

The `erw-l3` package*

January 28, 2022

Abstract

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

Contents

I	Usage	2
1	<code>boilerplate</code>	2
2	<code>quark</code>	2
3	<code>predicate</code>	2
4	<code>op's on lists</code>	3
5	<code>algo</code>	3
6	<code>code</code>	3
II	Other	3
1	Bibliography	3
2	Support	3
III	Implementation	3
1	<code>kernel</code>	4
2	<code>boilerplate</code>	4
3	<code>quark</code>	4
4	<code>predicate</code>	6

*This file describes version v4.2, last revised 2022-01-28.

5	keyval	7
6	op's on list	8
7	algo	10
	7.1 split	10
	7.2 thread sort	12
	7.3 merge sort	13
	7.4 filter	15
8	code	16

Part I

Usage

1 boilerplate

```
\erw_keys_set:n \erw_keys_set:n{<keyval list>}
\erw_keys_set:nn
```

```
\erw_identity:n *
\erw_int_incr:n *
\erw_swap:nn *
\erw_swap:ne *
\erw_name_signature_cs:N *
```

2 quark

```
\erw_all_q:w \erw_remove_last_q:w<tokenlist> \q_recursion_tail\q_recursion_stop
\erw_remove_first_q:w
\erw_first_q:w
\erw_remove_last_q:w
\erw_last_q:w
```

3 predicate

```
new_compare_p \erw_keys_set:n{ new_compare_p = {<name>}{<signature>}{<predicate>} }
Instance
erw_compare_p:nNnNn
erw_int_incr_p:nn
```

4 op's on lists

```
\erw_remove_first:n  
\erw_remove_last:n  
\erw_first:n  
\erw_last:n  
\erw_adjacent_insert:nn  
\erw_adjacent_insert:en
```

5 algo

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

6 code

```
\erw_parameter:n      \erw_parameter:n{<arity>}  
\erw_parameter:nn     \erw_parameter:nn{<start pos>}{<arity>}  
\argument:nn          \erw_argument:nn{<start pos>}{<signature>}
```

Part II

Other

1 Bibliography

- [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-13>.

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{\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:nnw{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_int_incr}
186 {nn}
187 {\exp_args:Ne
188 \int_compare_p:nNn{ \int_eval:n{#1+1} } = {#2} }
189 }

```

5 keyval

```

190 \cs_new:Npn\__erw_keyval_key:w #1 = #2 \q_recursion_stop{#1}
191 \cs_new:Npn\__erw_keyval_value:w #1 = #2 \q_recursion_stop{#2}
192 \cs_new:Npn \erw_keyval_key:n#1{\__erw_keyval_key:w #1 \q_recursion_stop}
193 \cs_new:Npn \erw_keyval_value:n#1{\__erw_keyval_value:w #1 \q_recursion_stop}
194 \cs_new:Npn \erw_keyval:nn#1#2{ #1 = #2 }
195 \erw_keys_set:n
196 {
197 new_compare_p = {erw_key_compare}
198 {nNn}{ \erw_compare_p:nNnNn
199 {int_compare_p}\erw_keyval_key:n{#1}#2{#3} },
200 new_compare_p = {erw_key_compare}
201 {n}{ \erw_compare_recurse_p:nnNN{#1}

```

```

202     {int_compare_p}\erw_keyval_key:n< }
203 }

```

6 op's on list

```

204 \cs_new:Npn
205 \erw_remove_first:n
206 #1 % <tokenlist>
207 {\erw_remove_first_q:w#1\q_recursion_tail\q_recursion_stop}
208 \cs_generate_variant:Nn\erw_remove_first:n{e}
209 \cs_new:Npn
210 \erw_remove_last:n
211 #1 % <tokenlist>
212 {\erw_remove_last_q:w#1\q_recursion_tail\q_recursion_stop}
213 \cs_generate_variant:Nn\erw_remove_last:n{e}
214 \cs_new:Npn
215 \erw_first:n
216 #1
217 {\erw_first_q:w#1\q_recursion_tail\q_recursion_stop}
218 \cs_generate_variant:Nn\erw_first:n{e}
219 \cs_new:Npn
220 \erw_last:n
221 #1 % <tokenlist>
222 {\erw_last_q:w#1\q_recursion_tail\q_recursion_stop}
223 \cs_generate_variant:Nn\erw_last:n{e}
224 \cs_new:Npn
225 \erw_adjacent_insert:nn
226 #1 % <list>
227 #2 % <separator>
228 {%
229   \erw_first:n{#1}
230   \erw_swap:en
231   { \erw_remove_first:n{#1} }
232   {%
233     \__erw_adjacent_insert:nw
234     {#2} % <separator>
235   }
236   \q_recursion_tail
237   \q_recursion_stop
238 }
239 \cs_generate_variant:Nn\erw_adjacent_insert:nn{e}
240 \cs_new:Npn
241 \__erw_adjacent_insert:nw
242 #1 % <separator>
243 #2 % <rest>
244 \q_recursion_stop
245 {%
246   \quark_if_recursion_tail_stop:n{#2}
247   \__erw_adjacent_insert:new {#1}{\tl_if_head_is_group_p:n{#2}}#2 \q_recursion_stop
248 }
249 \cs_new:Npn
250 \__erw_adjacent_insert:nw
251 #1 % <separator>
252 #2 % <head is group>

```



```

253 #3 % <head>
254 #4 % <rest>
255 \q_recursion_stop
256 {%
257   #1\bool_if:nTF{#2}{{#3}}{#3}
258   \__erw_adjacent_insert:nw{#1}#4\q_recursion_stop
259 }
260 \cs_generate_variant:Nn\__erw_adjacent_insert:nnw{ne}

261 \cs_new:Npn
262 \erw_clist_tl:nn
263 #1 % <bool>
264 #2 % <list>
265 { \erw_clist_tl:nnw {#1} #2 \q_recursion_tail\q_recursion_stop }
266 \cs_new:Npn
267 \erw_clist_tl:nnw #1 #2\q_recursion_stop
268 {\quark_if_recursion_tail_stop:n{#2}
269  \erw_clist_tl:nenw {#1}
270  {\tl_if_head_is_group_p:n{#2}} #2 \q_recursion_stop}
271 \cs_generate_variant:Nn\erw_clist_tl:nnw{ne}
272 \cs_new:Npn
273 \erw_clist_tl:nnnw
274 #1 % <bool>
275 #2 % <head is group>
276 #3 % <head>
277 #4 % <rest>
278 \q_recursion_stop
279 {
280   \quark_if_recursion_tail_stop_do:nn{#4}
281   {%
282     \bool_if:nTF
283     {\bool_lazy_and_p:nn{#1}{#2}}
284     {{#3}}{#3}
285   }
286   \bool_if:nTF{\bool_lazy_and_p:nn{#1}{#2}}
287   {{#3}}{#3},
288   \erw_clist_tl:nnw {#1} #4 \q_recursion_stop
289 }
290 \cs_generate_variant:Nn\erw_clist_tl:nnnw{ne}
291 \prg_new_conditional:Npnn
292 \erw_if_in_clist:nn
293 #1 % <value>
294 #2 % <clist>
295 {p}
296 { \__erw_clist_if_in:nw {#1} #2, \q_recursion_tail \q_recursion_stop }
297 \cs_new:Npn
298 \__erw_clist_if_in:nw #1 #2 \q_recursion_stop
299 {%
300   \quark_if_recursion_tail_stop:n{#2}
301   \__erw_clist_if_in:nnw {#1} #2 \q_recursion_stop
302 }
303 \cs_new:Nn
304 \__erw_clist_if_in:nn
305 {\__erw_clist_if_in:nw{#1} #2 \q_recursion_stop}
306 \cs_new:Npn

```

```

307 \__erw_clist_if_in:nnw #1 #2, #3 \q_recursion_stop
308 {%
309   \quark_if_recursion_tail_stop_do:nn{#3}
310   {%
311     \str_if_eq:nnTF{#1}{#2}
312     {\prg_return_true:}{\prg_return_false:}
313   }
314   \str_if_eq:nnTF{#1}{#2}
315   {\prg_return_true:}
316   {\__erw_clist_if_in:nw {#1} #3 \q_recursion_stop}
317   \__erw_empty:w\q_recursion_stop
318 }

```

7 algo

7.1 split

```

319 \cs_new:Npn
320 \erw_split_even:n
321 #1 % <tokenlist>
322 {%
323   \tl_if_empty:nF{#1}
324   {%
325     \exp_last_unbraced:Ne
326     \__erw_split_even:nnnw
327     {%
328       {\__erw_split_even_threshold:n{#1}} % <count>
329       {\tl_if_head_is_group_p:n{#1}} % <head is group>
330     }
331     #1 % <tokenlist>
332     \q_recursion_tail
333     \q_recursion_stop
334   }
335 }
336 \cs_generate_variant:Nn\erw_split_even:n{e}
337 \cs_new:Npn
338 \__erw_split_even_threshold:n
339 #1 % <tokenlist>
340 {\exp_args:Ne
341   \int_div_round:nn{\tl_count:n{#1}}{2}}
342 \cs_new:Npn
343 \__erw_split_even:nnnw
344 #1 % <threshold>
345 #2 % <head is group>
346 #3 % <head>
347 #4 % <rest>
348 \q_recursion_stop
349 {%
350   \quark_if_recursion_tail_stop_do:nn{#4}
351   { { \bool_if:nTF{#2}{#3}{#3} }{ } }
352   \exp_last_unbraced:Ne
353   \__erw_split_even:nnnw
354   {%
355     {1} % <left size>

```

```

356     { \tl_if_head_is_group_p:n{#4} }
357     {#1} % <threshold count>
358     { \bool_if:nTF{#2}{#{#3}}{#3} } % <left list>
359   }
360   #4 % <right list>
361   \q_recursion_stop
362 }
363 \cs_new:Npn
364   \__erw_split_even:nnnw
365   #1 % <left size>
366   #2 % <right head is group>
367   #3 % <threshold count>
368   #4 % <left list>
369   #5 % <right head>
370   #6 % <right rest>
371   \q_recursion_stop
372   {%
373     \bool_if:nTF
374     { \int_compare_p:nNn {#1}<{#3} }
375     {%
376       \exp_last_unbraced:Ne
377       \__erw_split_even:nnnw
378       {
379         { \int_eval:n{#1+1} } % <left size>
380         { \tl_if_head_is_group_p:n{#6} } % <right head is group>
381         {#3} % <threshold count>
382         {#4\bool_if:nTF{#2}{#{#5}}{#5}} % <left list>
383       }
384       #6
385       \q_recursion_stop
386     }
387     {%
388       {#4}
389       {%
390         \bool_if:nTF{#2}{#{#5}}{#5}
391         \erw_remove_last_q:w#6\q_recursion_stop\erw_last_q:w#6\q_recursion_stop}
392     }
393   }

```

7.2 thread sort

```

394 \cs_new:Npn
395   \erw_thread_sort:nnNn
396   #1 % <first sorted list>
397   #2 % <second sorted list>
398   #3 % <compare predicate name>
399   #4 % <compare operator>
400   {%
401     \__erw_thread_sort:nNnnn
402     {#3} % <compare predicate name>
403     #4 % <compare operator>
404     {c_empty_tl} % <accum>
405     {#1}
406     {#2}
407   }

```

```

408 \cs_generate_variant:Nn\erw_thread_sort:nNn{ee}
409 \cs_new:Npn
410 \__erw_thread_sort:nNnnn
411 #1 % <compare predicate name>
412 #2 % <compare operator>
413 #3 % <sorted>
414 #4 % <first>
415 #5 % <second>
416 {%
417 \__erw_thread_sort:nNnw
418 {#1} % <compare predicate name>
419 {#2} % <compare operator>
420 {#3} % <sorted>
421 #4 \q_recursion_tail% <first>
422 \q_stop
423 #5 \q_recursion_tail% <second>
424 \q_recursion_stop
425 }
426 \cs_generate_variant:Nn\__erw_thread_sort:nNnnn{nNnee}
427 \cs_new:Npn
428 \__erw_thread_sort:nNnw
429 #1 % <compare predicate name>
430 #2 % <compare operator>
431 #3 % <sorted>
432 #4 % <first>
433 \q_stop
434 #5 % <second>
435 \q_recursion_stop
436 {%
437 \quark_if_recursion_tail_stop_do:n{#4}
438 { #3 \erw_all_q:w #5 \q_recursion_stop }
439 \quark_if_recursion_tail_stop_do:n{#5}
440 { #3 \erw_all_q:w #4 \q_recursion_stop }
441 \__erw_thread_sort:nNneeww
442 {#1}#2{#3}
443 { \tl_if_head_is_group_p:n{#4} }
444 { \tl_if_head_is_group_p:n{#5} }
445 #4\q_stop
446 #5\q_recursion_stop
447 }
448 \cs_new:Npn
449 \__erw_thread_sort:nNnnnw
450 #1 % <compare predicate name>
451 #2 % <compare operator>
452 #3 % <sorted>
453 #4 % <head is begin>
454 #5 % <head is begin>
455 #6 % <first head>
456 #7 % <first rest>
457 \q_stop
458 #8 % <second head>
459 #9 % <second rest>
460 \q_recursion_stop
461 {%

```

```

462 \bool_if:nTF
463 { \use:c{#1:nNn}{#6}#2{#8} }
464 {%
465   \__erw_thread_sort:nNeee
466   {#1}
467   #2
468   {#3\bool_if:nTF{#4}{#6}{#6}}
469   {\erw_all_q:w#7\q_recursion_stop}
470   {\bool_if:nTF{#5}{#8}{#8}\erw_all_q:w#9\q_recursion_stop}
471 }
472 {%
473   \__erw_thread_sort:nNeee
474   {#1}
475   #2
476   {#3\bool_if:nTF{#5}{#8}{#8}}
477   {\bool_if:nTF{#4}{#6}{#6}\erw_all_q:w#7\q_recursion_stop}
478   {\erw_all_q:w#9\q_recursion_stop}
479 }
480 }
481 \cs_generate_variant:Nn\__erw_thread_sort:nNnnnw{nNnee}

```

7.3 merge sort

```

482 \cs_new:Npn
483 \erw_merge_sort:nNn
484 #1 % <compare predicate name>
485 #2 % <compare operator>
486 #3 % <unsorted list>
487 {%
488   \tl_if_empty:nF{#3}
489   {%
490     \__erw_sort_merge:enNw
491     {\tl_if_head_is_group_p:n{#3}} % <head is group>
492     {#1} % <compare predicate name>
493     #2 % <compare operator>
494     #3 % <unsorted list>
495     \q_recursion_tail
496     \q_recursion_stop
497   }
498 }
499 \cs_generate_variant:Nn\erw_merge_sort:nNn{nNe}
500 \cs_new:Npn
501 \__erw_sort_merge:nnNw
502 #1 % <head is group>
503 #2 % <compare predicate name>
504 #3 % <compare operator>
505 #4 % <unsorted list head>
506 #5 % <unsorted list rest>
507 \q_recursion_stop
508 {%
509   \quark_if_recursion_tail_stop_do:nn{#5}
510   { \bool_if:nTF{#1}{#4}{#4} }
511   \exp_last_unbraced:Ne
512   \__erw_sort_merge:nnnN
513   {%

```

```

514 \erw_split_even:e
515 {%
516 \bool_if:nTF{#1}{#{4}}{#4}
517 \erw_all_q:w#5\q_recursion_stop
518 }
519 } % {<first sorted list>}{<second sorted list>}
520 {#2} % <compare predicate name>
521 #3 % <compare operator>
522 \__erw_empty:w \q_recursion_stop
523 }
524 \cs_generate_variant:Nn\__erw_sort_merge:nnNw{e}
525 \cs_new:Npn
526 \__erw_sort_merge:nnnN
527 #1 % <left unsorted list>
528 #2 % <right unsorted list>
529 #3 % <compare predicate name>
530 #4 % <compare operator>
531 {%
532 \erw_thread_sort:eeNn
533 {%
534 \__erw_sort_merge:enNw
535 {\tl_if_head_is_group_p:n{#1}}
536 {#3} % <compare predicate name>
537 #4 % <compare operator>
538 #1 % <unsorted list>
539 \q_recursion_tail
540 \q_recursion_stop
541 } % <first sorted list>
542 {%
543 \__erw_sort_merge:enNw
544 {\tl_if_head_is_group_p:n{#2}}
545 {#3} % <compare predicate name>
546 #4 % <compare operator>
547 #2 % <unsorted list>
548 \q_recursion_tail
549 \q_recursion_stop
550 } % <second sorted list>
551 {#3} % <compare predicate name>
552 #4 % <operator>
553 }

```

7.4 filter

```

554 \msg_new:nnn{\__erw}{tokenlist-incr}
555 {expecting-an-ascending-tokenlist-got-#1-followed-by-#2}
556 \cs_new:Npn
557 \__erw_filter_uniq:nnw
558 #1 % <compare predicate>
559 #2 % <greatest>
560 #3 % <tokenlist>
561 \q_recursion_stop
562 { %
563 \quark_if_recursion_tail_stop:n{#3}
564 \__erw_filter_uniq_aux:nnw{#1}{#2}#3\q_recursion_stop}
565 \cs_new:Npn

```

```

566 \__erw_filter_uniq_aux:nw
567 #1 % <compare predicate>
568 #2 % <tokenlist head>
569 #3 % <tokenlist rest>
570 \q_recursion_stop
571 {%
572   {#2}
573   \__erw_filter_uniq:nw
574   {#1} % <compare predicate>
575   {#2} #3 % <tokenlist>
576   \q_recursion_stop }
577 \cs_new:Npn
578 \__erw_filter_uniq_aux:nw
579 #1 % <compare predicate>
580 #2 % <last>
581 #3 % <head token>
582 #4 % <rest token>
583 \q_recursion_stop
584 { %
585   \bool_if:nTF{\use:c{#1:nNn}{#3}<{#2}}
586   {\msg_error:nnnn{\__erw}{tokenlist-incr}{#2}{#3}}
587   {%
588     \bool_if:nF
589     {\use:c{#1:nNn}{#3}={#2}}
590     % ^^A   {#3}}
591   {\tl_if_single_token:nTF{#3}{#3}{#3}}
592   }
593   \quark_if_recursion_tail_stop:n{#4}
594   % ^^A \__erw_filter_uniq:nw{#1}{#3}#4\q_recursion_stop }
595   \__erw_filter_uniq:nw{#1}{#3}#4\q_recursion_stop }
596   \cs_new:Npn
597   \__erw_filter_uniq:nw
598   #1 % <compare predicate>
599   #2 % <tokenlist>
600   {%
601     \quark_if_recursion_tail_stop_do:nn{#2}{\c_empty_tl}
602     \__erw_filter_uniq_aux:nw {#1}#2 \q_recursion_stop}
603   \cs_new:Npn
604   \erw_filter_uniq:nn
605   #1 % <compare predicate>
606   #2 % <tokenlist>
607   {%
608     \__erw_filter_uniq_aux:nw
609     {#1} % <compare predicate>
610     #2
611     \q_recursion_tail % <head token>
612     \q_recursion_stop}
613   \cs_new:Npn
614   \erw_filter_uniq:n
615   #1 % <ascending integers>
616   { \erw_filter_uniq:nn{int_compare_p}{#1} }
617   \cs_generate_variant:Nn\erw_filter_uniq:nn{ne}

```

8 code

```
618 \keys_define:nn{__erw}
619 { clist_map_inline.code:n = \__erw_map_inline_clist:nnn#1 }
620 \cs_new_protected:Npn
621 \__erw_map_inline_clist:nnn
622 #1 % <clist>
623 #2 % <signature>
624 #3 % <code>
625 {
626   \cs_new_protected:cn
627   {__erw_do:#2}{#3}
628   \clist_map_inline:nn
629   {#1}
630   {\use:c{__erw_do:#2}##1}
631 }
632 \cs_new:Npn
633 \erw_parameter:n
634 #1 %^^A <arity>
635 {## #1}
636 \cs_new:Npn
637 \__erw_parameter_aux:nn
638 #1 % <finish>
639 #2 % <start>
640 { \int_step_function:nnN {#2}{#1}\erw_parameter:n}
641 \cs_new:Npn
642 \erw_parameter:nn
643 #1 % <start>
644 #2 % <count>
645 {%
646   \exp_args:Ne
647   \__erw_parameter_aux:nn
648   {\int_eval:n{#1+#2-1}}{#1}}
649 \cs_new:Npn
650 \erw_argument:nn
651 #1 % <position>
652 #2 % <signature>
653 {\__erw_argument:nw{#1}#2\q_recursion_tail\q_recursion_stop}
654 \cs_new:Npn
655 \__erw_argument_unit:nn
656 #1 % <position>
657 #2 % <n|N>
658 {\use:c{__erw_argument_#2:w} #1 \q_recursion_stop}
659 \cs_new:Npn\__erw_argument_n:w #1 \q_recursion_stop{## #1}}
660 \cs_new:Npn\__erw_argument_N:w #1 \q_recursion_stop{## #1}
661 \cs_new:Npn
662 \__erw_argument:nw
663 #1 % <position>
664 #2 % <signature list>
665 \q_recursion_stop
666 { \quark_if_recursion_tail_stop:n{#2}
667   \__erw_argument:nnw{#1}#2\q_recursion_stop }
668 \cs_new:Npn
669 \__erw_argument:nnw
```



```
670 #1 % <position>
671 #2 % <n|N>
672 #3 % <signature rest>
673 \q_recursion_stop
674 {%
675   \_erw_argument_unit:nn{#1}{#2}
676   \exp_args:Ne
677   \_erw_argument:nw
678   {\erw_int_incr:n{#1}}#3\q_recursion_stop }
679 \ProcessKeysOptions{\_erw}
680 \ExplSyntaxOff
681 \endpackage
```