

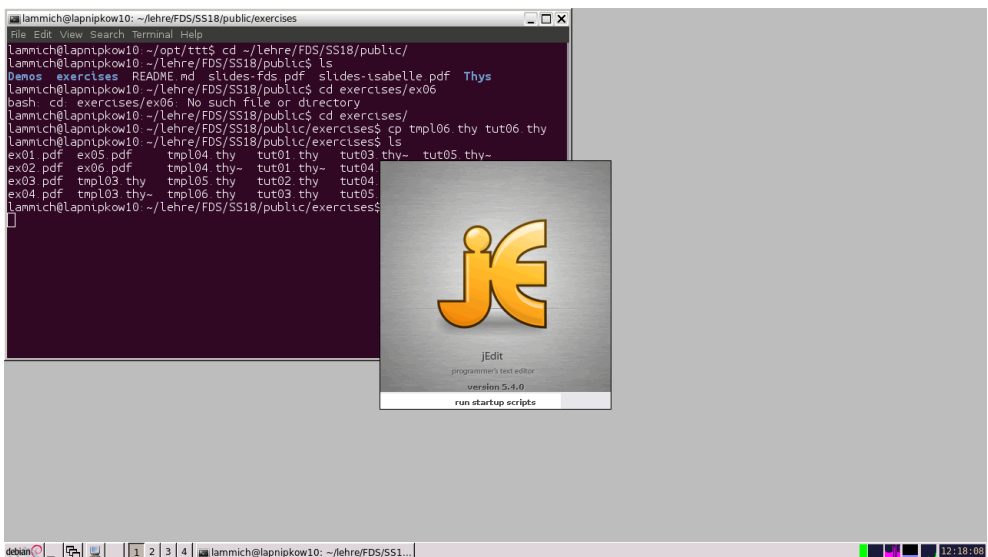
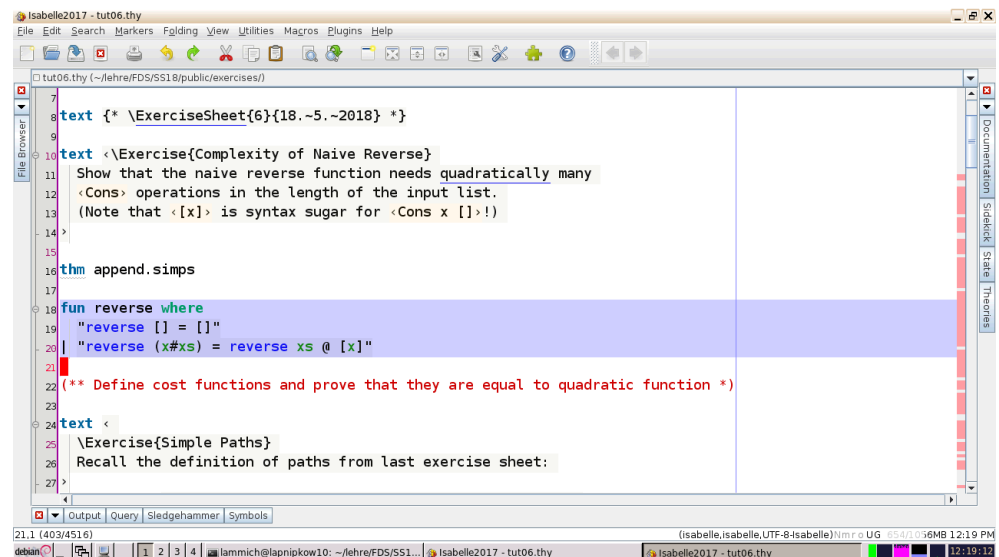
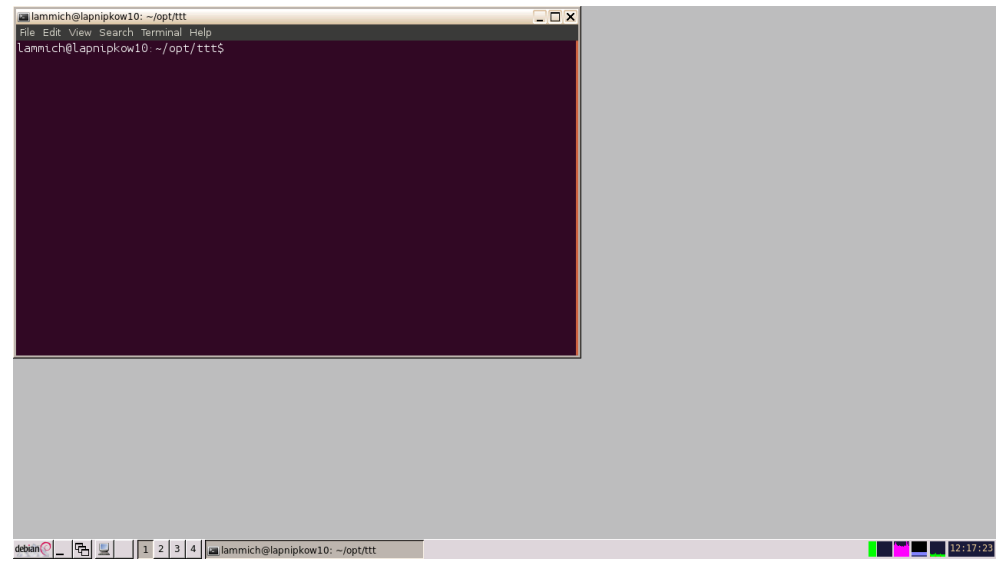
## Script generated by TTT

Title: Lammich: FDS Tutorial (18.05.2018)

Date: Fri May 18 12:17:23 CEST 2018

Duration: 76:40 min

Pages: 87



```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
7 text {* \ExerciseSheet{6}{18.-5.-2018} *}
8
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <x> is syntax sugar for <Cons x []>!)
14 >
15
16 thm append.simps
17
18 fun reverse where
19   "reverse [] = []"
20 | "reverse (x#xs) = reverse xs @ [x]"
21
22 (** Define cost functions and prove that they are equal to quadratic function *)
23
24 text <
25   \Exercise{Simple Paths}
26   Recall the definition of paths from last exercise sheet:
27 >
Output Query Sledgehammer Symbols
20.38 (402/4516) (isabelle.isabelle,UTF-8-isabelle)NmroUG 660/1056MB 12:19 PM

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
7 text {* \ExerciseSheet{6}{18.-5.-2018} *}
8
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <x> is syntax sugar for <Cons x []>!)
14 >
15
16 thm append.simps
17
18 fun reverse where
19   "reverse [] = []"
20 | "reverse (x#xs) = reverse xs @ [x]"
21
22 (** Define cost functions and prove that they are equal to quadratic function *)
23
24 text <
25   \Exercise{Simple Paths}
26   Recall the definition of paths from last exercise sheet:
27 >
Output Query Sledgehammer Symbols
20.1 (365/4516) (isabelle.isabelle,UTF-8-isabelle)NmroUG 669/1056MB 12:20 PM

```

```

Isabelle2017 - List.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
List.thy ($ISABELLE_HOME/src/HOL/)
61 "butlast (x # xs) = (if xs = [] then [] else x # butlast xs)"
62
63 lemma set_rec: "set xs = rec_list {} (\x _. insert x) xs"
64 by (induct xs) auto
65
66 definition coset :: "'a list => 'a set" where
67 [simp]: "coset xs = - set xs"
68
69 primrec append :: "'a list => 'a list => 'a list" (infixr "@" 65) where
70 append_Nil: "[] @ ys = ys" |
71 append_Cons: "(x#xs) @ ys = x # xs @ ys"
72
73 primrec rev :: "'a list => 'a list" where
74 "rev [] = []" |
75 "rev (x # xs) = rev xs @ [x]"
76
Proof state Auto update Update Search: 100%
[ ] @ ?ys = ?ys
(?x # ?xs) @ ?ys = ?x # ?xs @ ?ys
Output Query Sledgehammer Symbols
69.1 (1497/236142) Input/output complete (isabelle.isabelle,UTF-8-isabelle)NmroUG 67/1174MB 12:20 PM

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
7 text {* \ExerciseSheet{6}{18.-5.-2018} *}
8
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <x> is syntax sugar for <Cons x []>!)
14 >
15
16 thm append.simps
17
18 fun reverse where
19   "reverse [] = []"
20 | "reverse (x#xs) = reverse xs @ [x]"
21
22 (** Define cost functions and prove that they are equal to quadratic function *)
23
Proof state Auto update Update Search: 100%
[ ] @ ?ys = ?ys
(?x # ?xs) @ ?ys = ?x # ?xs @ ?ys
Output Query Sledgehammer Symbols
16.5 (313/4516) (isabelle.isabelle,UTF-8-isabelle)NmroUG 67/1179MB 12:21 PM

```

Isabelle2017 - tut06.thy

```

7 text {* \ExerciseSheet{6}{18.-5.-2018} *}
8
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <[x]> is syntax sugar for <Cons x []>!)
14 >
15
16 thm append.simps
17
18 fun reverse where
19   "reverse [] = []"
20 | "reverse (x#xs) = reverse xs @ [x]"
21
22 (** Define cost functions and prove that they are equal to quadratic function *)

```

Proof state Auto update Update Search: 100%

```

[] @ ?ys = ?ys
(?x # ?xs) @ ?ys = ?x # ?xs @ ?ys

```

Output Query Sledgehammer Symbols

17.1 (326/4516) (isabelle.isabelle.UTF-8-isabelle)Nm r o UG 350/1179MB 12:21 PM

Isabelle2017 - tut06.thy (modified)

```

8 text {* \ExerciseSheet{6}{18.-5.-2018} *}
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <[x]> is syntax sugar for <Cons x []>!)
14 >
15
16 fun append :: "'a list => 'a list => 'a list" where
17   append_Nil: "append [] ys = ys" |
18   append_Cons: "append (x#xs) ys = x # append xs ys"
19
20 thm append.simps
21
22 fun reverse where
23   "reverse [] = []"

```

Proof state Auto update Update Search: 100%

```

consts
append :: "'a list => 'a list => 'a list"
Found termination order: "(λp. length (fst p)) < *mlex* {}"

```

Output Query Sledgehammer Symbols

16.50 (358/4655) (isabelle.isabelle.UTF-8-isabelle)Nm r o UG 520/1335MB 12:22 PM

Isabelle2017 - tut06.thy (modified)

```

8 text {* \ExerciseSheet{6}{18.-5.-2018} *}
9
10 text <\Exercise{Complexity of Naive Reverse}
11 Show that the naive reverse function needs quadratically many
12 <Cons> operations in the length of the input list.
13 (Note that <[x]> is syntax sugar for <Cons x []>!)
14 >
15
16 fun t_append :: "'a list => 'a list => 'a list" where
17   append_Nil: "t_append [] ys = 0" |
18   append_Cons: "t_append (x#xs) ys = x # t_append xs ys"
19
20 thm append.simps
21
22 fun reverse where

```

Proof state Auto update Update Search: 100%

```

Type unification failed: No type arity list :: zero
Type error in application: incompatible operand type

```

Output Query Sledgehammer Symbols

16.31 (339/4664) (isabelle.isabelle.UTF-8-isabelle)Nm r o UG 350/135MB 12:23 PM

Isabelle2017 - tut06.thy (modified)

```

19 thm append.simps
20
21 fun reverse where
22   "reverse [] = []"
23 | "reverse (x#xs) = reverse xs @ [x]"
24
25
26 fun t_reverse where
27   "t_reverse [] = 0"
28 | "t_reverse (x#xs) = t_reverse xs + 1 @ [x]"
29
30 (** Define cost functions and prove that they are equal to quadratic function *)
31
32

```

Proof state Auto update Update Search: 100%

```

Inner syntax error
Failed to parse prop

```

Output Query Sledgehammer Symbols

28.41 (603/4723) (isabelle.isabelle.UTF-8-isabelle)Nm r o UG 760/135MB 12:26 PM

Isabelle2017 - tut06.thy (modified)

```

19 thm append.simps
20
21 fun reverse where
22   "reverse [] = []"
23 | "reverse (x#xs) = reverse xs @ [x]"
24
25 fun t_reverse where
26   "t_reverse [] = 0"
27 | "t_reverse (x#xs) = t_reverse xs + 1 + t_append (t_) @ [x]"
28
29
30 (** Define cost functions and prove that they are equal to quadratic function *)
31
32

```

Inner syntax error  
Failed to parse prop

28.54 (616/4742) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 1071/298MB 12:26 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:26:34

Isabelle2017 - tut06.thy

```

21 fun reverse where
22   "reverse [] = []"
23 | "reverse (x#xs) = reverse xs @ [x]"
24
25 fun t_reverse where
26   "t_reverse [] = 0"
27 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
28
29
30 (** Define cost functions and prove that they are equal to quadratic function *)
31
32
33 text <
34   \Exercise{Simple Paths}

```

```

consts
reverse :: "'a list ⇒ 'a list"
Found termination order: "length < *mlex* > {}"

```

28.51 (613/4745) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 363/1298MB 12:27 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:27:20

Isabelle2017 - tut06.thy

```

21 fun reverse where
22   "reverse [] = []"
23 | "reverse (x#xs) = reverse xs @ [x]"
24
25 fun t_reverse where
26   "t_reverse [] = 0"
27 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
28
29
30 (** Define cost functions and prove that they are equal to quadratic function *)
31
32
33 text <
34   \Exercise{Simple Paths}

```

```

consts
t_reverse :: "'a list ⇒ nat"
Found termination order: "length < *mlex* > {}"

```

28.46 (608/4745) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 457/1298MB 12:27 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:27:40

Isabelle2017 - tut06.thy

```

13 (Note that <[x]> is syntax sugar for <Cons x []>!)
14
15 fun t_append :: "'a list ⇒ 'a list ⇒ nat" where
16   "t_append [] ys = 0" |
17   "t_append (x#xs) ys = 1 + t_append xs ys"
18
19 thm append.simps
20
21 fun reverse where
22   "reverse [] = []"
23 | "reverse (x#xs) = reverse xs @ [x]"
24
25 fun t_reverse where

```

```

• [] @ ?ys = ?ys
• (?x # ?xs) @ ?ys = ?x # ?xs @ ?ys

```

21.1 (444/4745) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 467/1298MB 12:28 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:28:14

Isabelle2017 - tut06.thy (modified)

```

13 (Note that <x> is syntax sugar for <Cons x []>!)
14
15
16 fun t_append :: "'a list ⇒ 'a list ⇒ nat" where
17   "t_append [] ys = 0" |
18   "t_append (x#xs) ys = 1 + t_append xs ys"
19
20 Lemma "t_append xs ys = 1 .."
21
22
23 thm append.simps
24
25 fun reverse where
26   "reverse [] = []"

```

Inner syntax error: unexpected end of input  
Failed to parse prop

20.25 (451/4782) Input/output complete (Isabelle, Isabelle, UTF-8-Isabelle) Nm r o UG 736/1.98MB 12:28 PM

Isabelle2017 - tut06.thy (modified)

```

16 fun t_append :: "'a list ⇒ 'a list ⇒ nat" where
17   "t_append [] ys = 0" |
18   "t_append (x#xs) ys = 1 + t_append xs ys"
19
20 Lemma "t_append xs ys = length xs + length ys"
21
22
23 thm append.simps

```

proof (prove)  
goal (1 subgoal):  
1. t\_append xs ys = length xs + length ys  
Auto Quickcheck found a counterexample:  
xs = []  
ys = [a,]  
Evaluated terms:  
t\_append xs ys = 0  
length xs + length ys = 1

20.45 (471/4802) (Isabelle, Isabelle, UTF-8-Isabelle) Nm r o UG 371.335MB 12:29 PM

Isabelle2017 - tut06.thy

```

27 | "reverse (x#xs) = reverse xs @ [x]"
28
29 Lemma [simp]: "length (reverse xs) = length xs" by (induction xs) auto
30
31
32 fun t_reverse where
33   "t_reverse [] = 0"
34 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36
37
38
39
40 (** Define cost functions and prove that they are equal to quadratic function *)

```

consts  
t\_reverse :: "'a list ⇒ nat"  
Found termination order: "length < \*mlex\* > {}"

36.1 (773/4889) (Isabelle, Isabelle, UTF-8-Isabelle) Nm r o UG 802.334MB 12:31 PM

Isabelle2017 - tut06.thy (modified)

```

32 fun t_reverse where
33   "t_reverse [] = 0"
34 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36
37 (*
38   t 0 = 0
39   t (n+1) = t n + n + 1
40 *)

```

consts  
t\_reverse :: "'a list ⇒ nat"  
Found termination order: "length < \*mlex\* > {}"

37.1 (776/4829) (Isabelle, Isabelle, UTF-8-Isabelle) Nm r o UG 808.34MB 12:32 PM

Isabelle2017 - tut06.thy (modified)

```

31 fun t_reverse where
32   "t_reverse [] = 0"
33 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
34
35 (*
36  t 0 = 0
37  t (n+1) = t n + n + 1
38 *)
39
40 (** Define cost functions and prove that they are equal to quadratic function **)
41
42
43
44

```

consts  
t\_reverse :: "a list ⇒ nat"  
Found termination order: "length < \*mlex\* > {}"

Output Query Sledgehammer Symbols

38.1 (786/4929) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 72/1334MB 12:33 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:33:09

Isabelle2017 - tut06.thy

```

34 "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36 (*
37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)
40
41
42 value "map (λn. t_reverse (replicate n (0::int))) [0,1,2,3,4,5,6]"
43
44 (** Define cost functions and prove that they are equal to quadratic function **)
45
46
47 text <

```

"[0, 1, 3, 6, 10, 15, 21]"  
:: "nat list"

Output Query Sledgehammer Symbols

42.58 (872/4996) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 78/71 334MB 12:35 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:35:09

Isabelle2017 - tut06.thy

```

34 "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36 (*
37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)
40
41 value "map (λn. t_reverse (replicate n (0::int))) [0,1,2,3,4,5,6]"
42
43 (** Define cost functions and prove that they are equal to quadratic function **)
44
45
46
47 text <

```

"[0, 1, 3, 6, 10, 15, 21]"  
:: "nat list"

Output Query Sledgehammer Symbols

43.1 (882/4996) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 79/22 34MB 12:35 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:35:27

Isabelle2017 - tut06.thy

```

37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)
40
41
42 value "map (λn. t_reverse (replicate n (0::int))) [0,1,2,3,4,5,6]"
43
44 lemma "t_reverse xs = length xs * (length xs + 1) div 2"
45   apply (induction xs) apply auto done
46
47 (** Define cost functions and prove that they are equal to quadratic function **)
48
49
50

```

proof (prove)  
goal (1 subgoal):  
1.  $\forall a \text{ xs. } t\_reverse \text{ xs} = (\text{length } \text{xs} + \text{length } \text{xs} * \text{length } \text{xs}) \text{ div } 2 \implies$   
 $(\text{length } \text{xs} + \text{length } \text{xs} * \text{length } \text{xs}) \text{ div } 2 + t\_append (\text{reverse } \text{xs}) [a] =$   
 $(\text{length } \text{xs} + (\text{length } \text{xs} + (\text{length } \text{xs} + \text{length } \text{xs} * \text{length } \text{xs}))) \text{ div } 2$

Output Query Sledgehammer Symbols

45.30 (984/5107) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 80/21 336MB 12:37 PM  
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:37:08

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
33 | "t_reverse [] = 0"
34 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36
37 (*
38  t 0 = 0
39  t (n+1) = t n + n + 1
40 *)
41
42 value "map (\n. t_reverse (replicate n (0::int))) [0,1,2,3,4,5,6]"
43
44 Lemma "t_reverse xs = length xs * (length xs + 1) div 2"
45 apply (induction xs) apply (auto simp: t_append_conv reverse_length) done

proof (prove)
goal (1 subgoal):
1. t_reverse xs = length xs * (length xs + 1) div 2

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
27 | "reverse (x#xs) = reverse xs @ [x]"
28
29 Lemma reverse_length: "length (reverse xs) = length xs" by (induction xs) auto
30
31
32 fun t_reverse where
33   "t_reverse [] = 0"
34   "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36 (*
37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)

```

consts  
t\_reverse :: "'a list ⇒ nat"  
Found termination order: "length <math>mlex</math>> {}"

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
27 | "reverse (x#xs) = reverse xs @ [x]"
28
29 Lemma reverse_length: "length (reverse xs) = length xs" by (induction xs) auto
30
31
32 fun t_reverse where
33   "t_reverse [] = 0"
34   "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36 (*
37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)

```

theorem reverse\_length: length (reverse ?xs) = length ?xs

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
27 | "reverse (x#xs) = reverse xs @ [x]"
28
29 Lemma reverse_length: "length (reverse xs) = length xs" by (induction xs) auto
30
31
32 fun t_reverse where
33   "t_reverse [] = 0"
34   "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
35
36 (*
37  t 0 = 0
38  t (n+1) = t n + n + 1
39 *)

```

consts  
t\_reverse :: "'a list ⇒ nat"  
Found termination order: "length <math>mlex</math>> {}"

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
31 fun t_reverse where
32   "t_reverse [] = 0"
33 | "t_reverse (x#xs) = t_reverse xs + t_append (reverse xs) [x] + 1"
34
35 (*
36   t 0 = 0
37   t (n+1) = t n + n + 1
38 *)
39
40 value "map (λn. t_reverse (replicate n (0::int))) [0,1,2,3,4,5,6]"
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
40.1 (826/5144) (Isabelle, Isabelle.UTF-8-Isabelle)Nmr o UG 05/1337MB 12:41 PM
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:41:59

```

```

consts
t_reverse :: "'a list ⇒ nat"
Found termination order: "length < *mlex* > {}"

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
51 Recall the definition of paths from last exercise sheet:
52 >
53 fun path :: "('a ⇒ 'a ⇒ bool) ⇒ 'a ⇒ 'a list ⇒ 'a ⇒ bool"
54   where
55     "path G u [] v ⇔ u=v"
56   | "path G u (x#xs) v ⇔ G u x ∧ path G x xs v"
57
58 lemma path_append[simp]: "path G u (p1@p2) v ⇔ (∃w. path G u p1 w ∧ path G w p2 v)"
59   by (induction p1 arbitrary: u) auto
60
61
62 text <
63   A simple path is a path without loops, or, in other words, a path
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
60.1 (1467/5144) (Isabelle, Isabelle.UTF-8-Isabelle)Nmr o UG 31/9/1337MB 12:45 PM
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:45:59

```

```

consts
path :: "('a ⇒ 'a ⇒ bool) ⇒ 'a ⇒ 'a list ⇒ 'a ⇒ bool"
Found termination order: "(λp. length (fst (snd (snd p)))) < *mlex* > {}"

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
68 Hint: Induction on the length of the path
69 >
70 thm measure_induct_rule[where f=length, case_names shorter]
71
72
73 thm not_distinct_decomp
74
75 lemma exists_simple_path:
76   assumes "path G u p v"
77   shows "∃p'. path G u p' v ∧ distinct p'"
78   oops
79
80
81
82
83
84
85
86
87
88
89
90
77.24 (1960/5144) (Isabelle, Isabelle.UTF-8-Isabelle)Nmr o UG 31/9/1337MB 12:46 PM
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:46:19

```

```

distinct ?ws ⇒ ∃xs ys zs y. ?ws = xs @ [y] @ ys @ [y] @ zs

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
66 Show that for every path, there is a corresponding simple path.
67
68 Hint: Induction on the length of the path
69 >
70 thm measure_induct_rule[where f=length, case_names shorter]
71
72
73 thm not_distinct_decomp
74
75 lemma exists_simple_path:
76   assumes "path G u p v"
77   shows "∃p'. path G u p' v ∧ distinct p'"
78   oops
79
80
81
82
83
84
85
86
87
88
89
90
71.60 (1859/5144) (Isabelle, Isabelle.UTF-8-Isabelle)Nmr o UG 31/9/1337MB 12:48 PM
debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy Isabelle2017 - tut06.thy 12:48:14

```

```

(λx. (λy. length y < length x ⇒ ?P y) ⇒ ?P x) ⇒ ?P ?a

```



Isabelle2017 - tut06.thy (modified)

```

66 Show that for every path, there is a corresponding simple path.
67
68 Hint: Induction on the length of the path
69
70 >
71 thm measure_induct_rule
72 thm measure_induct_rule[where f=length, case_names shorter]
73
74 thm not_distinct_decomp
75
76 Lemma exists_simple_path:
77   assumes "path G u p v"
78   shows "∃p'. path G u p' v ∧ distinct p'"

```

Proof state  Auto update  Search:  100%

$$(\lambda x. (\lambda y. ?f\ y < ?f\ x \implies ?P\ y) \implies ?P\ x) \implies ?P\ ?a$$

71.24 (1.823/51.68) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 667/1.337MB 12:49 PM  
 debian [1] 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:49:09

Isabelle2017 - tut06.thy (modified)

```

66 Show that for every path, there is a corresponding simple path.
67
68 Hint: Induction on the length of the path
69
70 >
71 thm measure_induct_rule
72 thm measure_induct_rule[where f=length, case_names shorter]
73
74 thm not_distinct_decomp
75
76 Lemma exists_simple_path:
77   assumes "path G u p v"
78   shows "∃p'. path G u p' v ∧ distinct p'"

```

Proof state  Auto update  Search:  100%

$$(\lambda x. (\lambda y. \text{length } y < \text{length } x \implies ?P\ y) \implies ?P\ x) \implies ?P\ ?a$$

72.37 (1.860/51.68) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 677/1.337MB 12:49 PM  
 debian [1] 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:49:56

Isabelle2017 - tut06.thy (modified)

```

66 Show that for every path, there is a corresponding simple path.
67
68 Hint: Induction on the length of the path
69
70 >
71 thm measure_induct_rule
72 thm measure_induct_rule[where f=length, case_names shorter]
73
74 thm not_distinct_decomp
75
76 Lemma exists_simple_path:
77   assumes "path G u p v"
78   shows "∃p'. path G u p' v ∧ distinct p'"

```

Proof state  Auto update  Search:  100%

$$(\lambda x. (\lambda y. \text{length } y < \text{length } x \implies ?P\ y) \implies ?P\ x) \implies ?P\ ?a$$

72.40 (1.863/51.68) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 680/1.337MB 12:50 PM  
 debian [1] 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:50:18

Isabelle2017 - tut06.thy (modified)

```

66 Show that for every path, there is a corresponding simple path.
67
68 Hint: Induction on the length of the path
69
70 >
71 thm measure_induct_rule
72 thm measure_induct_rule[where f=length, case_names shorter]
73
74 thm not_distinct_decomp
75
76 Lemma exists_simple_path:
77   assumes "path G u p v"
78   shows "∃p'. path G u p' v ∧ distinct p'"

```

Proof state  Auto update  Search:  100%

$$(\lambda x. (\lambda y. \text{length } y < \text{length } x \implies ?P\ y) \implies ?P\ x) \implies ?P\ ?a$$

72.60 (1.883/51.68) (isabelle.isabelle.UTF-8-Isabelle)Nmr o UG 684/1.337MB 12:50 PM  
 debian [1] 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS18/public/exercises/ Isabelle2017 - tut06.thy (modified) Isabelle2017 - tut06.thy (modified) 12:50:34

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
71 thm measure_induct_rule
72 thm measure_induct_rule[where f=length, case_names shorter]
73
74 thm not_distinct_decomp
75
76 lemma exists_simple_path:
77   assumes "path G u p v"
78   shows "∃p'. path G u p' v ∧ distinct p'"
79
80
81
82 text <\NumHomework{Stability of Insertion Sort}{May 25}
83   Have a look at Isabelle's standard implementation of sorting: @{const sort_key}.

```

79.3 (2006/5164) (Isabelle, Isabelle, UTF-8-Isabelle) Nmr o UG 780/1:337MB 12:51 PM

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
73 thm not_distinct_decomp
74
75 lemma exists_simple_path:
76   assumes "path G u p v"
77   shows "∃p'. path G u p' v ∧ distinct p'"
78
79 proof (induction p rule: len_induct)
80
81
82 text <\NumHomework{Stability of Insertion Sort}{May 25}
83

```

```

proof (prove)
goal (1 subgoal):
1. ∃p'. path G u p' v ∧ distinct p'

```

78.1 (1977/5214) (Isabelle, Isabelle, UTF-8-Isabelle) Nmr o UG 565/1:282MB 12:52 PM

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
75 lemma exists_simple_path:
76   assumes "path G u p v"
77   shows "∃p'. path G u p' v ∧ distinct p'"
78
79 proof (induction p rule: len_induct)
80
81
82 text <\NumHomework{Stability of Insertion Sort}{May 25}
83   Have a look at Isabelle's standard implementation of sorting: @{const sort_key}.

```

```

proof (prove)
goal (1 subgoal):
1. ∃p'. path G u p' v ∧ distinct p'

```

79.3 (2022/5217) (Isabelle, Isabelle, UTF-8-Isabelle) Nmr o UG 600/0:282MB 12:53 PM

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
75 lemma exists_simple_path:
76   assumes "path G u p v"
77   shows "∃p'. path G u p' v ∧ distinct p'"
78
79 using assms
80 proof (induction p rule: len_induct)
81
82 text <\NumHomework{Stability of Insertion Sort}{May 25}
83   Have a look at Isabelle's standard implementation of sorting: @{const sort_key}.

```

```

proof (state)
goal (1 subgoal):
1. ∧x. [∧y. [length y < length x; path G u y v] ⇒ ∃p'. path G u p' v ∧ distinct p'; path G u x v]
⇒ ∃p'. path G u p' v ∧ distinct p'
Proof outline with cases:
case (shorter x)
then show ?case sorry
qed

```

80.14 (2047/5228) (Isabelle, Isabelle, UTF-8-Isabelle) Nmr o UG 711/0:82MB 12:54 PM

Isabelle2017 - tut06.thy (modified)

```

78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter x)
82 then show ?case sorry
83 qed
84
85 text <\NumHomework{Stability of Insertion Sort}(May 25)
86 Have a look at Isabelle's standard implementation of sorting: @{const sort key}

```

proof (state)  
goal (1 subgoal):  
1.  $\bigwedge x. [\bigwedge y. [\text{length } y < \text{length } x; \text{path } G u y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p']; \text{path } G u x v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$

Proof outline with cases:  
case (shorter x)  
then show ?case sorry  
qed

83.4 (2117/5275) Matches line 76: lemma exists\_simple\_path: (isabelle,isabelle,UTF-8-Isabelle)NmroUG 71.7/1.82MB 12:54 PM

Isabelle2017 - tut06.thy (modified)

```

77 assumes "path G u p v"
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter x)
82 then show ?case sorry
83 qed
84
85 text <\NumHomework{Stability of Insertion Sort}(May 25)

```

proof (state)  
this:  
1.  $[\text{length } ?y < \text{length } x; \text{path } G u ?y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$   
2.  $\text{path } G u x v$

goal (1 subgoal):  
1.  $\bigwedge x. [\bigwedge y. [\text{length } y < \text{length } x; \text{path } G u y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p']; \text{path } G u x v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$

81.19 (2089/5275) (isabelle,isabelle,UTF-8-Isabelle)NmroUG 786/1.82MB 12:54 PM

Isabelle2017 - tut06.thy (modified)

```

77 assumes "path G u p v"
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82 then show ?case sorry
83 qed
84
85 text <\NumHomework{Stability of Insertion Sort}(May 25)

```

proof (state)  
this:  
1.  $[\text{length } ?y < \text{length } p; \text{path } G u ?y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$   
2.  $\text{path } G u p v$

goal (1 subgoal):  
1.  $\bigwedge x. [\bigwedge y. [\text{length } y < \text{length } x; \text{path } G u y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p']; \text{path } G u x v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$

81.18 (2088/5275) (isabelle,isabelle,UTF-8-Isabelle)NmroUG 1.26/1.259MB 12:56 PM

Isabelle2017 - tut06.thy (modified)

```

77 assumes "path G u p v"
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82 then show ?case sorry
83 qed
84
85 text <\NumHomework{Stability of Insertion Sort}(May 25)

```

proof (state)  
this:  
1.  $[\text{length } ?y < \text{length } p; \text{path } G u ?y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$   
2.  $\text{path } G u p v$

goal (1 subgoal):  
1.  $\bigwedge x. [\bigwedge y. [\text{length } y < \text{length } x; \text{path } G u y v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p']; \text{path } G u x v] \Rightarrow \exists p'. \text{path } G u p' v \wedge \text{distinct } p'$

81.1 (2071/5275) (isabelle,isabelle,UTF-8-Isabelle)NmroUG 1.31/1.259MB 12:56 PM

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83
84 from not_distinct_decomp[of p].
85 then show ?case sorry
86 qed
87
proof (state)
this:
  · [length ?y < length p; path G u ?y v] ⇒ ∃p'. path G u p' v ∧ distinct p'
  · path G u p v
goal (1 subgoal):
1. ∧x. [∧y. [length y < length x; path G u y v] ⇒ ∃p'. path G u p' v ∧ distinct p'; path G u x v]
⇒ ∃p'. path G u p' v ∧ distinct p'

```

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83
84
85 from not_distinct_decomp[of p].
86 then show ?case sorry
87 qed
88
proof (state)
this:
  · [length ?y < length p; path G u ?y v] ⇒ ∃p'. path G u p' v ∧ distinct p'
  · path G u p v
goal (1 subgoal):
1. ∧x. [∧y. [length y < length x; path G u y v] ⇒ ∃p'. path G u p' v ∧ distinct p'; path G u x v]
⇒ ∃p'. path G u p' v ∧ distinct p'

```

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83
84
85 from not_distinct_decomp[of p].
86 then show ?case sorry
87 qed
88
proof (state)
this:
  · [length ?y < length p; path G u ?y v] ⇒ ∃p'. path G u p' v ∧ distinct p'
  · path G u p v
goal (1 subgoal):
1. ∧x. [∧y. [length y < length x; path G u y v] ⇒ ∃p'. path G u p' v ∧ distinct p'; path G u x v]
⇒ ∃p'. path G u p' v ∧ distinct p'

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83 show ?case proof cases
84 assume "distinct p" show ?thesis sorry
85 next
86 assume "-distinct p" show ?thesis sorry
87 qed
88
proof (prove)
goal (1 subgoal):
1. ∃p'. path G u p' v ∧ distinct p'

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83 show ?case proof cases
84   assume "distinct p" then show ?thesis using shorter.prem by blast
85 next
86   assume "~distinct p" show ?thesis sorry
87 qed
88
proof (state)
this:
  ~ distinct p

goal (1 subgoal):
1. ~ distinct p ==> exists p'. path G u p' v ^ distinct p'

```

86.23 (2216/5463) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 836/1216MB 1:00 PM  
 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy 13:00:39

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
84 assume "distinct p" then show ?thesis using shorter.prem by blast
85 next
86   assume "~distinct p"
87   from not_distinct_decomp[OF this] obt
88   ..
89   show ?thesis sorry
90 qed
91
from not_distinct_decomp[OF p1]

```

87.42 (2260/5523) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 93/1197MB 1:01 PM  
 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy (modified) 13:01:19

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
84 assume "distinct p" then show ?thesis using shorter.prem by blast
85 next
86   assume "~distinct p"
87   from not_distinct_decomp[OF this] obtain p1 p2 p3 u where "p=p1@u#p2@u#p3"
88   by auto
89   show ?thesis sorry
90 qed
91
from not_distinct_decomp[OF p]
then show ?case sorry

```

88.14 (2311/5561) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 86/1178MB 1:03 PM  
 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy 13:03:00

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
86 assume "~distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 u where "p=p1@u#p2@u#p3"
88 by auto
89
90 show ?thesis sorry
91 qed
92
from not_distinct_decomp[OF p1]

```

91.7 (2332/5582) (isabelle.isabelle,UTF-8-isabelle)tmr o UG 88/1178MB 1:03 PM  
 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy (modified) 13:03:34

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
85 next
86 assume "-distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89
90 hence "path G u (p1@x#p3) v"
91
92 show ?thesis sorry
93 qed
94
proof (prove)
using this:
  p = p1 @ x # p2 @ x # p3
goal (1 subgoal):
  1. path G u (p1 @ x # p3) v

```

90.33 (2345/5596) (isabelle.isabelle,UTF-8-isabelle)tmr o UG : :D/1139MB 1:04 PM  
 debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy 13:04:53

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
76 lemma exists_simple_path:
77 assumes "path G u p v"
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case [shorter p]
82
83 show ?case proof cases
84 assume "distinct p" then show ?thesis using shorter.prem by blast
85 next

```

81.19 (2089/5596) (isabelle.isabelle,UTF-8-isabelle)tmr o UG : :D/1139MB 1:05 PM  
 debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy 13:05:09

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
56 | "path G u (x#xs) v ↔ G u x ∧ path G x xs v"
57
58 lemma path_append[simp]: "path G u (p1@p2) v ↔ (∃w. path G u p1 w ∧ path G w p2 v)"
59 by (induction p1 arbitrary: u) auto
60
61 text <
62 A simple path is a path without loops, or, in other words, a path
63 where no node occurs twice. (Note that the first node of the path is
64 not included such that there may be a simple path from (u) to (u).)
65
theorem path_append: path ?G ?u (?p1.0 @ ?p2.0) ?v = (∃w. path ?G ?u ?p1.0 w ∧ path ?G w ?p2.0 ?v)

```

61.1 (1468/5596) (isabelle.isabelle,UTF-8-isabelle)tmr o UG : :D/1139MB 1:05 PM  
 debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy 13:05:29

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Magros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
85 next
86 assume "-distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89 hence "undefined (path G u p v)" apply simp.
90
91 hence "path G u (p1@x#p3) v"
92
93 show ?thesis sorry
94 qed
95
proof (prove)
goal (1 subgoal):
  1. p = p1 @ x # p2 @ x # p3 ⇒
    undefined (∃w. path G u p1 w ∧ G w x ∧ (∃w. path G x p2 w ∧ G w x ∧ path G x p3 v))

```

89.37 (2348/5646) (isabelle.isabelle,UTF-8-isabelle)tmr o UG : :D/1123MB 1:06 PM  
 debian 1 2 3 4 iamlich@lapnikow10: ~/lehre/FDS/SS1... Isabelle2017 - tut06.thy (modified) 13:06:03

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
85 next
86 assume "-distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89 hence "foo" using shorter.premis apply clarify
90
91 hence "path G u (p1@x#p3) v"
92
93 show ?thesis sorry
94 qed
95
proof (prove)
goal (1 subgoal):
1.  $\bigwedge w$  wa.  $[p = p1 @ x \# p2 @ x \# p3; \text{path } G \text{ u } p1 \text{ w}; G \text{ w } x; \text{path } G \text{ x } p2 \text{ wa}; G \text{ wa } x; \text{path } G \text{ x } p3 \text{ v}] \Rightarrow \text{foo}$ 

```

89.47 (2358/5647) (isabelle.isabelle.UTF-8-isabelle)lmmr o UG 81.7/1.23MB 1:07 PM

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
84 assume "distinct p" then show ?thesis using shorter.premis by blast
85 next
86 assume "-distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89 hence "path G u (p1@x#p3) v" using shorter.premis by auto
90
91 show ?thesis sorry
92 qed
93
have path G u (p1 @ x # p3) v
proof (state)
this:
path G u (p1 @ x # p3) v
goal (1 subgoal):
1.  $\neg \text{distinct } p \Rightarrow \exists p'. \text{path } G \text{ u } p' \text{ v} \wedge \text{distinct } p'$ 

```

87.1 (2219/5623) (isabelle.isabelle.UTF-8-isabelle)lmmr o UG 33/1.105MB 1:07 PM

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
88 by auto
89
90 from shorter.premis obtain x1 x2 where
91 "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
92 by auto
93 hence "path G u (p1@x#p3) v" using shorter.premis by auto
94
95 show ?thesis sorry
96 qed
97
have ( $\bigwedge p1 \text{ p2 } p3 \text{ x. } p = p1 @ x \# p2 @ x \# p3 \Rightarrow ?thesis$ )  $\Rightarrow ?thesis$ 
proof (state)
this:
p = p1 @ x # p2 @ x # p3
goal (1 subgoal):
1.  $\neg \text{distinct } p \Rightarrow \exists p'. \text{path } G \text{ u } p' \text{ v} \wedge \text{distinct } p'$ 

```

93.1 (2451/5763) (isabelle.isabelle.UTF-8-isabelle)lmmr o UG 365/1.059MB 1:09 PM

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
89 with shorter.premis obtain x1 x2 where
90 "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
91 by auto
92 have "path G u (p1@x#p3) v" apply auto
93
94 show ?thesis sorry
95 qed
96
proof (prove)
goal (1 subgoal):
1. path G u (p1 @ x # p3) v

```

92.9 (2450/5735) (isabelle.isabelle.UTF-8-isabelle)lmmr o UG 28/1.018MB 1:11 PM

```
Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Magros Plugins Help

tut06.thy (~/lehre/FDS/SS18/public/exercises/)
86 assume "--distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89 with shorter.prem obtain x1 x2 where
90   "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
91 by auto
92 then have "path G u (p1@x#p3) v" by auto
93
94
95 show ?thesis sorry
96 qed
97
98

have path G u (p1 @ x # p3) v
proof (state)
this:
path G u (p1 @ x # p3) v
end
```

```
Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Magros Plugins Help

tut06.thy (~/lehre/FDS/SS18/public/exercises/)
86 assume "--distinct p"
87 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where "p=p1@x#p2@x#p3"
88 by auto
89 with shorter.prem obtain x1 x2 where
90   "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
91 by auto
92 then have "path G u (p1@x#p3) v" by auto
93
94
95 thm shorter.IH[OF _ this]...
96 show ?thesis sorry
97 qed
98

from not_distinct_decomp[of p]

proof (prove)
using this:
x s y s z s y. p = xs @ [y] @ ys @ [y] @ zs
goal (1 subgoal):
end
```

```
Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help

tut06.thy (~/lehre/FDS/SS18/public/exercises/)
81 case (shorter p)
82 show ?case proof cases
83 assume "distinct p" then show ?thesis using shorter.prem by blast
84 next
85 assume "--distinct p"
86 from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2@x#p3"
87 by auto
88 from shorter.prem obtain x1 x2 where
89   "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
90 by auto
91 then have "path G u (p1@x#p3) v" by auto
92
93 from shorter.IH[OF _ this] show ?thesis by simp
94 qed
95 qed
96

proof (prove)
using this:
path G u p v
end
```

```
Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Magros Plugins Help

tut06.thy (~/lehre/FDS/SS18/public/exercises/)
81 proof (induction p rule: len_induct)
82   case (shorter p)
83   show ?case proof cases
84     assume "distinct p" then show ?thesis using shorter.prem by blast
85   next
86     assume "--distinct p"
87     from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2@x#p3"
88     by auto
89     from shorter.prem obtain x1 x2 where
90       "path G u p1 x1" "G x1 x" "path G x p2 x2" "G x2 x" "path G x p3 v"
91     by auto
92   qed
93 qed

proof (state)
goal (2 subgoals):
1. ?P ==> ?p'. path G u p' v ^ distinct p'
2. ~ ?P ==> ?p'. path G u p' v ^ distinct p'
Proof outline with cases:
case True
then show ?thesis sorry
next
end
```



Isabelle2017 - tut06.thy (modified)

```

77 assumes "path G u p v"
78 shows "∃p'. path G u p' v ∧ distinct p'"
79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter x)
82
83 term x term p
84
85 show ?case proof cases
86   assume "distinct p" then show ?thesis using shorter.prem1 oops by blast
87 next
88   assume "-distinct p"
89   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2"
90
91   proof (prove)
92     using this:
93       · distinct p
94       · path G u x v
95
96     goal (1 subgoal):
97       1. ∃p'. path G u p' v ∧ distinct p'
98
99   end
100
101   next
102     show ?thesis using shorter.prem2 oops by blast
103   end
104 end

```

83.16 (2106/5739) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 449/448MB 1:18 PM

Isabelle2017 - tut06.thy (modified)

```

79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter x)
82
83 term x term p
84
85 show ?case proof cases
86   assume "distinct p" then show ?thesis using shorter.prem1 oops by blast
87 next
88   assume "-distinct p"
89   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2"
90
91   proof (prove)
92     using this:
93       · distinct p
94       · path G u x v
95
96     goal (1 subgoal):
97       1. ∃p'. path G u p' v ∧ distinct p'
98
99   end
100
101   next
102     show ?thesis using shorter.prem2 oops by blast
103   end
104 end

```

82.1 (2090/5739) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 138/938MB 1:19 PM

Isabelle2017 - tut06.thy (modified)

```

79 using assms
80 proof (induction p rule: len_induct)
81 case (shorter p)
82
83 thm shorter.prem1 assms
84
85 show ?case proof cases
86   assume "distinct p" then show ?thesis using shorter.prem1 oops by blast
87 next
88   assume "-distinct p"
89   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2"
90
91   proof (prove)
92     using this:
93       · distinct p
94       · path G u p v
95
96     goal (1 subgoal):
97       1. ∃p'. path G u p' v ∧ distinct p'
98
99   end
100
101   next
102     show ?thesis using shorter.prem2 oops by blast
103   end
104 end

```

83.23 (2113/5752) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 34/929MB 1:20 PM

Isabelle2017 - tut06.thy (modified)

```

81 case (shorter p)
82
83 thm shorter.prem1 assms
84
85 show ?case proof cases
86   assume "distinct p" then show ?thesis using shorter.prem1 oops by blast
87 next
88   assume "-distinct p"
89   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2"
90
91   proof (prove)
92     using this:
93       · distinct p
94       · path G u p v
95
96     goal (1 subgoal):
97       1. ∃p'. path G u p' v ∧ distinct p'
98
99   end
100
101   next
102     show ?thesis using shorter.prem2 oops by blast
103   end
104 end

```

87.53 (2206/5749) Input/output complete (isabelle.isabelle.UTF-8-isabelle)tmr o UG 133/929MB 1:21 PM

```

Isabelle2017 - tut06.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
80 proof (induction p rule: len_induct)
81   case (shorter p)
82
83   thm shorter.premss assms
84
85
86 show ?case proof cases
87   assume "distinct p" then show ?thesis using PATH oops by blast
88 next
89   assume "--distinct p"
90   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2@x#p3"
91
92 proof (prove)
93 using this:
94   distinct p
95   path G u p v
96
97 goal (1 subgoal):
98 1.  $\exists p'. \text{path } G \ u \ p' \ v \wedge \text{distinct } p'$ 
99
100
101 Output Query Sledgehammer Symbols
83.26 (2122/5749) (isabelle,isabelle,UTF-8-isabelle)nmr o UG 44.2/129MB 1:21 PM

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
80 proof (induction p rule: len_induct)
81   case (shorter p)
82
83 show ?case proof cases
84   assume "distinct p" then show ?thesis using shorter.premss by blast
85 next
86   assume "--distinct p"
87   from not_distinct_decomp[OF this] obtain p1 p2 p3 x where [simp]: "p=p1@x#p2@x#p3"
88   by auto
89   from shorter.premss obtain x1 x2 where
90     "path G u p1 x1" "G x1 x2" "path G x p2 x2" "G x2 x" "path G x p3 v"
91   by auto
92   then have "path G u (p1@x#p3) v" by auto
93
94   from shorter.IH[OF _ this] show ?thesis by simp
95
96 show  $\exists p'. \text{path } G \ u \ p' \ v \wedge \text{distinct } p'$ 
97 Successful attempt to solve goal by exported rule:
98 ( $\text{distinct } p$ )  $\Rightarrow \exists p'. \text{path } G \ u \ p' \ v \wedge \text{distinct } p'$ 
99 proof (state)
100
101 Output Query Sledgehammer Symbols
85.1 (2187/5715) (isabelle,isabelle,UTF-8-isabelle)nmr o UG 37/910MB 1:22 PM

```

```

Isabelle2017 - tut06.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut06.thy (~/lehre/FDS/SS18/public/exercises/)
95 qed
96 qed
97
98 text <NumHomework{Stability of Insertion Sort}{May 25}>
99 Have a look at Isabelle's standard implementation of sorting: @{const sort_key}.
100 (Use Ctrl-Click to jump to the definition in @{file "~/src/HOL/List.thy"})
101 Show that this function is a stable sorting algorithm, i.e., the order of elements
102 with the same key is not changed during sorting!
103
104
105
106 lemma "[x←sort_key k xs. k x = a] = [x←xs. k x = a]"
107 oops
108
109 term "[x←xs. P x]"
110 text <
111
112 theorem exists_simple_path:
113 path ?G ?u ?p ?v  $\Rightarrow \exists p'. \text{path } ?G \ ?u \ p' \ ?v \wedge \text{distinct } p'$ 
114
115
116 Output Query Sledgehammer Symbols
99.1 (2559/5715) (isabelle,isabelle,UTF-8-isabelle)nmr o UG 93/901MB 1:22 PM

```

```

Isabelle2017 - List.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
List.thy (ISABELLE_HOME/src/HOL)
366 "sorted [x]  $\leftrightarrow$  True"
367 by simp_all
368
369 primrec insort_key :: "('b  $\Rightarrow$  'a)  $\Rightarrow$  'b list  $\Rightarrow$  'b list" where
370 "insort_key f x [] = [x]" |
371 "insort_key f x (y#ys) =
372   (if f x  $\leq$  f y then (x#y#ys) else y#(insort_key f x ys))"
373
374 definition sort_key :: "('b  $\Rightarrow$  'a)  $\Rightarrow$  'b list  $\Rightarrow$  'b list" where
375 "sort_key f xs = foldr (insort_key f) xs []"
376
377 definition insort_insert_key :: "('b  $\Rightarrow$  'a)  $\Rightarrow$  'b  $\Rightarrow$  'b list  $\Rightarrow$  'b list" where
378 "insort_insert_key f x xs =
379   (if f x  $\in$  f ` set xs then xs else insort_key f x xs)"
380
381 abbreviation "sort = sort_key (λx. x)"
382
383
384 Output Query Sledgehammer Symbols
375.33 (13058/236142) (isabelle,isabelle,UTF-8-isabelle)nmr o UG 89/896MB 1:23 PM

```

Isabelle2017 - tut06.thy

```

100 text <\NumHomework{Stability of Insertion Sort}{May 25}
101 Have a look at Isabelle's standard implementation of sorting: @{{const sort_key}}.
102 (Use Ctrl-Click to jump to the definition in @{{file "~/src/HOL/List.thy"}})
103 Show that this function is a stable sorting algorithm, i.e., the order of elements
104 with the same key is not changed during sorting!
105
106 Lemma "[x←sort_key k xs. k x = a] = [x←xs. k x = a]"
107 oops
108
109 term "[x←xs. P x]"
110 text <

```

proof (prove)  
goal (1 subgoal):  
1. [x←sort\_key k xs . k x = a] = [x←xs . k x = a]

106.16 (2931/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 409/396MB 1:23 PM

Isabelle2017 - tut06.thy

```

100 text <\NumHomework{Stability of Insertion Sort}{May 25}
101 Have a look at Isabelle's standard implementation of sorting: @{{const sort_key}}.
102 (Use Ctrl-Click to jump to the definition in @{{file "~/src/HOL/List.thy"}})
103 Show that this function is a stable sorting algorithm, i.e., the order of elements
104 with the same key is not changed during sorting!
105
106 Lemma "[x←sort_key k xs. k x = a] = [x←xs. k x = a]"
107 oops
108
109 term "[x←xs. P x]"
110 text <

```

106.11 (2926/5715) null parsing complete, 0 error(s) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 57/879MB 1:25 PM

Isabelle2017 - tut06.thy

```

100 text <\NumHomework{Stability of Insertion Sort}{May 25}
101 Have a look at Isabelle's standard implementation of sorting: @{{const sort_key}}.
102 (Use Ctrl-Click to jump to the definition in @{{file "~/src/HOL/List.thy"}})
103 Show that this function is a stable sorting algorithm, i.e., the order of elements
104 with the same key is not changed during sorting!
105
106 Lemma "[x←sort_key k xs. k x = a] = [x←xs. k x = a]"
107 oops
108
109 term "[x←xs. P x]"
110 text <
111 Note: @{{term [source] <[x←xs. P x]>}} is syntax sugar for @{{term [source] <filter P xs>}}.

```

proof (prove)  
goal (1 subgoal):  
1. [x←sort\_key k xs . k x = a] = [x←xs . k x = a]

106.42 (2957/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 77/879MB 1:26 PM

Isabelle2017 - tut06.thy

```

103 Show that this function is a stable sorting algorithm, i.e., the order of elements
104 with the same key is not changed during sorting!
105
106 Lemma "[x←sort_key k xs. k x = a] = [x←xs. k x = a]"
107 oops
108
109 term "[x←xs. P x]"
110 text <
111 Note: @{{term [source] <[x←xs. P x]>}} is syntax sugar for @{{term [source] <filter P xs>}}.
112 where the filter function returns only the elements of list <xs> for which <P xs = True>.
113
114 Hint: You do not necessarily need Isar, and the auxiliary lemmas
115 you need are already in Isabelle's library. @{{command find_theorems}} is your friend!

```

111.1 (3003/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 86/879MB 1:27 PM



Isabelle2017 - tut06.thy

```

139 fun quickselect :: "'a::linorder list ⇒ nat ⇒ 'a" where
140 "quickselect (x#xs) k = (let
141   xs1 = [y←xs. y<x];
142   xs2 = [y←xs. ¬(y<x)]
143 in
144   if k<length xs1 then quickselect xs1 k
145   else if k=length xs1 then x
146   else quickselect xs2 (k-length xs1-1)
147 )"
148 | "quickselect [] _ = undefined"
149
150 text <Your first task is to prove the crucial idea of quicksort, i.e., that
151 partitioning wrt. \ a pivot element $p$ is correct.
152 >
153
154

```

consts

```

quickselect :: "'a list ⇒ nat ⇒ 'a"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. length (fst p)) <*mlex*> {}"

```

143.1 (4495/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 372/871MB 1:30 PM

Isabelle2017 - tut06.thy

```

139 fun quickselect :: "'a::linorder list ⇒ nat ⇒ 'a" where
140 "quickselect (x#xs) k = (let
141   xs1 = [y←xs. y<x];
142   xs2 = [y←xs. ¬(y<x)]
143 in
144   if k<length xs1 then quickselect xs1 k
145   else if k=length xs1 then x
146   else quickselect xs2 (k-length xs1-1)
147 )"
148 | "quickselect [] _ = undefined"
149
150 text <Your first task is to prove the crucial idea of quicksort, i.e., that
151 partitioning wrt. \ a pivot element $p$ is correct.
152 >
153
154

```

consts

```

quickselect :: "'a list ⇒ nat ⇒ 'a"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. length (fst p)) <*mlex*> {}"

```

144.5 (4524/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 86/865MB 1:30 PM

Isabelle2017 - tut06.thy

```

147   else quickselect xs2 (k-length xs1-1)
148   )"
149 | "quickselect [] _ = undefined"
150
151 text <Your first task is to prove the crucial idea of quicksort, i.e., that
152 partitioning wrt. \ a pivot element $p$ is correct.
153 >
154
155
156 lemma partition_correct: "sort xs = sort [x←xs. x<p] @ sort [x←xs. ¬(x<p)]"
157 oops
158
159 text <
160 Hint: Induction, and auxiliary lemmas to transform a term of the
161 form @{term <insert x (xs@ys)>} when you know that <x> is greater than
162 all elements in <xs> / less than or equal all elements in <ys>.
163 >

```

proof (prove)

```

goal (1 subgoal):
1. sort xs = sort [x←xs. x < p] @ sort [x←xs. ¬ x < p]

```

156.32 (4845/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 30/865MB 1:31 PM

Isabelle2017 - tut06.thy

```

147   else quickselect xs2 (k-length xs1-1)
148   )"
149 | "quickselect [] _ = undefined"
150
151 text <Your first task is to prove the crucial idea of quicksort, i.e., that
152 partitioning wrt. \ a pivot element $p$ is correct.
153 >
154
155
156 lemma partition_correct: "sort xs = sort [x←xs. x<p] @ sort [x←xs. ¬(x<p)]"
157 oops
158
159 text <
160 Hint: Induction, and auxiliary lemmas to transform a term of the
161 form @{term <insert x (xs@ys)>} when you know that <x> is greater than
162 all elements in <xs> / less than or equal all elements in <ys>.
163 >

```

157.7 (4896/5715) (isabelle.isabelle.UTF-8-isabelle)tmr o UG 45/865MB 1:31 PM

```

151 text <Your first task is to prove the crucial idea of quicksort, i.e., that
152 partitioning wrt.\ a pivot element $p$ is correct.
153 >
154
155 lemma partition_correct: "sort xs = sort [x←xs. x<p] @ sort [x←xs. ¬(x<p)]"
156 oops
157
158 text
159 [
160 Hint: Induction, and auxiliary lemmas to transform a term of the
161 form @{term <insert x (xs@ys)>} when you know that <x> is greater than
162 all elements in <xs> / less than or equal all elements in <ys>.
163 ]
164
165
166

```

Proof state: [x] Auto update: [x] Update Search: [ ] 100%

```

169 text <Proceed by computation induction, and a case distinction according to the
170 cases in the body of the quickselect function>
171 proof (induction xs k rule: quickselect.induct)
172 case (1 x xs k)
173
174 text <Note: To make the induction hypothesis more readable,
175 you can collapse the first two premises of the form <?x=...>
176 by reflexivity:>
177 note IH = "1.IH"[OF refl refl]
178
179 text <Insert your proof here!>

```

Proof state: [x] Auto update: [x] Update Search: [ ] 100%

```

proof (state)
this:
  • [k < length [y←xs . y < x]; k < length [y←xs . y < x]]
    ⇒ quickselect [y←xs . y < x] k = sort [y←xs . y < x] ! k
  • [¬ k < length [y←xs . y < x]; k ≠ length [y←xs . y < x];
    k - length [y←xs . y < x] - 1 < length [y←xs . ¬ y < x]]
    ⇒ quickselect [y←xs . ¬ y < x] (k - length [y←xs . y < x] - 1) =
      sort [y←xs . ¬ y < x] ! (k - length [y←xs . y < x] - 1)

```

```

169 text <Proceed by computation induction, and a case distinction according to the
170 cases in the body of the quickselect function>
171 proof (induction xs k rule: quickselect.induct)
172 case (1 x xs k)
173
174 text <Note: To make the induction hypothesis more readable,
175 you can collapse the first two premises of the form <?x=...>
176 by reflexivity:>
177 note IH = "1.IH"[OF refl refl]
178
179 text <Insert your proof here!>

```

Proof state: [x] Auto update: [x] Update Search: [ ] 100%

```

proof (state)
this:
  • [k < length [y←xs . y < x]; k < length [y←xs . y < x]]
    ⇒ quickselect [y←xs . y < x] k = sort [y←xs . y < x] ! k
  • [¬ k < length [y←xs . y < x]; k ≠ length [y←xs . y < x];
    k - length [y←xs . y < x] - 1 < length [y←xs . ¬ y < x]]
    ⇒ quickselect [y←xs . ¬ y < x] (k - length [y←xs . y < x] - 1) =
      sort [y←xs . ¬ y < x] ! (k - length [y←xs . y < x] - 1)

```

```

145 if k < length xs1 then quickselect xs1 k
146 else if k = length xs1 then x
147 else quickselect xs2 (k - length xs1 - 1)
148 )"
149 "quickselect [] _ = undefined"
150
151
152 text <Your first task is to prove the crucial idea of quicksort, i.e., that
153 partitioning wrt.\ a pivot element $p$ is correct.
154 >
155

```

Proof state: [x] Auto update: [x] Update Search: [ ] 100%

```

consts
quickselect :: "'a list ⇒ nat ⇒ 'a"
Found termination order: "(λp. size (snd p)) <+mlex+> (λp. length (fst p)) <+mlex+> {}"

```