

Returns tree with minimal root, and remaining trees

$\text{del_min } ts =$

$(\text{case } \text{get_min_rest } ts \text{ of}$

$(\text{Node } r \ x \ ts_1, \ ts_2) \Rightarrow \text{merge } (\text{rev } ts_1) \ ts_2)$

Why *rev*? Rank decreasing in ts_1 but increasing in ts_2

215

Complexity

Recall: $|t| = 2^{\text{rank } t}$

Similarly for heap: $2^{\text{length } ts} \leq |ts| + 1$

216

Complexity

Recall: $|t| = 2^{\text{rank } t}$

Similarly for heap: $2^{\text{length } ts} \leq |ts| + 1$

Complexity of operations: linear in length of heap

216

Complexity

Recall: $|t| = 2^{\text{rank } t}$

Similarly for heap: $2^{\text{length } ts} \leq |ts| + 1$

Complexity of operations: linear in length of heap
i.e., logarithmic in number of elements

216

Complexity of *merge*

```
merge (t1 # ts1) (t2 # ts2) =  
(if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)  
 else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2  
  else ins_tree (link t1 t2) (merge ts1 ts2))
```

217

Complexity of *merge*

```
merge (t1 # ts1) (t2 # ts2) =  
(if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)  
 else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2  
  else ins_tree (link t1 t2) (merge ts1 ts2))
```

Complexity of *ins_tree*: $t_{ins_tree} t ts \leq length ts + 1$

A call *merge* t₁ t₂ (where $length t_1 = length t_2 = n$) can lead to calls of *ins_tree* on lists of length 1, ..., n.

$\Sigma \in O(n^2)$

217

Complexity of *merge*

```
merge (t1 # ts1) (t2 # ts2) =  
(if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)  
 else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2  
  else ins_tree (link t1 t2) (merge ts1 ts2))
```

Relate time and length of input/output:

$t_{ins_tree} t ts + length (ins_tree t ts) = 2 + length ts$

$length (merge ts_1 ts_2) + t_{merge} ts_1 ts_2$

$\leq 2 * (length ts_1 + length ts_2) + 1$

Yields desired linear bound!

218

Sources

The inventor of the binomial heap:

Jean Vuillemin.

A Data Structure for Manipulating Priority Queues.
CACM, 1978.

219

Complexity of *merge*

```
merge (t1 # ts1) (t2 # ts2) =  
(if rank t1 < rank t2 then t1 # merge ts1 (t2 # ts2)  
 else if rank t2 < rank t1 then t2 # merge (t1 # ts1) ts2  
  else ins_tree (link t1 t2) (merge ts1 ts2))
```

Relate time and length of input/output:

$$t_ins_tree\ t\ ts + length\ (ins_tree\ t\ ts) = 2 + length\ ts$$
$$length\ (merge\ ts_1\ ts_2) + t_merge\ ts_1\ ts_2$$
$$\leq 2 * (length\ ts_1 + length\ ts_2) + 1$$

Yields desired linear bound!

218

- 15 Priority Queues
- 16 Leftist Heap
- 17 Priority Queue via Braun Tree
- 18 Binomial Heap
- 19 Skew Binomial Heap

220

Skew Binomial Heap

Similar to binomial heap, but involving also
skew binary numbers:

222

Skew Binomial Heap

Similar to binomial heap, but involving also
skew binary numbers:

$d_1 \dots d_n$ represents $\sum_{i=1}^n d_i * (2^{i+1} - 1)$
where $d_i \in \{0, 1, 2\}$

222

Complexity

Skew binomial heap:

insert in time $O(1)$
del_min and merge still $O(\log n)$

223

Complexity

Skew binomial heap:

insert in time $O(1)$
del_min and merge still $O(\log n)$

223

Complexity

Skew binomial heap:

insert in time $O(1)$
del_min and merge still $O(\log n)$

Fibonacci heap (imperative!):

insert and merge in time $O(1)$
del_min still $O(\log n)$

Every operation in time $O(1)$?

223

Puzzle

Design a functional queue
with (worst case) constant time *enq* and *deq* functions

224

Puzzle

Design a functional queue with (worst case) constant time *enq* and *deq* functions

224

Complexity

Skew binomial heap:

insert in time $O(1)$
del_min and *merge* still $O(\log n)$

Fibonacci heap (imperative!):

insert and *merge* in time $O(1)$
del_min still $O(\log n)$

Every operation in time $O(1)$?

223

The terminal window shows the following commands and output:

```

lammich@lappnikow10: ~/lehre/FDS/SS18/public
File Edit View Search Terminal Help
d by any service files
Az
[1]+ Stopped evince slides-fds.pdf
lammich@lappnikow10: ~/lehre/FDS/SS18/public$ bg
[1]+ evince slides-fds.pdf &
lammich@lappnikow10: ~/lehre/FDS/SS18/public$ isabelle jedit ~/opt/Isabelle2017/s
rc/HOL/Data
Data_Structures/ Datatype_Examples/
lammich@lappnikow10: ~/lehre/FDS/SS18/public$ isabelle jedit ~/opt/Isabelle2017/s
rc/HOL/Data_Structures/
AA_Map.thy      lsn2.thy      Sorting.thy
AA_Set.thy      Leftist_Heap.thy  Tree234_Map.thy
AList_UpDown.thy  Less_False.thy  Tree234_Set.thy
AVL_Map.thy     List_Ins_Del.thy  Tree234.thy
AVL_Set.thy     Lookup2.thy      Tree23_Map.thy
Balance.thy     Map_by_Ordered.thy  Tree23_Set.thy
Base_FDS.thy    Priority_Queue.thy  Tree23.thy
Binomial_Heap.thy  RBT_Map.thy      Tree2.thy
Brother12_Map.thy  RBT_Set.thy      Tree_Map.thy
Brother12_Set.thy  RBT.thy          Tree_Set.thy
Cnp.thy         Set_by_Ordered.thy
document/       Sorted_Less.thy
lammich@lappnikow10: ~/lehre/FDS/SS18/public$ isabelle jedit ~/opt/Isabelle2017/s
rc/HOL/Data_Structures/

```

The presentation slide titled "Complexity" contains the following text:

Complexity

insert in time $O(1)$
and *merge* still $O(\log n)$

imperative!):

insert and *merge* in time $O(1)$
del_min still $O(\log n)$

Every operation in time $O(1)$?

The Isabelle2017 IDE window shows the following code in Binomial_Heap.thy:

```

19 proofs are straightforward and automatic.
20
21
22 subsection <Binomial Tree and Heap Datatype>
23
24 datatype 'a tree = Node (rank: nat) (root: 'a) (children: "'a tree list")
25
26 type_synonym 'a heap = "'a tree list"
27
28 subsection <Multiset of elements>
29
30 fun mset_tree :: "'a::linorder tree => 'a multiset" where
31   "mset_tree (Node _ a c) = {#a#} + (∑t<#mset c. mset_tree t)"
32
33 definition mset_heap :: "'a::linorder heap => 'a multiset" where..
34   "mset_heap c = (∑t<#mset c. mset_tree t)"
35
36 lemma mset_tree_simp_alt[simp]:
37   "mset_tree (Node r a c) = {#a#} + mset_heap c"
38   unfolding mset_heap_def by auto
39 declare mset_tree.simps[simp del]...

```

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOL/Data_Structures/)
26 type_synonym 'a heap = "'a tree list"
27
28 subsection <Multiset of elements>
29
30 fun mset_tree :: "'a::linorder tree => 'a multiset" where
31   "mset_tree (Node _ a c) = {#a#} + ( $\sum$  t $\in$ #mset c. mset_tree t)"
32 ..
33 definition mset_heap :: "'a::linorder heap => 'a multiset" where..
34   "mset_heap c = ( $\sum$  t $\in$ #mset c. mset_tree t)"
35 ..
36 lemma mset_tree_simp_alt[simp]:
37   "mset_tree (Node r a c) = {#a#} + mset_heap c"
38   unfolding mset_heap_def by auto
39   declare mset_tree.simps[simp del]...
40 ..
41 lemma mset_tree_nonempty[simp]: "mset_tree t  $\neq$  {#}"..
42 by (cases t) auto
43 ..
44 lemma mset_heap_Nil[simp]:
45   "mset_heap [] = {#}"
46 by (auto simp: mset_heap_def)

```

32.3 (739/22946) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 804/1143MB 9:51 AM
 debian 1 2 3 4 iamlich@lapnikow10: ~/le... slides-fds.pdf Isabelle2017 - Binomial_Heap... Isabelle2017 - Binomial_Heap... 09:51:49

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOL/Data_Structures/)
36 lemma mset_tree_simp_alt[simp]:
37   "mset_tree (Node r a c) = {#a#} + mset_heap c"
38   unfolding mset_heap_def by auto
39   declare mset_tree.simps[simp del]...
40 ..
41 lemma mset_tree_nonempty[simp]: "mset_tree t  $\neq$  {#}"..
42 by (cases t) auto
43 ..
44 lemma mset_heap_Nil[simp]:
45   "mset_heap [] = {#}"
46 by (auto simp: mset_heap_def)
47
48 lemma mset_heap_Cons[simp]: "mset_heap (t#ts) = mset_tree t + mset_heap ts"
49 by (auto simp: mset_heap_def)
50 ..
51 lemma mset_heap_empty_iff[simp]: "mset_heap ts = {#}  $\iff$  ts=[]"
52 by (auto simp: mset_heap_def)
53 ..
54 lemma root_in_mset[simp]: "root t  $\in$ # mset_tree t"
55 by (cases t) auto...
56 ..

```

53.1 (1368/22946) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 804/1143MB 9:52 AM
 debian 1 2 3 4 iamlich@lapnikow10: ~/le... slides-fds.pdf Isabelle2017 - Binomial_Heap... Isabelle2017 - Binomial_Heap... 09:52:38

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOL/Data_Structures/)
63 fun invar_btree :: "'a::linorder tree => bool" where
64   "invar_btree (Node r x ts)  $\iff$ 
65     ( $\forall$ t $\in$ set ts. invar_btree t)  $\wedge$  map rank ts = rev [0.. $r$ ]"
66
67 definition invar_bheap :: "'a::linorder heap => bool" where
68   "invar_bheap ts
69      $\iff$  ( $\forall$ t $\in$ set ts. invar_btree t)  $\wedge$  (sorted_wrt (op <) (map rank ts))"
70
71 text <Ordering (heap) invariant>
72 fun invar_otree :: "'a::linorder tree => bool" where
73   "invar_otree (Node _ x ts)  $\iff$  ( $\forall$ t $\in$ set ts. invar_otree t  $\wedge$  x <= root t)"
74
75 definition invar_ohheap :: "'a::linorder heap => bool" where
76   "invar_ohheap ts  $\iff$  ( $\forall$ t $\in$ set ts. invar_otree t)"
77 ..
78 definition invar :: "'a::linorder heap => bool" where
79   "invar ts  $\iff$  invar_bheap ts  $\wedge$  invar_ohheap ts"
80 ..
81 text <The children of a node are a valid heap>
82 lemma invar_ohheap_children:
83   "invar_otree (Node r v ts)  $\implies$  invar_ohheap (rev ts)".

```

75.1 (2057/22946) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 804/1143MB 9:53 AM
 debian 1 2 3 4 iamlich@lapnikow10: ~/le... slides-fds.pdf Isabelle2017 - Binomial_Heap... Isabelle2017 - Binomial_Heap... 09:53:58

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOL/Data_Structures/)
129 "ins_tree t [] = [t]"
130 "ins_tree t1 (t2#ts) =
131   (if rank t1 < rank t2 then t1#t2#ts else ins_tree (link t1 t2) ts)".
132 ...
133 lemma invar_bheap_Cons[simp]:
134   "invar_bheap (t#ts).
135      $\iff$  invar_btree t  $\wedge$  invar_bheap ts  $\wedge$  ( $\forall$ t' $\in$ set ts. rank t < rank t')"
136 by (auto simp: sorted_wrt_Cons invar_bheap_def)
137 ..
138 lemma invar_btree_ins_tree:
139   assumes "invar_btree t".
140   assumes "invar_bheap ts"
141   assumes " $\forall$ t' $\in$ set ts. rank t <= rank t'".
142   shows "invar_bheap (ins_tree t ts)".
143 using assms
144 by (induction t ts rule: ins_tree.induct) (auto simp: invar_btree_link_less_eq_Suc_le[symmetric])
145 ..
146 lemma invar_ohheap_Cons[simp]:
147   "invar_ohheap (t#ts)  $\iff$  invar_otree t  $\wedge$  invar_ohheap ts"....
148 by (auto simp: invar_ohheap_def)
149 ..

```

144.1 (4120/22946) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 804/1143MB 9:55 AM
 debian 1 2 3 4 iamlich@lapnikow10: ~/le... slides-fds.pdf Isabelle2017 - Binomial_Heap... Isabelle2017 - Binomial_Heap... 09:55:11

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOUData_Structures/)
267 shows "root t ≤ x"..
268 using assms
269 by (induction t arbitrary: x rule: mset_tree.induct) (fastforce simp: mset_heap_def)
270 ..
271 Lemma get_min_mset_aux:
272   assumes "ts ≠ []"....
273   assumes "invar_oheap ts"
274   assumes "x ∈# mset_heap ts"..
275   shows "get_min ts ≤ x"
276   using assms .
277 apply (induction ts arbitrary: x rule: get_min.induct)..
278 apply (auto.
279   simp: invar_otree_root_min min_def intro: order_trans;
280   meson linear order_trans invar_otree_root_min
281   )+
282 done..
283
284 Lemma get_min_mset:
285   assumes "ts ≠ []"....
286   assumes "invar ts"
287   assumes "x ∈# mset_heap ts"..

```

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOUData_Structures/)
551 hence "(2::nat)^t_merge ts₁ ts₂ ≤ 2^(2 * (length ts₁ + length ts₂) + 1)".
552   by (rule power_increasing) auto
553 also have "... = 2*(2^(length ts₁))^2*(2^(length ts₂))^2"....
554   by (auto simp: algebra_simps power_add power_mult)
555 also note BINVARS(1)[THEN size_mset_heap]
556 also note BINVARS(2)[THEN size_mset_heap]
557 finally have "2 ^ t_merge ts₁ ts₂ ≤ 2 * (n₁ + 1)² * (n₂ + 1)²".
558   by (auto simp: power2_nat_le_eq_le n₁_def n₂_def)
559 from le_log2_of_power[OF this] have "t_merge ts₁ ts₂ ≤ log 2 ..."....

```

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOUData_Structures/)
265 assumes "invar_otree t"
266 assumes "x ∈# mset_tree t".
267 shows "root t ≤ x"..
268 using assms
269 by (induction t arbitrary: x rule: mset_tree.induct) (fastforce simp: mset_heap_def)
270 ..
271 Lemma get_min_mset_aux:
272   assumes "ts ≠ []"....
273   assumes "invar_oheap ts"
274   assumes "x ∈# mset_heap ts"..
275   shows "get_min ts ≤ x"
276   using assms .
277 apply (induction ts arbitrary: x rule: get_min.induct)..
278 apply (auto.
279   simp: invar_otree_root_min min_def intro: order_trans;
280   meson linear order_trans invar_otree_root_min
281   )+
282 done..
283
284 Lemma get_min_mset:
285   assumes "ts ≠ []"....

```

```

Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOUData_Structures/)
520 "t_merge [] ts₂ = 1"..
521 "t_merge (t₁#ts₁) (t₂#ts₂) = 1 + (
522   if rank t₁ < rank t₂ then t_merge ts₁ (t₂#ts₂)
523   else if rank t₂ < rank t₁ then t_merge (t₁#ts₁) ts₂
524   else t_ins_tree (link t₁ t₂) (merge ts₁ ts₂) + t_merge ts₁ ts₂
525 )"..
526 ..
527 text <A crucial idea is to estimate the time in correlation with the
528 result length, as each carry reduces the length of the result.>..
529
530 Lemma t_ins_tree_length:
531   "t_ins_tree t ts + length (ins_tree t ts) = 2 + length ts"
532 by (induction t ts rule: ins_tree.induct) auto
533
534 Lemma t_merge_length:
535   "length (merge ts₁ ts₂) + t_merge ts₁ ts₂ ≤ 2 * (length ts₁ + length ts₂) + 1"
536 by (induction ts₁ ts₂ rule: t_merge.induct)..
537   (auto simp: t_ins_tree_length algebra_simps)
538
539 text <Finally, we get the desired logarithmic bound>
540 Lemma t_merge_bound_aux:

```

```
Isabelle2017 - Binomial_Heap.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Binomial_Heap.thy ($ISABELLE_HOME/src/HOUData_Structures/)
697 using t_merge_bound_aux[OF <invar_bheap (rev ts₁)> <invar_bheap ts₂>]
698 by (auto simp: n₁_def n₂_def algebra_simps)
699 also have "n₁ + n₂ ≤ n"
700 unfolding n₁_def n₂_def n_def
701 using mset_get_min_rest[OF GM <ts≠[]>]
702 by (auto simp: mset_heap_def)
703 finally have "t_del_min ts ≤ 6 * log 2 (n+1) + 3".
704 by auto
705 thus ?thesis by (simp add: algebra_simps)
706 qed .....
707 ..
708 Lemma t_del_min_bound:
709 fixes ts
710 defines "n ≡ size (mset_heap ts)"
711 assumes "invar ts"
712 assumes "ts≠[]"
713 shows "t_del_min ts ≤ 6 * log 2 (n+1) + 3".
714 using assms t_del_min_bound_aux unfolding invar_def by blast
715
716 end
717

Output Query Sledgehammer Symbols
716.1 (22942/22946) Matches line 7: theory Binomial_Heap (isabelle.isabelle.UTF-8-isabelle)Nmr o UG 36/1232MB 10:01 AM
debian 1 2 3 4 iamich@lapnikow10: ~/le... slides-fds.pdf Isabelle2017 - Binomial_Heap... Isabelle2017 - Binomial_Heap... 10:01:54
```