

## Script generated by TTT

Title: Lammich: FDS Tutorial (14.07.2017)

Date: Fri Jul 14 09:37:05 CEST 2017

Duration: 25:51 min

Pages: 23

## 16 Amortized Complexity

Motivation

Formalization

Simple Classical Examples

189

The screenshot shows a terminal window with the following commands and output:

```
lammich@lapnikow10: ~/lehre/FDS/Private/Slides
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/
2015_SS_Automata2/ logics/ tdp/
2016_SS_Seminar_SAT/ programmoptimierung/ theo/
FDS/ semantics/ tries_thests/
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/FDS/
Private/ Public/ web/
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/FDS/Private/Slides/
lammich@lapnikow10: ~/Lehre/FDS/Private/Slides$ ls
Anor_Slides.thy Leftist_Heap_Slides.thy Tree23_Slides.thy
Binomial_Tree_Slides.thy RBT_Slides.thy Tree_Notation.thy
Binomial_Tree_Slides.thy- Search_Trees.thy Tree_Slides.thy
lammich@lapnikow10: ~/opt/ttt
lammich@lapnikow10: ~/opt/ttt$
```

A jEdit editor window is open, displaying the file `Thys/Amortized_Examples.thy`. The editor shows a large blank white area, indicating the content of the file is not fully visible.

The screenshot shows a terminal window with the following commands and output:

```
lammich@lapnikow10: ~/lehre/FDS/Private/Slides
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/
2015_SS_Automata2/ logics/ tdp/
2016_SS_Seminar_SAT/ programmoptimierung/ theo/
FDS/ semantics/ tries_thests/
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/FDS/
Private/ Public/ web/
lammich@lapnikow10: ~/opt/ttt$ cd ~/Lehre/FDS/Private/Slides/
lammich@lapnikow10: ~/Lehre/FDS/Private/Slides$ ls
Anor_Slides.thy Leftist_Heap_Slides.thy Tree23_Slides.thy
Binomial_Tree_Slides.thy RBT_Slides.thy Tree_Notation.thy
Binomial_Tree_Slides.thy- Search_Trees.thy Tree_Slides.thy
lammich@lapnikow10: ~/Lehre/FDS/Private/Slides
lammich@lapnikow10: ~/Lehre/FDS/Private/Slides$ cd ~/Lehre/FDS/Public/Thys/
lammich@lapnikow10: ~/Lehre/FDS/Public/Thys$ ls
Amortized_Examples.thy Priority_Queue.thy Sort
Anortized_Examples.thy- RBT_Set.thy Tree
BinHeap.thy RBT.thy Trie
Heap_Prelim.thy Skew_Heap_Analysis.thy Trie
Leftist_Heap.thy Skew_Heap.thy
lammich@lapnikow10: ~/Lehre/FDS/Public/Thys$ isabelle
hy
lammich@lapnikow10: ~/Lehre/FDS/Public/Thys$
```

A jEdit editor window is open, displaying the file `Thys/Amortized_Examples.thy`. A large orange logo with the letters 'jE' is overlaid on the editor window. The editor shows a large blank white area, indicating the content of the file is not fully visible.

```
Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
Amortized_Examples.thy (~lehre/FDS/Public/Thys/)
(* Application: Doing n increments *)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"
fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"
| "t_n_increments (Suc n) bs = t_incr bs + t_n_increments n (incr bs)"
(* Generalized lemma with arbitrary start state.
Telescoping sum is done implicitly in this proof. *)
lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi$  bs -  $\Phi$  (n_increments n bs)"
proof (induction n bs rule: n_increments.induct)
  case (1 bs)
  then show ?case by simp
next
  case (2 n bs)
  then show ?case by simp
end
Output Query Sledgehammer Symbols
55.1 (1.288/8219) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 11/177MB 9:41 AM
debian 1 2 3 4 lammich@lapnikow10: ~/le... document.pdf lammich@lapnikow10: ~/le... Isabelle2016-1 - Amortized_E... 09:41:21
```

```
Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
Amortized_Examples.thy (~lehre/FDS/Public/Thys/)
(* Application: Doing n increments *)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"
fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"
| "t_n_increments (Suc n) bs = t_incr bs + t_n_increments n (incr bs)"
(* Generalized lemma with arbitrary start state.
Telescoping sum is done implicitly in this proof. *)
lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi$  bs -  $\Phi$  (n_increments n bs)"
proof (induction n bs rule: n_increments.induct)
  case (1 bs)
  then show ?case by simp
next
  case (2 n bs)
  then show ?case by simp
end
Output Query Sledgehammer Symbols
54.28 (1.250/8219) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 11/177MB 9:42 AM
debian 1 2 3 4 lammich@lapnikow10: ~/le... document.pdf lammich@lapnikow10: ~/le... Isabelle2016-1 - Amortized_E... 09:42:19
```

```
Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
Amortized_Examples.thy (~lehre/FDS/Public/Thys/)
(* Application: Doing n increments *)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"
fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"
| "t_n_increments (Suc n) bs = t_incr bs + t_n_increments n (incr bs)"
(* Generalized lemma with arbitrary start state.
Telescoping sum is done implicitly in this proof. *)
lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi$  bs -  $\Phi$  (n_increments n bs)"
proof (induction n bs rule: n_increments.induct)
  case (1 bs)
  then show ?case by simp
next
  case (2 n bs)
  then show ?case by simp
end
Output Query Sledgehammer Symbols
54.44 (1.266/8219) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 11/177MB 9:43 AM
debian 1 2 3 4 lammich@lapnikow10: ~/le... document.pdf lammich@lapnikow10: ~/le... Isabelle2016-1 - Amortized_E... 09:43:04
```

```
Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
Amortized_Examples.thy (~lehre/FDS/Public/Thys/)
(* Application: Doing n increments *)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"
fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"
| "t_n_increments (Suc n) bs = t_incr bs + t_n_increments n (incr bs)"
(* Generalized lemma with arbitrary start state.
Telescoping sum is done implicitly in this proof. *)
lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi$  bs -  $\Phi$  (n_increments n bs)"
proof (induction n bs rule: n_increments.induct)
  case (1 bs)
  then show ?case by simp
next
  case (2 n bs)
  then show ?case by simp
end
Output Query Sledgehammer Symbols
54.39 (1.261/8219) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 11/177MB 9:43 AM
debian 1 2 3 4 lammich@lapnikow10: ~/le... document.pdf lammich@lapnikow10: ~/le... Isabelle2016-1 - Amortized_E... 09:43:54
```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"

fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"

proof (state)
this:

goal (2 subgoals):
1.  $\wedge bs. \text{int} (t\_n\_increments\ 0\ bs) = 1 + 2 * \text{int}\ 0 + \Phi\ bs - \Phi (n\_increments\ 0\ bs)$ 
2.  $\wedge n\ bs. \text{int} (t\_n\_increments\ n\ (incr\ bs)) = 1 + 2 * \text{int}\ n + \Phi (incr\ bs) - \Phi (n\_increments\ n\ (incr\ bs)) \implies \text{int} (t\_n\_increments\ (Suc\ n)\ bs) = 1 + 2 * \text{int}\ (Suc\ n) + \Phi\ bs - \Phi (n\_increments\ (Suc\ n)\ bs)$ 

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
fun n_increments where
  "n_increments 0 bs = bs"
| "n_increments (Suc n) bs = n_increments n (incr bs)"

fun t_n_increments :: "nat => counter => nat" where
  "t_n_increments 0 bs = 1"
| "t_n_increments (Suc n) bs = t_incr bs + t_n_increments n (incr bs)"

(* Generalized lemma with arbitrary start state.
Telescoping sum is done implicitly in this proof. *)
lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi\ bs - \Phi (n\_increments\ n\ bs)$ "
proof (induction n bs rule: n_increments.induct)

proof (state)
this:

goal (2 subgoals):

```

```

Isabelle2016-1 - Amortized_Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
"t_n_increments n bs = 1 + 2*n +  $\Phi\ bs - \Phi (n\_increments\ n\ bs)$ "
proof (induction n bs rule: n_increments.induct)
case (1 bs)
then show ?case by simp
next
case (2 n bs)
then show ?case apply simp

using a_incr[of bs] by auto
qed

proof (prove)
using this:
  int (t_n_increments n (incr bs)) = 1 + 2 * int n +  $\Phi (incr\ bs) - \Phi (n\_increments\ n\ (incr\ bs))$ 

goal (1 subgoal):
1. int (t_n_increments (Suc n) bs) = 1 + 2 * int (Suc n) +  $\Phi\ bs - \Phi (n\_increments\ (Suc\ n)\ bs)$ 

```

```

Isabelle2016-1 - Amortized_Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
proof (induction n bs rule: n_increments.induct)
case (1 bs)
then show ?case by simp
next
case (2 n bs)
then show ?case apply simp

using a_incr[of bs] by auto
qed

(* Estimation for initial state *)

proof (prove)
goal (1 subgoal):
1. int (t_n_increments n (incr bs)) = 1 + 2 * int n +  $\Phi (incr\ bs) - \Phi (n\_increments\ n\ (incr\ bs)) \implies \text{int} (t\_incr\ bs) + \Phi (incr\ bs) = 2 + \Phi\ bs$ 

```

```

Isabelle2016-1 - Amortized_Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
case (1 bs)
  then show ?case by simp
next
case (2 n bs)
  then show ?case apply simp

using a_incr[of bs] by auto
qed

(* Estimation for initial state *)

proof (prove)
goal (1 subgoal):
1. int (t_n_increments n (incr bs)) = 1 + 2 * int n +  $\Phi$  (incr bs) -  $\Phi$  (n_increments n (incr bs))  $\implies$ 
  int (t_incr bs) +  $\Phi$  (incr bs) = 2 +  $\Phi$  bs

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
Lemma t_n_increments_aux:
  "t_n_increments n bs = 1 + 2*n +  $\Phi$  bs -  $\Phi$  (n_increments n bs)"
proof (induction n bs rule: n_increments.induct)
case (1 bs)
  then show ?case by simp
next
case (2 n bs)
  then show ?case
  using a_incr[of bs] by auto
qed

proof (state)
goal (1 subgoal):
1.  $\wedge$  n bs. int (t_n_increments n (incr bs)) =
  1 + 2 * int n +  $\Phi$  (incr bs) -  $\Phi$  (n_increments n (incr bs))  $\implies$ 
  int (t_n_increments (Suc n) bs) = 1 + 2 * int (Suc n) +  $\Phi$  bs -  $\Phi$  (n_increments (Suc n) bs)

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
then show ?case
  using a_incr[of bs] by auto
qed

(* Estimation for initial state *)
Lemma t_n_increments: "t_n_increments n init  $\leq$  1+2*n"
using
  t_n_increments_aux[of n init]
   $\Phi$ _non_neg[of "n_increments n init"] (* Exploit that potential cannot be negative *)
   $\Phi$ _init
  by auto

proof (prove)
using this:
  int (t_n_increments n init) = 1 + 2 * int n +  $\Phi$  init -  $\Phi$  (n_increments n init)
  0  $\leq$   $\Phi$  (n_increments n init)
   $\Phi$  init = 0
goal (1 subgoal):

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
"t_incr [] = 1" |
"t_incr (False # bs) = 1" |
"t_incr (True # bs) = t_incr bs + 1"

definition  $\Phi$  :: "counter  $\Rightarrow$  int" where
" $\Phi$  bs = length(filter id bs)"
Lemma  $\Phi$ _non_neg: " $\Phi$  bs  $\geq$  0"
by (simp add:  $\Phi$ _def)
Lemma  $\Phi$ _init: " $\Phi$  init = 0"

consts
 $\Phi$  :: "bool list  $\Rightarrow$  int"

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
Lemma  $\Phi$ _init: " $\Phi$  init = 0"
by(simp add:  $\Phi$ _def init_def)

Lemma a_incr: "t_incr bs +  $\Phi$ (incr bs) -  $\Phi$  bs = 2"
apply(induction bs rule: incr.induct)
apply(simp_all add:  $\Phi$ _def)
done

(* Application: Doing n increments *)

proof(prove)
goal (1 subgoal):
1. int (t_incr bs) +  $\Phi$  (incr bs) -  $\Phi$  bs = 2

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
Lemma  $\Phi$ _init: " $\Phi$  init = 0"
by(simp add:  $\Phi$ _def init_def)

Lemma a_incr: "t_incr bs +  $\Phi$ (incr bs) -  $\Phi$  bs = 2"
apply(induction bs rule: incr.induct)
apply(simp_all add:  $\Phi$ _def)
done

(* Application: Doing n increments *)

theorem a_incr: int (t_incr 7bs) +  $\Phi$  (incr 7bs) -  $\Phi$  7bs = 2

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
then show ?case
using a_incr[of bs] by auto
qed

(* Estimation for initial state *)
Lemma t_n_increments: "t_n_increments n init  $\leq$  1+2*n"
using
t_n_increments_aux[of n init]
 $\Phi$ _non_neg[of "n_increments n init"] (* Exploit that potential cannot be negative *)
 $\Phi$ _init
by auto

proof(prove)
goal (1 subgoal):
1. t_n_increments n init  $\leq$  1 + 2 * n

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
by auto

end

subsection "Dynamic tables: insert only"

locale Dyn_Tab
begin

text: Because we are not interested in the elements of the dynamic table but merely its size we abstract it to a pair  $\langle n, l \rangle$  where  $\langle n \rangle$  is the number of elements in the table and  $\langle l \rangle$  the size of the table.

type_synonym tab = "nat  $\times$  nat"

fun invar :: "tab  $\Rightarrow$  bool" where
"invar (n, l) = (n  $\leq$  l)"

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
we abstract it to a pair  $\langle n, l \rangle$  where  $\langle n \rangle$  is the number of elements in the table and  $\langle l \rangle$ 
the size of the table.

type_synonym tab = "nat  $\times$  nat"

fun invar :: "tab  $\Rightarrow$  bool" where
  "invar (n,l) = (n  $\leq$  l)"

definition init :: tab where
  "init = (0,0)"

text<Insertion: the element does not matter>
fun ins :: "tab  $\Rightarrow$  tab" where
  "ins (n,l) = (n+1, if n=l then l else if l=0 then 1 else 2*l)"

text<Time: if the table overflows, we count only the time for copying elements>
fun t_ins :: "nat  $\times$  nat  $\Rightarrow$  nat" where
  "t_ins (n,l) = (if n<l then 1 else n+1)"

lemma "invar init"

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
fun invar :: "tab  $\Rightarrow$  bool" where
  "invar (n,l) = (n  $\leq$  l)"

definition init :: tab where
  "init = (0,0)"

text<Insertion: the element does not matter>
fun ins :: "tab  $\Rightarrow$  tab" where
  "ins (n,l) = (n+1, if n<l then l else if l=0 then 1 else 2*l)"

text<Time: if the table overflows, we count only the time for copying elements>
fun t_ins :: "nat  $\times$  nat  $\Rightarrow$  nat" where
  "t_ins (n,l) = (if n<l then 1 else n+1)"

lemma "invar init"
by(simp add: init_def)

lemma "invar t  $\Rightarrow$  invar (ins t)"
apply(cases t)

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
lemma "invar init"
by(simp add: init_def)

lemma "invar t  $\Rightarrow$  invar (ins t)"
apply(cases t)
apply(auto)
done

fun  $\Phi$  :: "tab  $\Rightarrow$  nat" where
  " $\Phi$  (n,l) = 2*n - l"

text <Careful:  $\langle n \rangle$  and  $\langle l \rangle$  are natural numbers.
Thus  $\langle 2*n \rangle$  can be less than  $\langle l \rangle$  (in which case  $\langle 2*n - l \rangle = 0$ !)
because the invariant does not exclude this case, although it cannot arise.
If you go through the proof of lemma <a_ins> in detail
you will understand why this case also works out fine.>

lemma "invar t  $\Rightarrow$   $\Phi$  t  $\geq$  0"
apply(cases t)
apply(auto)

```

```

Isabelle2016-1 - Amortized_Examples.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Amortized_Examples.thy (~/lehre/FDS/Public/Thys/)
because the invariant does not exclude this case, although it cannot arise.
If you go through the proof of lemma <a_ins> in detail
you will understand why this case also works out fine.>

lemma "invar t  $\Rightarrow$   $\Phi$  t  $\geq$  0"
apply(cases t)
apply(auto)
done

lemma " $\Phi$  init = 0"
by(simp add: init_def)

lemma a_ins: "invar t  $\Rightarrow$  t_ins t +  $\Phi$  (ins t) -  $\Phi$  t  $\leq$  3"
apply(cases t)
apply(auto split: if_splits)
done

end

```