

The `erw-l3` package*

January 27, 2022

Abstract

Provides utilities based on L^AT_EX3[1], such as `\erw_merge_sort:nNn`.

Contents

I	Usage	2
1	boilerplate	2
2	quark	2
3	predicate	3
4	keyval	3
5	op's on lists	3
6	Algo	3
7	code	4
II	Other	4
1	Bibliography	4
2	Support	4
III	Implementation	4
1	kernel	4
2	boilerplate	4
3	quark	5

*This file describes version v4.0, last revised 2022-01-13.

4	predicate	6
5	keyval	9
6	op's on list	10
7	Algo	13
7.1	split	13
7.2	thread sort	15
7.3	merge sort	16
7.4	filter	18
8	code	19
	Index	23

Part I

Usage

1 boilerplate

<code>\erw_keys_set:n</code>	<code>\erw_keys_set:n{⟨<i>keyval list</i>⟩}</code>
<code>\erw_keys_set:nn</code>	

<code>\erw_identity:n</code>	★
<code>\erw_int_incr:n</code>	★
<code>\erw_swap:nn</code>	★
<code>\erw_swap:ne</code>	★
<code>\erw_name_signature_cs:N</code>	★

2 quark

<code>\erw_all_q:w</code>	<code>\erw_remove_last_q:w⟨<i>tokenlist</i>⟩ \q_recursion_tail\q_recursion_stop</code>
<code>\erw_remove_first_q:w</code>	
<code>\erw_first_q:w</code>	
<code>\erw_remove_last_q:w</code>	
<code>\erw_last_q:w</code>	

3 predicate

<code>new_compare_p</code>	<code>\erw_keys_set:n{ new_compare_p = {<name>}{<signature>}{<predicate>} }</code>
----------------------------	--

Instance

`erw_compare_p:nNnNn`

`erw_compare_recurse_p:nnN`

`erw_int_incr_p:nn`

`erw_key_compare_p:nNn`

`erw_key_compare_p:n`

4 keyval

<code>\erw_keyval_key:n</code>
<code>\erw_keyval_value:n</code>
<code>\erw_keyval:nn</code>
<code>\erw_keyval_dispatch:NNn</code>
<code>\erw_keyval_dispatch_protected:NNn</code>

5 op's on lists

<code>\erw_remove_first:n</code>
<code>\erw_remove_last:n</code>
<code>\erw_first:n</code>
<code>\erw_last:n</code>
<code>\erw_adjacent_insert:nn</code>
<code>\erw_adjacent_insert:en</code>

6 algo

<code>\erw_split_even:n</code>	<code>\erw_thread_sort:nnNn{<first sorted list>}{<second sorted list>}{<compare predicate name>}< ></code>
<code>\erw_split_even:e</code>	
<code>\erw_merge_sort:nNn</code>	<code>\erw_merge_sort:nNn{<compare predicate name>}< >{<unsorted list>}</code>
<code>\thread_sort:nnNn</code>	<code>\erw_filter_uniq:nn{<compare predicate>}{<tokenlist>}</code>
<code>\erw_filter_uniq:nn</code>	<code>\erw_filter_uniq:n{<ascending intergers>}</code>
<code>\erw_filter_uniq:n</code>	

7 code

<code>\erw_parameter:n</code>	<code>\erw_parameter:n{<arity>}</code>
<code>\erw_parameter:nn</code>	<code>\erw_parameter:nn{<start pos>}{<arity>}</code>
<code>\argument:nn</code>	<code>\erw_argument:nn{<start pos>}{<signature>}</code>
<code>\erw_code_analyze:n</code>	
<code>\erw_signature:n</code>	

Part II Other

1 Bibliograhyy

- [1] The L^AT_EX3 Project Team. *The L^AT_EX3 interfaces*. <https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf>. 2019.

2 Support

This package is available from <https://github.com/rogard/erw-l3>.

Part III Implementation

```
1 <*package>
2 <@@=erw>
3 %      \ExplSyntaxOn
```

1 kernel

```
4 \cs_generate_variant:Nn\int_compare_p:nNn{eNe}
5 \cs_generate_variant:Nn\int_eval:n{e}
6 \cs_generate_variant:Nn\prg_new_conditional:Nnn{c}
7 \cs_generate_variant:Nn\prg_replicate:nn{e}
8 \cs_generate_variant:Nn\regex_gset:Nn{c}
9 \cs_generate_variant:Nn\regex_log:N{c}
10 \cs_generate_variant:Nn\regex_match:NnTF{c}
11 \cs_generate_variant:Nn\tl_to_str:n{e}
12 \cs_generate_variant:Nn\prop_put:Nnn{Nne}
```

2 boilerplate

```
13 \msg_new:nnnn{__erw}{text}{text~is~not~loaded}{load~amsmath}
14 \cs_new:Npn \__erw_text:n #1
15 {\cs_if_exist:NTF\text{#1}}{\msg_error:nn{__erw}{text}}
16 \cs_new:Npn \__erw_empty:w #1 \q_recursion_stop {\c_empty_tl}
17 \cs_new_protected:Nn \erw_keys_set:n{ \keys_set:nn{__erw}{#1} }
```

```

18 \cs_new_protected:Nn\erw_keys_set:nn{ \keys_set:nn{__erw / #1}{#2} }
19 \cs_generate_variant:Nn\erw_apply:Nw{c}
20 \cs_new:Npn \erw_identity:n#1{#1}
21 \cs_new:Npn \erw_int_incr:n#1{\int_eval:n{#1+1}}
22 \cs_new:Npn \erw_swap:nn#1#2{#2#1}
23 \cs_generate_variant:Nn \erw_swap:nn{e}
24 \cs_new:Npn \erw_name_signature_cs:N #1
25 { \exp_last_unbraced:Ne
26   \__erw_name_signature_cs:nnn{\cs_split_function:N#1}}
27 \cs_new:Nn \__erw_name_signature_cs:nnn{{#1}{#2}}

```

3 quark

```

28 \msg_new:nnn{erw}{quark-only-tail}
29 {requires~tail;~got~'~#1';~\msg_line_context:}
30 \cs_new:Npn
31 \erw_all_q:w
32 #1
33 \q_recursion_stop
34 {%
35   \erw_remove_last_q:w#1\q_recursion_stop
36   \erw_last_q:w#1\q_recursion_stop
37 }
38 \cs_new:Npn
39 \erw_remove_first_q:w
40 #1 % <tokenlist ending with recursion tail>
41 \q_recursion_stop
42 {\quark_if_recursion_tail_stop:n{#1}
43   \__erw_remove_first_q:nw#1\q_recursion_stop}
44 \cs_new:Npn
45 \__erw_remove_first_q:nw
46 #1 % <head>
47 #2 % <rest>
48 \q_recursion_stop
49 {\erw_remove_last_q:w#2\q_recursion_stop
50   \erw_last_q:w#2\q_recursion_stop}
51 \cs_new:Npn
52 \erw_first_q:w
53 #1
54 \q_recursion_stop
55 {%
56   \quark_if_recursion_tail_stop:n{#1}
57   \__erw_first_q:enw{ \tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
58 \cs_new:Npn
59 \__erw_first_q:nnw
60 #1 % <head is group>
61 #2 % <head>
62 #3 % <rest>
63 \q_recursion_stop
64 {%
65   \bool_if:nTF{#1}{#2}}{#2}
66 }
67 \cs_generate_variant:Nn\__erw_first_q:nnw{e}
68 \cs_new:Npn

```

```

69 \erw_remove_last_q:w #1 \q_recursion_stop
70 {%
71   \quark_if_recursion_tail_stop:n{#1}
72   \__erw_remove_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
73 \cs_new:Npn
74 \__erw_remove_last_q:nw
75 #1 % <head is group>
76 #2 % <tokenlist>
77 \q_recursion_stop
78 { \__erw_remove_last_q:nnw{#1}#2\q_recursion_stop }
79 \cs_generate_variant:Nn\__erw_remove_last_q:nw{e}
80 \cs_new:Npn
81 \__erw_remove_last_q:nnw
82 #1 % <head is group>
83 #2 % <head>
84 #3 % <rest>
85 \q_recursion_stop
86 {%
87   \quark_if_recursion_tail_stop:n{#3}
88   \bool_if:nTF{#1}{#2}{#2}
89   \__erw_remove_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
90 }
91 \cs_generate_variant:Nn\__erw_remove_last_q:nnw{e}
92 \cs_new:Npn
93 \erw_last_q:w #1 \q_recursion_stop
94 {\quark_if_recursion_tail_stop:n{#1}
95   \__erw_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop}
96 \cs_new:Npn
97 \__erw_last_q:nw
98 #1 % <head is group>
99 #2 % <tokenlist>
100 \q_recursion_stop
101 { \__erw_last_q:nnw{#1}#2\q_recursion_stop }
102 \cs_generate_variant:Nn\__erw_last_q:nw{e}
103 \cs_new:Npn
104 \__erw_last_q:nnw
105 #1 % <head is group>
106 #2 % <head>
107 #3 % <rest>
108 \q_recursion_stop
109 {%
110   \quark_if_recursion_tail_stop_do:nn{#3}{ \bool_if:nTF{#1}{#2}{#2} }
111   \__erw_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
112 }
113 \cs_generate_variant:Nn\__erw_last_q:nnw{e}

```

4 predicate

```

114 \msg_new:nnn{\__erw}{predicate-empty}
115 {empty~expression~in~predicate}
116 \prg_new_conditional:Npnn
117 \erw_and_tl:nn
118 #1 % <predicate expression>
119 #2 % <tokens>

```

```

120 {p}
121 {%^~A
122   \__erw_and_tl:nw {#1}#2 \q_recursion_tail\q_recursion_stop
123 }
124 \cs_new:Npn
125   \__erw_and_tl:nw
126   #1 % <predicate expression>
127   #2 % <value>
128   \q_recursion_stop
129   {%
130     \quark_if_recursion_tail_stop_do:nn{#2}
131     { \prg_return_true: }
132     \__erw_and_tl:nnw
133     {#1} % <predicate expression>
134     #2 % <value>
135     \q_recursion_stop
136   }
137 \cs_new:Npn
138   \__erw_and_tl:nnw
139   #1 % <predicate expression>
140   #2 % <value>
141   #3 % <rest>
142   \q_recursion_stop
143   {%
144     \bool_if:nTF
145     {#1{#2}}
146     {\__erw_and_tl:nw{#1}#3\q_recursion_stop}
147     { \prg_return_false: }
148   }
149 \cs_new:Npn \__erw_new_compare_p:nnn
150   #1 % <name>
151   #2 % <signature>
152   #3 % <code>
153   {%
154     \prg_new_conditional:cnn{#1:#2}
155     {p}
156     {%
157       \bool_if:nTF
158       {#3}
159       {\prg_return_true:}
160       {\prg_return_false:}
161     }
162   }
163 \keys_define:nn{ __erw }
164 {
165   new_compare_p.code:n = {\__erw_new_compare_p:nnn#1}
166 }
167 \erw_keys_set:n
168 {%
169   new_compare_p =
170   {erw_compare} % <name>
171   {nNnNn}
172   { \__erw_compare:eecN{ #2{#3} }{ #2{#5} }{ #1:nNn }#4 }
173 }

```

```

174 \cs_new:Npn
175 \__erw_compare:nnNN
176 #1 % <first>
177 #2 % <second>
178 #3 % <predicate>
179 #4 % <operator>
180 { #3{ #1 }#4{ #2 } }
181 \cs_generate_variant:Nn\__erw_compare:nnNN{eec}
182 \erw_keys_set:n
183 {%
184   new_compare_p =
185   {erw_compare_recurse} % <name>
186   {nnN}
187   { \__erw_compare_recurse:nnN{#1}{#2}#3 }
188 }
189 \cs_new:Npn
190 \__erw_compare_recurse:nnN
191 #1 % <tokenlist>
192 #2 % <compare predicate name>
193 #3 % <operator>
194 {%
195   \erw_compare_recurse_p:nnNN
196   {#1} % <tokenlist>
197   {#2} % <compare predicate name>
198   \erw_identity:n % <function>
199   #3 % <operator>
200 }
201 \erw_keys_set:n
202 {%
203   new_compare_p =
204   {erw_compare_recurse} % <name>
205   {nnNN}
206   { \__erw_compare_recurse:nnNN{#1}{#2}#3#4 }
207 }
208 \cs_new:Npn
209 \__erw_compare_recurse:nnNN
210 #1 % <tokenlist>
211 #2 % <compare predicate name>
212 #3 % <function>
213 #4 % <operator>
214 { \__erw_compare_recurse:nnNw {#2}#3#4#1 \q_recursion_tail\q_recursion_stop }
215 \cs_new:Npn
216 \__erw_compare_recurse:nnNw
217 #1 % <compare predicate name>
218 #2 % <convert>
219 #3 % <operator>
220 #4 % <tokenlist>
221 \q_recursion_stop
222 {%
223   \quark_if_recursion_tail_stop_do:nn{#4}{\c_true_bool}
224   \__erw_compare_recurse:nnNnw{#1}{#2}{#3}#4\q_recursion_stop
225 }
226 \cs_new:Npn
227 \__erw_compare_recurse:nnNnw

```



```

228 #1 % <compare predicate name>
229 #2 % <convert>
230 #3 % <operator>
231 #4 % <head>
232 #5 % <rest>
233 \q_recursion_stop
234 {%
235   \quark_if_recursion_tail_stop_do:nn{#5}{\c_true_bool}
236   \__erw_compare_recurse:nNNnnw
237   {#1} % <compare predicate name>
238   #2 % <convert>
239   #3 % <operator>
240   {#4} % <head>
241   #5 % <rest>
242   \q_recursion_stop
243 }
244 \cs_new:Npn
245 \__erw_compare_recurse:nNNnnw
246 #1 % <compare predicate name>
247 #2 % <convert>
248 #3 % <operator>
249 #4 % <first>
250 #5 % <second>
251 #6 % <rest>
252 \q_recursion_stop
253 {%
254   \bool_if:nTF
255   { \erw_compare_p:nNnNn{#1}#2{#4}#3{#5} }
256   {%
257     \__erw_compare_recurse:nNNnw
258     {#1} % <compare predicate name>
259     #2 % <convert>
260     #3 % <operator>
261     {#5} % <head>
262     #6 % <rest>
263     \q_recursion_stop
264   }
265   {\c_false_bool}
266 }
267 \erw_keys_set:n
268 {%
269   new_compare_p =
270   {erw_int_incr}
271   {nn}
272   {\exp_args:Ne
273     \int_compare_p:nNn{ \int_eval:n{#1+1} } = {#2} }
274 }

```

5 keyval

```

275 \cs_new:Npn\__erw_keyval_key:w #1 = #2 \q_recursion_stop{#1}
276 \cs_new:Npn\__erw_keyval_value:w #1 = #2 \q_recursion_stop{#2}
277 \cs_new:Npn \erw_keyval_key:n#1{\__erw_keyval_key:w #1 \q_recursion_stop}
278 \cs_new:Npn \erw_keyval_value:n#1{\__erw_keyval_value:w #1 \q_recursion_stop}

```

```

279 \cs_new:Npn \erw_keyval:nn#1#2{ #1 = #2 }
280 \erw_keys_set:n
281 {
282   new_compare_p = {erw_key_compare}
283   {nNn}{ \erw_compare_p:nNnNn
284     {int_compare_p}\erw_keyval_key:n{#1}#2{#3} },
285   new_compare_p = {erw_key_compare}
286   {n}{ \erw_compare_recurse_p:nnNN{#1}
287     {int_compare_p}\erw_keyval_key:n< }
288 }
289 \cs_new_protected:Npn
290 __erw_keyval_dispatch_build:nn
291 #1 % <|_protected>
292 #2 % <ext>
293 {
294   \use:c{cs_new#1:cpn}
295   {erw_keyval_dispatch#2:NNn}
296   ##1 % <unary>
297   ##2 % <binary>
298   ##3 % <keyval list>
299   { \use:c{__erw_keyval_dispatch#2:NNw}##1##2##3=\q_recursion_tail\q_recursion_stop }
300   \use:c{cs_new#1:cpn}
301   {__erw_keyval_dispatch#2:NNw}##1##2##3=##4\q_recursion_stop
302   { \quark_if_recursion_tail_stop_do:nn{##4}{##1{##3}}
303     \use:c{__erw_keyval_dispatch#2:Nw}##2##3=##4\q_recursion_stop }
304   \use:c{cs_new#1:cpn}
305   {__erw_keyval_dispatch#2:Nw}##1##2=##3=\q_recursion_tail\q_recursion_stop
306   {##1{##2}{##3}}
307 }
308 __erw_keyval_dispatch_build:nn{}{}
309 __erw_keyval_dispatch_build:nn{_protected}{_protected}

```

6 op's on list

```

310 \cs_new:Npn
311 \erw_remove_first:n
312 #1 % <tokenlist>
313 {\erw_remove_first_q:w#1\q_recursion_tail\q_recursion_stop}
314 \cs_generate_variant:Nn\erw_remove_first:n{e}
315 \cs_new:Npn
316 \erw_remove_last:n
317 #1 % <tokenlist>
318 {\erw_remove_last_q:w#1\q_recursion_tail\q_recursion_stop}
319 \cs_generate_variant:Nn\erw_remove_last:n{e}
320 \cs_new:Npn
321 \erw_first:n
322 #1
323 {\erw_first_q:w#1\q_recursion_tail\q_recursion_stop}
324 \cs_generate_variant:Nn\erw_first:n{e}
325 \cs_new:Npn
326 \erw_last:n
327 #1 % <tokenlist>
328 {\erw_last_q:w#1\q_recursion_tail\q_recursion_stop}
329 \cs_generate_variant:Nn\erw_last:n{e}

```

```

330 \cs_new:Npn
331 \erw_adjacent_insert:nn
332 #1 % <list>
333 #2 % <separator>
334 {%
335   \erw_first:n{#1}
336   \erw_swap:en
337   { \erw_remove_first:n{#1} }
338   {%
339     \__erw_adjacent_insert:nw
340     {#2} % <separator>
341   }
342   \q_recursion_tail
343   \q_recursion_stop
344 }
345 \cs_generate_variant:Nn\erw_adjacent_insert:nn{e}
346 \cs_new:Npn
347 \__erw_adjacent_insert:nw
348 #1 % <separator>
349 #2 % <rest>
350 \q_recursion_stop
351 {%
352   \quark_if_recursion_tail_stop:n{#2}
353   \__erw_adjacent_insert:new {#1}{\tl_if_head_is_group_p:n{#2}}#2 \q_recursion_stop
354 }
355 \cs_new:Npn
356 \__erw_adjacent_insert:nnw
357 #1 % <separator>
358 #2 % <head is group>
359 #3 % <head>
360 #4 % <rest>
361 \q_recursion_stop
362 {%
363   #1\bool_if:nTF{#2}{#{#3}}{#{3}}
364   \__erw_adjacent_insert:nw{#1}#4\q_recursion_stop
365 }
366 \cs_generate_variant:Nn\__erw_adjacent_insert:nnw{ne}
367 % ^^A% ^^A% ^^A <TRASH>
368 % ^^A\cs_new:Npn
369 % ^^A\__erw_thread:NNnw #1 #2 #3 #4 #5 / #6 #7 \q_recursion_stop
370 % ^^A{%
371 % ^^A \quark_if_recursion_tail_stop:n{#6}
372 % ^^A #1{#3}{#4}{#6}
373 % ^^A \__erw_thread:NNnw #1 #2 {#2{#3}} #5 / #7 \q_recursion_stop}
374 % ^^A\cs_new:Npn
375 % ^^A\erw_thread:NNnnn
376 % ^^A#1 % \<name_1>:nnn
377 % ^^A#2 % \<name_2>:n
378 % ^^A#3 % <position>
379 % ^^A#4 % <...{tl a_i}...>
380 % ^^A#5 % <...{tl b_i}...>
381 % ^^A{ \__erw_thread:NNnw #1#2{#3}#4 \q_recursion_tail / #5 \q_recursion_tail \q_recursion_stop}
382 % ^^A\cs_new:Npn
383 % ^^A\erw_thread_index:NNnn

```

```

384 % ^^A#1 % \<function name>:nnn
385 % ^^A#2 % <start index>
386 % ^^A#3 % <...{tl a_i}...>
387 % ^^A#4 % <...{tl b_i}...>
388 % ^^A{ \erw_thread:NNnn#1\int_incr:n{#2}{#3}{#4} }
389 % ^^A\cs_new:Npn
390 % ^^A\__erw_thread:Nnw #1 #2 #3 / #4 #5 \q_recursion_stop
391 % ^^A{\quark_if_recursion_tail_stop:n{#4} #1{#2}{#4}
392 % ^^A \__erw_thread:Nnw #1 #3 / #5 \q_recursion_stop}
393 % ^^A\cs_new:Npn
394 % ^^A\erw_thread:Nnn
395 % ^^A#1 % \<function name>:nn
396 % ^^A#2 % <...{tl a_i}...>
397 % ^^A#3 % <...{tl b_i}...>
398 % ^^A{\__erw_thread:Nnw #1 #2 \q_recursion_tail / #3 \q_recursion_tail \q_recursion_stop}
399 % ^^A\cs_new:Npn
400 % ^^A\__erw_map:NNnw #1 #2 #3 #4 #5 \q_recursion_stop
401 % ^^A{\quark_if_recursion_tail_stop:n{#4} #1{#3}{#4}
402 % ^^A \__erw_map:NNnw #1 #2 {#2{#3}} #5 \q_recursion_stop}
403 % ^^A\cs_new:Npn
404 % ^^A\erw_map:NNnn
405 % ^^A#1 % \<function name>:nn
406 % ^^A#2 % \<function name>:n
407 % ^^A#3 % <start index>
408 % ^^A#4 % <...{tl_i}...>
409 % ^^A{\__erw_map:NNnw #1 #2 {#3} #4 \q_recursion_tail \q_recursion_stop}
410 % ^^A\cs_new:Npn
411 % ^^A\erw_map_index:Nnn
412 % ^^A#1 % \<function name>:nn
413 % ^^A#2 % <start index>
414 % ^^A#3 % <...{tl_i}...>
415 % ^^A{ \erw_map:NNnn #1 \erw_int_incr:n {#2}{#3} }
416 % ^^A%^^A <TRASH>

417 \cs_new:Npn
418 \erw_clist_tl:nn
419 #1 % <bool>
420 #2 % <list>
421 { \erw_clist_tl:nnw {#1} #2 \q_recursion_tail\q_recursion_stop }
422 \cs_new:Npn
423 \erw_clist_tl:nnw #1 #2 \q_recursion_stop
424 {\quark_if_recursion_tail_stop:n{#2}
425 \erw_clist_tl:nenw {#1}
426 {\tl_if_head_is_group_p:n{#2}} #2 \q_recursion_stop}
427 \cs_generate_variant:Nn\erw_clist_tl:nnw{ne}
428 \cs_new:Npn
429 \erw_clist_tl:nnnw
430 #1 % <bool>
431 #2 % <head is group>
432 #3 % <head>
433 #4 % <rest>
434 \q_recursion_stop
435 {
436 \quark_if_recursion_tail_stop_do:nn{#4}
437 {%

```

```

438     \bool_if:nTF
439     {\bool_lazy_and_p:nn{#1}{#2}}
440     {{#3}}{#3}
441   }
442   \bool_if:nTF{\bool_lazy_and_p:nn{#1}{#2}}
443   {{#3}}{#3},
444   \erw_clist_tl:nnw {#1} #4 \q_recursion_stop
445 }
446 \cs_generate_variant:Nn\erw_clist_tl:nnnw{ne}
447 \prg_new_conditional:Npnn
448 \erw_if_in_clist:nn
449 #1 % <value>
450 #2 % <clist>
451 {p}
452 { \__erw_clist_if_in:nw {#1} #2, \q_recursion_tail \q_recursion_stop }
453 \cs_new:Npn
454 \__erw_clist_if_in:nw #1 #2 \q_recursion_stop
455 {%
456   \quark_if_recursion_tail_stop:n{#2}
457   \__erw_clist_if_in:nnw {#1} #2 \q_recursion_stop
458 }
459 \cs_new:Nn
460 \__erw_clist_if_in:nn
461 {\__erw_clist_if_in:nw{#1} #2 \q_recursion_stop}
462 \cs_new:Npn
463 \__erw_clist_if_in:nnw #1 #2, #3 \q_recursion_stop
464 {%
465   \quark_if_recursion_tail_stop_do:nn{#3}
466   {%
467     \str_if_eq:nnTF{#1}{#2}
468     {\prg_return_true:}{\prg_return_false:}
469   }
470   \str_if_eq:nnTF{#1}{#2}
471   {\prg_return_true:}
472   {\__erw_clist_if_in:nw {#1} #3 \q_recursion_stop}
473   \__erw_empty:w\q_recursion_stop
474 }

```

7 algo

7.1 split

```

475 \cs_new:Npn
476 \erw_split_even:n
477 #1 % <tokenlist>
478 {%
479   \tl_if_empty:nF{#1}
480   {%
481     \exp_last_unbraced:Ne
482     \__erw_split_even:nnnw
483     {%
484       {\__erw_split_even_threshold:n{#1}} % <count>
485       {\tl_if_head_is_group_p:n{#1}} % <head is group>
486     }

```

```

487     #1 % <tokenlist>
488     \q_recursion_tail
489     \q_recursion_stop
490 }
491 }
492 \cs_generate_variant:Nn\erw_split_even:n{e}
493 \cs_new:Npn
494   \__erw_split_even_threshold:n
495   #1 % <tokenlist>
496   {\exp_args:Ne
497     \int_div_round:nn{\tl_count:n{#1}}{2}}
498   \cs_new:Npn
499     \__erw_split_even:nnnw
500     #1 % <threshold>
501     #2 % <head is group>
502     #3 % <head>
503     #4 % <rest>
504     \q_recursion_stop
505     {%
506       \quark_if_recursion_tail_stop_do:nn{#4}
507       { { \bool_if:nTF{#2}{#{3}}{#3} }{ } }
508       \exp_last_unbraced:Ne
509       \__erw_split_even:nnnw
510       {%
511         {1} % <left size>
512         { \tl_if_head_is_group_p:n{#4} }
513         {#1} % <threshold count>
514         { \bool_if:nTF{#2}{#{3}}{#3} } % <left list>
515       }
516       #4 % <right list>
517       \q_recursion_stop
518     }
519     \cs_new:Npn
520       \__erw_split_even:nnnw
521       #1 % <left size>
522       #2 % <right head is group>
523       #3 % <threshold count>
524       #4 % <left list>
525       #5 % <right head>
526       #6 % <right rest>
527       \q_recursion_stop
528       {%
529         \bool_if:nTF
530         { \int_compare_p:nNn {#1}<{#3} }
531         {%
532           \exp_last_unbraced:Ne
533           \__erw_split_even:nnnw
534           {
535             { \int_eval:n{#1+1} } % <left size>
536             { \tl_if_head_is_group_p:n{#6} } % <right head is group>
537             {#3} % <threshold count>
538             {#4\bool_if:nTF{#2}{#{5}}{#5}} % <left list>
539           }
540           #6

```

```

541     \q_recursion_stop
542   }
543   {%
544     {#4}
545     {%
546       \bool_if:nTF{#2}{{#5}}{{#5}}
547       \erw_remove_last_q:w#6\q_recursion_stop\erw_last_q:w#6\q_recursion_stop}
548     }
549   }

```

7.2 thread sort

```

550 \cs_new:Npn
551 \erw_thread_sort:nnNn
552 #1 % <first sorted list>
553 #2 % <second sorted list>
554 #3 % <compare predicate name>
555 #4 % <compare operator>
556 {%
557   \__erw_thread_sort:nNnnn
558   {#3} % <compare predicate name>
559   #4 % <compare operator>
560   {\c_empty_tl} % <accum>
561   {#1}
562   {#2}
563 }
564 \cs_generate_variant:Nn\erw_thread_sort:nnNn{ee}
565 \cs_new:Npn
566 \__erw_thread_sort:nNnnn
567 #1 % <compare predicate name>
568 #2 % <compare operator>
569 #3 % <sorted>
570 #4 % <first>
571 #5 % <second>
572 {%
573   \__erw_thread_sort:nNnw
574   {#1} % <compare predicate name>
575   {#2} % <compare operator>
576   {#3} % <sorted>
577   #4 \q_recursion_tail% <first>
578   \q_stop
579   #5 \q_recursion_tail% <second>
580   \q_recursion_stop
581 }
582 \cs_generate_variant:Nn\__erw_thread_sort:nNnnn{nNnee}
583 \cs_new:Npn
584 \__erw_thread_sort:nNnw
585 #1 % <compare predicate name>
586 #2 % <compare operator>
587 #3 % <sorted>
588 #4 % <first>
589 \q_stop
590 #5 % <second>
591 \q_recursion_stop
592 {%

```

```

593 \quark_if_recursion_tail_stop_do:nn{#4}
594 { #3 \erw_all_q:w #5 \q_recursion_stop }
595 \quark_if_recursion_tail_stop_do:nn{#5}
596 { #3 \erw_all_q:w #4 \q_recursion_stop }
597 \__erw_thread_sort:nNneeww
598 {#1}#2{#3}
599 { \tl_if_head_is_group_p:n{#4} }
600 { \tl_if_head_is_group_p:n{#5} }
601 #4\q_stop
602 #5\q_recursion_stop
603 }
604 \cs_new:Npn
605 \__erw_thread_sort:nNnnnw
606 #1 % <compare predicate name>
607 #2 % <compare operator>
608 #3 % <sorted>
609 #4 % <head is begin>
610 #5 % <head is begin>
611 #6 % <first head>
612 #7 % <first rest>
613 \q_stop
614 #8 % <second head>
615 #9 % <second rest>
616 \q_recursion_stop
617 {%
618 \bool_if:nTF
619 { \use:c{#1:nNn}{#6}#2{#8} }
620 {%
621 \__erw_thread_sort:nNeee
622 {#1}
623 #2
624 {#3\bool_if:nTF{#4}{{#6}}{{#6}}}
625 {\erw_all_q:w#7\q_recursion_stop}
626 {\bool_if:nTF{#5}{{#8}}{{#8}}\erw_all_q:w#9\q_recursion_stop}
627 }
628 {%
629 \__erw_thread_sort:nNeee
630 {#1}
631 #2
632 {#3\bool_if:nTF{#5}{{#8}}{{#8}}}
633 {\bool_if:nTF{#4}{{#6}}{{#6}}\erw_all_q:w#7\q_recursion_stop}
634 {\erw_all_q:w#9\q_recursion_stop}
635 }
636 }
637 \cs_generate_variant:Nn\__erw_thread_sort:nNnnnw{nNnee}

```

7.3 merge sort

```

638 \cs_new:Npn
639 \erw_merge_sort:nN
640 #1 % <compare predicate name>
641 #2 % <compare operator>
642 #3 % <unsorted list>
643 {%
644 \tl_if_empty:nF{#3}

```



```

645  {%
646    \__erw_sort_merge:enNw
647    {\tl_if_head_is_group_p:n{#3}} % <head is group>
648    {#1} % <compare predicate name>
649    #2 % <compare operator>
650    #3 % <unsorted list>
651    \q_recursion_tail
652    \q_recursion_stop
653  }
654 }
655 \cs_generate_variant:Nn\erw_merge_sort:nNn{nNe}
656 \cs_new:Npn
657   \__erw_sort_merge:nnNw
658   #1 % <head is group>
659   #2 % <compare predicate name>
660   #3 % <compare operator>
661   #4 % <unsorted list head>
662   #5 % <unsorted list rest>
663   \q_recursion_stop
664   {%
665     \quark_if_recursion_tail_stop_do:nn{#5}
666     { \bool_if:nTF{#1}{{#4}}{#4} }
667     \exp_last_unbraced:Ne
668     \__erw_sort_merge:nnnN
669     {%
670       \erw_split_even:e
671       {%
672         \bool_if:nTF{#1}{{#4}}{#4}
673         \erw_all_q:w#5\q_recursion_stop
674       }
675       } % {<first sorted list>}{<second sorted list>}
676       {#2} % <compare predicate name>
677       #3 % <compare operator>
678       \__erw_empty:w \q_recursion_stop
679     }
680   \cs_generate_variant:Nn\__erw_sort_merge:nnNw{e}
681   \cs_new:Npn
682     \__erw_sort_merge:nnnN
683     #1 % <left unsorted list>
684     #2 % <right unsorted list>
685     #3 % <compare predicate name>
686     #4 % <compare operator>
687     {%
688       \erw_thread_sort:eeNn
689       {%
690         \__erw_sort_merge:enNw
691         {\tl_if_head_is_group_p:n{#1}}
692         {#3} % <compare predicate name>
693         #4 % <compare operator>
694         #1 % <unsorted list>
695         \q_recursion_tail
696         \q_recursion_stop
697       } % <first sorted list>
698     }

```

```

699 \__erw_sort_merge:enNw
700 {\tl_if_head_is_group_p:n{#2}}
701 {#3} % <compare predicate name>
702 #4 % <compare operator>
703 #2 % <unsorted list>
704 \q_recursion_tail
705 \q_recursion_stop
706 } % <second sorted list>
707 {#3} % <compare predicate name>
708 #4 % <operator>
709 }

```

7.4 filter

```

710 \msg_new:nnn{\__erw}{tokenlist-incr}
711 {expecting-an-ascending-tokenlist-got-#1-followed-by-#2}
712 \cs_new:Npn
713 \__erw_filter_uniq:nnw
714 #1 % <compare predicate>
715 #2 % <greatest>
716 #3 % <tokenlist>
717 \q_recursion_stop
718 { %
719   \quark_if_recursion_tail_stop:n{#3}
720   \__erw_filter_uniq_aux:nnw{#1}{#2}#3\q_recursion_stop}
721 \cs_new:Npn
722 \__erw_filter_uniq_aux:nw
723 #1 % <compare predicate>
724 #2 % <tokenlist head>
725 #3 % <tokenlist rest>
726 \q_recursion_stop
727 {%
728   {#2}
729   \__erw_filter_uniq:nnw
730   {#1} % <compare predicate>
731   {#2} #3 % <tokenlist>
732   \q_recursion_stop }
733 \cs_new:Npn
734 \__erw_filter_uniq_aux:nnw
735 #1 % <compare predicate>
736 #2 % <last>
737 #3 % <head token>
738 #4 % <rest token>
739 \q_recursion_stop
740 { %
741   \bool_if:nTF{\use:c{#1:nNn}{#3}<{#2}}
742   {\msg_error:nnnn{\__erw}{tokenlist-incr}{#2}{#3}}
743   {%
744     \bool_if:nF
745     {\use:c{#1:nNn}{#3}={#2}}
746     % ^^A    {{#3}}
747     {\tl_if_single_token:nTF{#3}{#3}{{#3}}}
748   }
749   \quark_if_recursion_tail_stop:n{#4}
750   % ^^A \__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }

```

```

751 \__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }
752 \cs_new:Npn
753 \__erw_filter_uniq:nw
754 #1 % <compare predicate>
755 #2 % <tokenlist>
756 {%
757   \quark_if_recursion_tail_stop_do:nn{#2}{\c_empty_tl}
758   \__erw_filter_uniq_aux:nw {#1}#2 \q_recursion_stop}
759 \cs_new:Npn
760 \erw_filter_uniq:nn
761 #1 % <compare predicate>
762 #2 % <tokenlist>
763 {%
764   \__erw_filter_uniq_aux:nw
765   {#1} % <compare predicate>
766   #2
767   \q_recursion_tail % <head token>
768   \q_recursion_stop}
769 \cs_new:Npn
770 \erw_filter_uniq:n
771 #1 % <ascending integers>
772 { \erw_filter_uniq:nn{int_compare_p}{#1} }
773 \cs_generate_variant:Nn\erw_filter_uniq:nn{ne}

```

8 code

```

774 \keys_define:nn{__erw}
775 { clist_map_inline.code:n = \__erw_map_inline_clist:nnn#1 }
776 \cs_new_protected:Npn
777 \__erw_map_inline_clist:nnn
778 #1 % <clist>
779 #2 % <signature>
780 #3 % <code>
781 {
782   \cs_new_protected:cn
783   {__erw_do:#2}{#3}
784   \clist_map_inline:nn
785   {#1}
786   {\use:c{__erw_do:#2}##1}
787 }
788 \cs_new:Npn
789 \erw_parameter:n
790 #1 % ^^A <arity>
791 {## #1}
792 \cs_new:Npn
793 \__erw_parameter_aux:nn
794 #1 % <finish>
795 #2 % <start>
796 { \int_step_function:nnN {#2}{#1}\erw_parameter:n}
797 \cs_new:Npn
798 \erw_parameter:nn
799 #1 % <start>
800 #2 % <count>
801 {%

```

```

802 \exp_args:Ne
803 \__erw_parameter_aux:nn
804 {\int_eval:n{#1+#2-1}}{#1}}
805 \cs_new:Npn
806 \erw_argument:nn
807 #1 % <position>
808 #2 % <signature>
809 {\__erw_argument:nw{#1}#2\q_recursion_tail\q_recursion_stop}
810 \cs_new:Npn
811 \__erw_argument_unit:nn
812 #1 % <position>
813 #2 % <n|N>
814 {\use:c{\__erw_argument_#2:w} #1 \q_recursion_stop}
815 \cs_new:Npn\__erw_argument_n:w #1 \q_recursion_stop{## #1}}
816 \cs_new:Npn\__erw_argument_N:w #1 \q_recursion_stop{## #1}
817 \cs_new:Npn
818 \__erw_argument:nw
819 #1 % <position>
820 #2 % <signature list>
821 \q_recursion_stop
822 { \quark_if_recursion_tail_stop:n{#2}
823   \__erw_argument:nnw{#1}#2\q_recursion_stop }
824 \cs_new:Npn
825 \__erw_argument:nnw
826 #1 % <position>
827 #2 % <n|N>
828 #3 % <signature rest>
829 \q_recursion_stop
830 {%
831   \__erw_argument_unit:nn{#1}{#2}
832   \exp_args:Ne
833   \__erw_argument:nw
834   {\erw_int_incr:n{#1}}#3\q_recursion_stop }
835 \makeatletter
836 % ^A https://tex.stackexchange.com/a/614151/112708
837 \def\__erw_make_group_token_const#1
838 {\__erw_make_group_token_constAux#1\@gobbletwo{}}
839 \def\__erw_make_group_token_constAux#1#
840 {#1\__erw_make_group_token_constAuxii}
841 \def\__erw_make_group_token_constAuxii#1
842 {\c_group_begin_token
843   \__erw_make_group_token_const{#1}
844   \c_group_end_token
845   \__erw_make_group_token_constAuxi}
846 \makeatother
847 \cs_new:Npn
848 \__erw_code_analyze:nw
849 #1 % <accum>
850 #2 % <first token>
851 #3 % <second token>
852 #4 % <third token>
853 #5 % <rest>
854 \q_recursion_stop
855 {%

```

```

856 \quark_if_recursion_tail_stop_do:nn{#3}{#1}
857 \token_if_parameter:NTF#2
858 {%
859   \token_if_group_begin:NTF#4
860   {%
861     \__erw_code_analyze:nn
862     {#1{#3=N}}
863     {#5}
864   }
865   {%
866     \__erw_code_analyze:nn
867     {#1{#3=n}}
868     {#4#5}
869   }
870 }
871 {%
872   \__erw_code_analyze:nn{#1}{#3#4#5}
873 }
874 \__erw_empty:w \q_recursion_stop
875 }
876 \cs_new:Npn
877 \__erw_code_analyze:nn
878 #1 % <accum>
879 #2 % <first token>
880 % ^^A <second token>
881 % ^^A <third token>
882 % ^^A <rest>
883 {%
884   \__erw_code_analyze:nw
885   {#1} % <accum>
886   #2 % <first token>
887   % <second token>
888   % <third token>
889   % <rest>
890   \q_recursion_stop
891 }
892 \cs_new:Npn
893 \__erw_code_analyze_i:n
894 #1 % <code-tokenlist>
895 {%
896   \exp_args:Ne
897   \__erw_code_analyze:nn{\c_empty_tl}
898   {#1
899     \q_recursion_tail
900     \q_recursion_tail
901     \q_recursion_tail
902   }
903 }
904 \cs_new:Npn
905 \__erw_code_analyze:n
906 #1 % ^^A <code>
907 {%
908   \exp_args:Ne
909   \__erw_code_analyze_i:n

```

```

910   { \__erw_make_group_token_const{#1} }
911 }
912 \cs_generate_variant:Nn\__erw_code_analyze:n{e}
913 \cs_new:Npn
914 \erw_code_analyze:n
915 #1 % ^^A <code>
916 {%
917   \erw_filter_uniq:ne
918   {erw_key_compare_p}
919   {%
920     \erw_merge_sort:nNe
921     {erw_key_compare_p}<
922     { \__erw_code_analyze:n{#1} }
923   }
924 }
925 \msg_new:nnn{\__erw}
926 {code-adjacent-parameters}
927 {code-has-non-adjacent-parameters;~#1}
928 \cs_new:Npn
929 \__erw_signature:n
930 #1 % <position signature>
931 {%
932   \bool_if:nTF
933   { \erw_compare_recurse_p:nnN{#1}{erw_key_compare_p}< }
934   { \tl_map_function:nnN{#1}\erw_keyval_value:n }
935   {%
936     \msg_error:nnn{\__erw}
937     {code-adjacent-parameters}{#1}
938   }
939 }
940 \cs_generate_variant:Nn\__erw_signature:n{e}
941 \cs_new:Npn
942 \erw_signature:n
943 #1 % <position signature>
944 {%
945   \__erw_signature:e
946   { \erw_code_analyze:n{#1} }
947 }
948 \ProcessKeysOptions{\__erw}
949 \ExplSyntaxOff
950 </package>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols

\< 376, 377, 384, 395, 405, 406, 412

B

bool commands:

\bool_if:nTF 65,
88, 110, 144, 157, 254, 363, 438, 442,
507, 514, 529, 538, 546, 618, 624,
626, 632, 633, 666, 672, 741, 744, 932
\bool_lazy_and_p:nn 439, 442
\c_false_bool 265
\c_true_bool 223, 235

C

clist commands:

\clist_map_inline:nn 784

cs commands:

\cs_generate_variant:Nn 4,
5, 6, 7, 8, 9, 10, 11, 12, 19, 23,
67, 79, 91, 102, 113, 181, 314, 319,
324, 329, 345, 366, 427, 446, 492,
564, 582, 637, 655, 680, 773, 912, 940
\cs_if_exist:Nn 15
\cs_new:Nn 27, 459
\cs_new:Npn 14,
16, 20, 21, 22, 24, 30, 38, 44, 51, 58,
68, 73, 80, 92, 96, 103, 124, 137, 149,
174, 189, 208, 215, 226, 244, 275,
276, 277, 278, 279, 310, 315, 320,
325, 330, 346, 355, 368, 374, 382,
389, 393, 399, 403, 410, 417, 422,
428, 453, 462, 475, 493, 498, 519,
550, 565, 583, 604, 638, 656, 681,
712, 721, 733, 752, 759, 769, 788,
792, 797, 805, 810, 815, 816, 817,
824, 847, 876, 892, 904, 913, 928, 941
\cs_new_protected:Nn 17, 18, 782
\cs_new_protected:Npn 289, 776
\cs_split_function:N 26

D

\def 837, 839, 841

E

erw commands:

\erw_adjacent_insert:nn 331, 345
\erw_all_q:w
. 31, 594, 596, 625, 626, 633, 634, 673
\erw_and_tl:nn 117

\erw_apply:Nw 19
\erw_argument:nn 806
\erw_clist_tl:nn 418
\erw_clist_tl:nnnw 425, 429, 446
\erw_clist_tl:nnw .. 421, 423, 427, 444
\erw_code_analyze:n 914, 946
\erw_compare_p:nNnNn 255, 283
\erw_compare_recurse_p:nnN 933
\erw_compare_recurse_p:nnNN 195, 286
\erw_filter_uniq:n 770
\erw_filter_uniq:nn 760, 772, 773, 917
\erw_first:n 321, 324, 335
\erw_first_q:w 52, 323
\erw_identity:n 20, 198
\erw_if_in_clist:nn 448
\erw_int_incr:n 21, 415, 834
\erw_keys_set:n
..... 17, 167, 182, 201, 267, 280
\erw_keys_set:nn 18
\erw_keyval:nn 279
\erw_keyval_key:n 277, 284, 287
\erw_keyval_value:n 278, 934
\erw_last:n 326, 329
\erw_last_q:w 36, 50, 93, 328, 547
\erw_map:NNnn 404, 415
\erw_map_index:Nnn 411
\erw_merge_sort:nNn ... 639, 655, 920
\erw_name_signature_cs:N 24
\erw_parameter:n 789, 796
\erw_parameter:nn 798
\erw_remove_first:n ... 311, 314, 337
\erw_remove_first_q:w 39, 313
\erw_remove_last:n 316, 319
\erw_remove_last_q:w 35, 49, 69, 318, 547
\erw_signature:n 942
\erw_split_even:n 476, 492, 670
\erw_swap:nn 22, 23, 336
\erw_thread:Nnn 394
\erw_thread:NNnnn 375, 388
\erw_thread_index:Nnnn 383
\erw_thread_sort:nnNn . 551, 564, 688

erw internal commands:

__erw_adjacent_insert:nnw
..... 353, 356, 366
__erw_adjacent_insert:nw
..... 339, 347, 364
__erw_and_tl:nnw 132, 138
__erw_and_tl:nw 122, 125, 146

prop commands:		R	
\prop_put:Nnn	12	regex commands:	
		\regex_gset:Nn	8
		\regex_log:N	9
		\regex_match:NnTF	10
Q		S	
quark commands:		str commands:	
\quark_if_recursion_tail_stop:n		\str_if_eq:nnTF	467, 470
	42, 56, 71, 87, 94, 352,		
	371, 391, 401, 424, 456, 719, 749, 822		
\quark_if_recursion_tail_stop_-		T	
do:nn	110, 130, 223, 235, 302,	TeX and L ^A T _E X 2 _ε commands:	
	436, 465, 506, 593, 595, 665, 757, 856	\@gobbletwo	838
\q_recursion_stop	16, 33, 35, 36, 41,	\text	15
	43, 48, 49, 50, 54, 57, 63, 69, 72, 77,	tl commands:	
	78, 85, 89, 93, 95, 100, 101, 108, 111,	\c_empty_tl	16, 560, 757, 897
	122, 128, 135, 142, 146, 214, 221,	\tl_count:n	497
	224, 233, 242, 252, 263, 275, 276,	\tl_if_empty:nTF	479, 644
	277, 278, 299, 301, 303, 305, 313,	\tl_if_head_is_group_p:n	
	318, 323, 328, 343, 350, 353, 361,		57, 72, 89, 95, 111, 353, 426,
	364, 369, 373, 381, 390, 392, 398,		485, 512, 536, 599, 600, 647, 691, 700
	400, 402, 409, 421, 423, 426, 434,	\tl_if_single_token:nTF	747
	444, 452, 454, 457, 461, 463, 472,	\tl_map_function:nN	934
	473, 489, 504, 517, 527, 541, 547,	\tl_to_str:n	11
	580, 591, 594, 596, 602, 616, 625,	token commands:	
	626, 633, 634, 652, 663, 673, 678,	\token_if_group_begin:NnTF	859
	696, 705, 717, 720, 726, 732, 739,	\token_if_parameter:NnTF	857
	750, 751, 758, 768, 809, 814, 815,		
	816, 821, 823, 829, 834, 854, 874, 890	U	
\q_recursion_tail	122, 214, 299,	use commands:	
	305, 313, 318, 323, 328, 342, 381,	\use:N	294, 299,
	398, 409, 421, 452, 488, 577, 579,		300, 303, 304, 619, 741, 745, 786, 814
	651, 695, 704, 767, 809, 899, 900, 901		
\q_stop	578, 589, 601, 613		