A potential third method for testing if an elt w is FC.

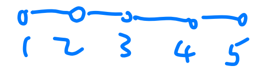
(no reduced-word-graph( ) or reduced_words( ) needed)

- Input: a reduced word w; Output: a list of all red words of w if w is FC, "False" if not.
- "Elementary" steps to be repeated:

  apply one commutation relation to a current word x to return a new word y; check y for long braids, stopping if there is one and moving on otherwise

For efficiency, if at some point we used a 'current' word $x = 2\underline{31}4$ to create the 'new' word $y = 2\underline{13}4$, we should mark the position where the commutation happened, in order to avoid re-generating x.

— Initially, set the input w to be the 'current' word x.

— There's a choice as to how to organize/order the generation of new words. We use examples on  to illustrate:

Method 1. : "Breadth first" Input: 53145

For each given word x, apply all possible commutations on x first, without doing commutations on the results.

red: marking starting position of the braid

$$x = 53145 \quad \rightarrow \text{first stage}$$

$$\overset{0}{\swarrow} \qquad \overset{1}{\downarrow} \qquad \overset{2}{\searrow}$$

$$y_1 = \underline{35}145 \qquad y_2 = 5\underline{13}45 \qquad y_3 = 53\underline{41}5$$

This completes the first stage, producing 3 new red words w/ no long braids, so we add y_1, y_2, y_3 to our list of reduced words and move on,

starting by repeating the elementary step on y_1:

$x = 53145$

$\xrightarrow{0}$ $y_1 = \underline{35}145$ $\quad$ $\xrightarrow{1}$ $y_2 = 5\underline{13}45$ $\quad$ $\xrightarrow{2}$ $y_3 = 53\underline{41}5$

4 elementary steps

$y_1 = 35145$: $\cancel{\xrightarrow{0}}$ $\quad$ $\xrightarrow{1}$ 31545 — long braid $\Rightarrow$ not FC. Done!

(have words x, y_1, y_2, y_3, 31545 so far)

y_1 came from x via the braid starting at position 0, so we'd go back to x if we used same braid again. — Don't!

after the generation of a new word y we will check if y has a long braid. This is true here, so we can stop now!

Method 2: "Depth first" — once we get one y from x, update x to y and try to repeat, instead of considering all possible commutations on x.

use the leftmost possible commutation which does not repeat the last one

e.g. A_5. $w = 53145$

53145

0

35145

→ first stage

0   1

a repeat is still bad.

31545

Not FC. Done!

3 elementary steps

E.g. If we started from $w = 35145$ as input, we'd have

Method 2.

$$35145 \xrightarrow{0} 53145$$

$$53145 \xrightarrow{1} 51345$$

1 – earliest possible

$$51345 \xrightarrow{0} 15345 \xrightarrow{1} 13545$$

4 elementary steps

Method 1.

$$35145 \xrightarrow{0} 53145$$

$$35145 \xrightarrow{1} 31545$$

2 elt. steps!

Eg: A_5 $w = 135$. The reduced word graph is not a tree, and both method will end up producing a word already in our list. (i) : i th elt. step

Method 1.

$$
\begin{tikzcd}
 & 135 \arrow[dl, "1"'] \arrow[dr, "2"] & \\
315 \arrow[d, "3"'] & & 153 \arrow[d, "4"] \\
351 \arrow[dr, "5"'] & & 513 \arrow[dl, "6"] \\
 & 531 &
\end{tikzcd}
$$

(6) — doesn't produce a new elt

this is fine: simply don't add 531 to our list twice.

Method 2.

$$
\begin{tikzcd}
135 \arrow[d, "1"'] \\
315 \arrow[d, "2"'] \\
351 \arrow[d, "3"'] \\
531 \arrow[d, "4"'] \\
513 \arrow[d, "5"'] \\
153 \arrow[uuuuu, bend right, "6"']
\end{tikzcd}
$$

(6) : no new elt.

Remarks / Questions / To do :

(1) By Matsumoto's Thm, (I think) both the breadth first search (BFS) and the depth first search (DFS) method are guaranteed to get us all reduced words if we ignore the FC problem, so we won't miss any red word we should check in either method.

(2) We've seen both an example where the BFS method ends up using fewer steps and one where DFS uses fewer. Is there a better one in general? (BFS, DFS are well-known, and I only know a little about them. Do you know more?)

(3) We should write the code, for both methods!

Maybe we can time them to see if there's a winner.

(<u>Q:</u> Since BFS / DFS are so common, are there known smart implementations we can borrow from?)