begin cargo algorithm

Let $\pi \in S_n$

Write M carriers chained together, all containing n e's. Denote this string of carriers C_M

Denote the rightmost carrier c_1, the second rightmost carrier, c_2, the j^{th} rightmost carrier c_j, and the M^{th} rightmost (leftmost) carrier c_M

Write π to the right of C_M

for all i in $\{1, 2, ..., n\}$ **do**

Set $p := \pi_i$

begin element insertion process

for all j in $\{1, 2, ..., M\}$ **do**

Insert p into c_j using the traditional carrier algorithm rules

Set $p :=$ the element bumped from c_j

if $j = M$ **then**

Place p to the left of C_M

end if

end for

end element insertion process

end for

begin flushing non-e elements in C_M

while there are non-e elements in C_M **do**

Set $p := e$

Perform the element insertion process

end while

end flushing non-e elements in C_M

end cargo algorithm

- if p is a number in $[n]$, new p is the smallest elt that is larger than p
- if $p = e$, insert p at the end and bump the smallest elt in carrier c_j
- if $p = e$ & carrier is empty, trivial

takes $M \times$ # balls in carrier $C(1)$ steps

the details may be helpful for some conjectures, so include them

if π is a permutation, or is a configuration containing a permutation, the while loop will run $M \times i(w)$ times, where $i(w)$ is the size of a largest soliton

I've been thinking about a possible "simple" proof for (a) [Applying cargo train algorithm with M carriers is the same as applying the usual carrier algorithm M times] using induction. I was wondering if you see any logical flaws in this argument, since it somehow seems too elementary.

Proof sketch using induction:

Base case (add 1 additional carrier to the traditional carrier algorithm): Clearly, elements of the first carrier are ejected in the same order they are using the traditional carrier algorithm; there is nothing "new" yet. The only difference is that we only have information about one element at a time before we insert that element into the second carrier. But there is nothing wrong with only knowing information about one element in the traditional carrier algorithm. If elements enter the second carrier in the correct order, then they must leave in the correct order as well. So elements are ejected in the same order from the second carrier as they would be if we just applied the traditional carrier algorithm twice. (I think the key idea here is that after elements are ejected from the traditional carrier, they can't change order or interact at all.)

Inductive step: Here, we may assume that adding k-1 carriers to the traditional carrier algorithm (for a total of k) is the same as applying the traditional carrier algorithm k times. Now, if we add a k'th additional carrier, we can apply the same logic as in the base case. That is, adding a carrier is just like performing another traditional carrier move. Hence, having a total of k+1 carriers is like performing the traditional carrier algorithm k+1 times.

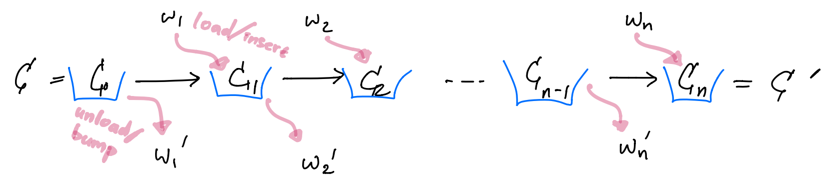
~Ben

1.12 Carrier algorithm

(Ref: Fukuda Sec 3.3 "Carrier algorithm")
and Sec 3.4 "Time evolution w/ a carrier")

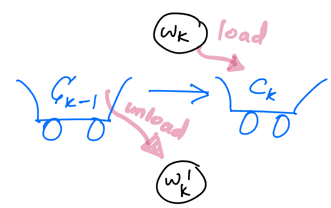
The carrier algorithm is a way to transform a finite sequence $w = (w_1, w_2, \dots, w_n)$
into another sequence $w' = (w_1', w_2', \dots, w_n')$ using
a weakly increasing sequence $\zeta = (c_1, c_2, \dots, c_m)$, called the carrier.

The carrier moves along w from left to right,
while the carrier passes a number w_k,
the carrier loads w_k and unloads w_k'.



Rule of loading and unloading for the carrier :

- Say the carrier is about to pass the number w_k in the sequence w,
  So we are about to load w_k onto the carrier.



- Let $\zeta_{k-1} = (c_1, c_2, \dots, c_m)$ be the sequence of
  $c_1 \leq c_2 \leq \cdots \leq c_m$
numbers which have been loaded on the carrier.

- Compare w_k with the numbers in the carrier ζ_{k-1}.

→ If there are numbers larger than w_k in the carrier ζ_{k-1},
then unload a smallest of those numbers,
and load w_k in its place, into the carrier.

→ Otherwise, all the numbers in the carrier ζ_{k-1} are smaller/equal to w_k.
Then unload a minimum entry in the carrier ζ_{k-1},
and load w_k onto the carrier.

More precisely,

$$w_k' = \begin{cases} \min \{ c_i \in \zeta_{k-1} \mid w_k < c_i \} & \text{if there are numbers in the carrier } \zeta_{k-1} \text{ larger than } w_k, \\ c_1 & \text{otherwise.} \end{cases}$$

ζ_k is the sequence we get from the previous carrier
by unloading w_k' and loading w_k.

Def Given sequences $\zeta = (c_1, c_2, \dots, c_m)$ and $w = (w_1, w_2, \dots, w_n)$,
$c_1 \leq c_2 \leq \cdots \leq c_m$
start with $\zeta_0 := \zeta$ and repeat the rule of unloading/loading for each $k = 1, \dots, n$
to get new sequences $\zeta' := \zeta_n$ and w'.
Call the transformation $(\zeta, w) \to (\zeta', w')$ the carrier algorithm.

[Prop 3.2, Fukuda] "Original" Carrier algorithm using the configuration

(Fukuda)
One BBS move can be described by the carrier algorithm,

with $C = (\underbrace{e, e, \dots, e}_{n})$ as the initial state of the carrier

The final state of the carrier will again be $(\overbrace{e, e, \dots, e}^{n})$.

- Applying one BBS move to a configuration $W = (\overset{\text{non-}e}{w_1}, \dots, \overset{\text{non-}e}{w_\ell})$

  is equivalent to starting with

  $$|\overbrace{e\, e \dots e}^{n}|\, w_1 w_2 \dots w_\ell \underbrace{e\, e \dots e}_{2\cdot i(w)}$$

  $i(\pi) :=$ the size of a longest soliton

  (if π is a permutation, $i(\pi)$ = size of a longest increasing subsequence)

  explained in [Fukuda], or you can check for yourself

  then applying the carrier algorithm once to get

  $$w' = w_1' w_2' \dots w_{\ell'}' \underbrace{e \dots e}_{i(w)} |e \dots e|$$

  where $\ell' = \ell + i(w)$

  (after $\ell + 2\cdot i(w)$ insertions/bumpings loadings/unloadings)

- Applying two BBS moves to $w_1, \dots, w_\ell \overbrace{e \dots e}^{i(w)} \overbrace{e \dots e}^{i(w)}$ is equivalent to applying the carrier algorithm to $w' \underbrace{e \dots e}_{i(w)}$ with again $(\underbrace{e, e, \dots, e}_{n})$ as the initial state of the carrier.

  We start with

  $$|\overbrace{e\, e \dots e}^{n}|\, w_1' w_2' \dots w_{\ell'}' \overbrace{e \dots e}^{i(w)}$$

  where $\ell' = \ell + i(w)$

  and end with

  $$w_1'' w_2'' \dots w_{\ell''}'' |\overbrace{e\, e \dots e}^{n}|$$

  where $\ell'' = \ell + 2\, i(w)$

  (after ℓ'' loadings/unloadings)

Process I

Consider the $\ell+2\cdot i(w)$ loadings/unloadings that happen in the first carrier algorithm from

$$\backslash e \cdots e / w_1 w_2 \cdots w_\ell \underbrace{e e \cdots e}_{i(w)} \underbrace{e e \cdots e}_{i(w)}$$

to

$$w_1' w_2' \cdots \underbrace{w_\ell' \cdots w_{\ell'}'}_{\text{first soliton}} \underbrace{e e \cdots e}_{i(w)} \backslash e \cdots e /$$

Call this process I.

The first loading/unloading happens when w_1 is inserted into the first carrier and $w_1' = e$ is moved from the first carrier to the left of the first carrier.

The second loading/unloading happens when w_2 is inserted into the first carrier and w_2' is moved from inside the first carrier to the left of it.

Continue in this manner until we have inserted w_ℓ into the first carrier and w_ℓ' is moved from inside the first carrier to its left.

At this point, there are exactly $i(w)$ copies of e's to the right of the first carrier, and there are exactly $i(w)$ many elements not equal to e in the carrier.

The loading/unloading rule tells us that inserting e into the carrier which contains a number which is not e results in moving the smallest number in the carrier to the left of the carrier.

So the next $i(w)$ loadings/unloadings are the following (trivial) Knuth moves where we insert e and bump c_k.

Let $i := i(w)$.

$$w_1' w_2' \cdots w_\ell' \lfloor c_1 c_2 \cdots c_i e e \cdots e / \underbrace{e e \cdots e}_{i} \longrightarrow w_1' w_2' \cdots w_\ell' c_1 \lfloor c_2 c_3 \cdots c_i e e \cdots e / \underbrace{e \cdots e}_{i-1}$$

where $c_1, c_2, \ldots, c_i \notin \{e\}$

$$\longrightarrow w_1' w_2' \cdots w_\ell' c_1 c_2 \lfloor c_3 \cdots c_i e \cdots e / \underbrace{e \cdots e}_{i-2}$$

$$\longrightarrow w_1' w_2' \cdots w_\ell' c_1 c_2 c_3 \lfloor \cdots c_i e \cdots e / \underbrace{e \cdots e}_{i-3}$$

$$\vdots$$

$$\longrightarrow w_1' w_2' \cdots w_\ell' c_1 c_2 c_3 \cdots c_{i-1} \lfloor c_i e e \cdots e / e$$

$$\longrightarrow \underbrace{w_1' w_2' \cdots w_\ell' c_1 c_2 c_3 \cdots c_i}_{\ell' = \ell + i} \lfloor e e \cdots e /$$

The final $i(w)$ loadings/unloadings are the moves where we insert e and bump e, until nothing is left to the right of the carrier

$$w_1' w_2' \cdots w_\ell' c_1 c_2 \cdots c_i e e \cdots \lfloor e e \cdots e / e \cdots e$$

Process II

$\ell'' = \ell + 2i(w) = \ell' + i(w)$

Consider the ℓ'' loadings/unloadings that happen in the second carrier algorithm

from $\lfloor e \cdots e / w_1' w_2' \cdots \underbrace{w_\ell' \ldots w_{\ell'}'}_{\substack{\text{first soliton} \\ c_1, c_2, \ldots, c_i}} \underbrace{e \cdots e}_{i(w)}$

to $w_1'' w_2'' \cdots \underbrace{w_\ell'' \cdots w_{\ell'}''}_{i(w)} \cdots \underbrace{w_{\ell''}''}_{\substack{\text{first soliton} \\ c_1, c_2, \ldots, c_i}} \lfloor e \cdots e /$

Call this Process II

The first loading/unloading happens when $w_1' = e$ is inserted into the second carrier and $w_1'' = e$ is moved from the second carrier to the left of the second carrier.

The second loading/unloading happens when w_2' is inserted into the second carrier and w_2'' is moved from inside the second carrier to the left of it.

Continue in this manner until we have inserted $w_{\ell'}'$ into the second carrier and $w_{\ell'}''$ is moved from inside the second carrier to its left.

At this point, there are exactly $i(w)$ copies of e to the right of the second carrier, and there are exactly $i(w)$ many elements not equal to e in the carrier.

The loading/unloading rule tells us that inserting e into the carrier which contains a number which is not e results in moving the smallest number in the carrier to the left of the carrier.

So the remaining $i(w)$ loadings/unloadings are the following trivial Knuth moves.

Let $i := i(w)$.

$w_1'' w_2'' \cdots w_{\ell'}'' \underbrace{\lfloor c_1 c_2 \ldots c_i e e \ldots e /}_{\text{2nd carrier}} \underbrace{e e \ldots e}_{i} \longrightarrow w_1'' w_2'' \cdots w_{\ell'}'' c_1 \lfloor c_2 c_3 \ldots c_i e e \ldots e / \underbrace{e \ldots e}_{i-1}$

where $c_1, c_2, \ldots, c_i \notin \{e\}$
(the same numbers that were left in the first carrier after ℓ loadings/unloadings starting from $w_1 w_2 \ldots w_\ell \underbrace{e e \ldots e}_{i(w)}$, the numbers in the first soliton for w)

$\longrightarrow w_1'' w_2'' \cdots w_{\ell'}'' c_1 c_2 \lfloor c_3 \ldots c_i e \ldots e / \underbrace{e \ldots e}_{i-2}$

$\longrightarrow w_1'' w_2'' \cdots w_{\ell'}'' c_1 c_2 c_3 \lfloor \ldots c_i e \ldots e / \underbrace{e \ldots e}_{i-3}$

$\vdots$

$\longrightarrow w_1'' w_2'' \cdots w_{\ell'}'' c_1 c_2 c_3 \ldots c_{i-1} \lfloor c_i e e \ldots e / e$

$\longrightarrow w_1'' w_2'' \cdots w_{\ell'}'' c_1 c_2 c_3 \ldots c_i \lfloor e e \ldots e /$

2-carrier algorithm

We claim that

performing the ℓ' loadings/unloadings that happen in the first carrier algorithm, $\ell' = \ell + i(w)$,

from $\quad \backslash\underline{e\cdots e}/ \, w_1 w_2 \cdots w_\ell \underbrace{ee\cdots e}_{i(w)} \underbrace{ee\cdots e}_{i(w)}$

to $\quad w_1' w_2' \cdots w_\ell' \underbrace{\cdots w_{\ell'}'}_{\substack{\text{first soliton} \\ c_1, c_2, \ldots, c_i}} \underbrace{ee\cdots e}_{i(w)} \; \backslash\underline{e\cdots e}/$

and then

$\ell'' = \ell + 2i(w) = \ell' + i(w)$

performing the ℓ'' loadings/unloadings that happen in the second carrier algorithm

from $\quad \backslash\underline{e\cdots e}/ \, w_1' w_2' \cdots w_\ell' \underbrace{\cdots w_{\ell'}'}_{\substack{\text{first soliton} \\ c_1, c_2, \ldots, c_i}} \underbrace{ee\cdots e}_{i(w)}$

to $\quad w_1'' w_2'' \cdots w_\ell'' \underbrace{\cdots w_{\ell'}''}_{i(w)} \underbrace{\cdots w_{\ell''}''}_{\substack{\text{first soliton} \\ c_1, c_2, \ldots, c_i}} \backslash\underline{e\cdots e}/$

would give the same result, $w_1'' w_2'' \cdots w_{\ell''}''$,

as performing one cargo-train ~~train~~ algorithm with two carriers

from $\quad \backslash\underline{e\cdots e}/\backslash\underline{e\cdots e}/ \, w_1 w_2 \cdots w_\ell \underbrace{\underbrace{ee\cdots e}_{i(w)} \underbrace{e\cdots e}_{i(w)}}_{\text{total } 2\cdot i(w) \text{ trailing } e\text{'s}}$

to $\quad w_1'' w_2'' \cdots w_{\ell'}'' \cdots w_{\ell''}'' \backslash\underline{e\cdots e}/\backslash\underline{e\cdots e}/$

$\ell'' = \ell + 2\cdot i(w)$

The first step in this 2-carrier algorithm is

to insert w_1 into the first carrier
and remove $w_1' = e$ from the first carrier, } 1st load/unload of Process I
then
to insert $w_1' = e$ into the second carrier
and move $w_1'' = e$ from the second carrier to the left of the second carrier. } 1st load/unload of Process II

initial 2-carrier & w: $\quad \backslash\underline{e\cdots e}/\backslash\underline{e\cdots e}/ \, w_1 w_2 \cdots w_\ell \underbrace{ee\cdots e}_{i(w)} \underbrace{e\cdots e}_{i(w)}$

after step 1: $\quad \overset{w_1''=e}{e} \; \backslash\underline{e\cdots e}/\backslash\underline{\overset{w_1'=e}{\ } w_1 e\cdots e}/ \overset{w_1}{\ } w_2 w_3 \cdots w_\ell \underbrace{ee\cdots e}_{i(w)} \underbrace{e\cdots e}_{i(w)}$

after step 2: $\quad \underbrace{w_1''}_{e} w_2'' \; \overset{w_2''}{\ }\backslash\underline{\underset{\text{maybe } e}{z}\cdots e}/\backslash\underline{\overset{w_2'}{\ } x\, \underset{\text{maybe } e}{y} \cdots e}/ \overset{w_2}{\ } w_3 \cdots w_\ell \underbrace{ee\cdots e}_{i(w)} \underbrace{e\cdots e}_{i(w)}$

The second step in this 2-carrier algorithm is

to insert w_2 into the first carrier
and remove w_2' from the first carrier, } 2nd load/unload of process I then
to insert w_2' into the second carrier
and move w_2'' from the second carrier to the left of the second carrier. } 2nd load/unload of process II

Continue in this fashion until the ℓ-th step in this 2-carrier algorithm,
which is to insert w_ℓ into the first carrier
and remove w_ℓ' from the first carrier, then
to insert w_ℓ' into the second carrier
and move w_ℓ'' from the second carrier to the left of the second carrier.

After the ℓ-th step: $$w_1'' w_2'' \cdots w_\ell'' \underbrace{\lfloor z_1 \cdots\cdots ee\cdots e /}_{\text{2nd carrier}} \underbrace{\lfloor c_1 c_2 \cdots c_i ee\cdots e /}_{\text{first carrier}} \underbrace{ee\cdots e}_{i} \underbrace{ee\cdots e}_{i}$$

where $c_1, c_2, \ldots, c_i \notin \{e\}$

At this point, there are exactly $2 \cdot i(w)$ copies of e to the right of the first carrier,
and the first carrier contains the first soliton of w
(if w is a permutation, then the first soliton is equal to the first row in the insertion tableau $P(w)$).

The next $i(w)$ steps are the following. (Let $i := i(w)$)

Insert e into the first carrier
and remove c_1 from the first carrier, then
insert c_1 into the second carrier
and move $w''_{\ell+1}$ from the second carrier to the left of the second carrier.

$$w_1'' w_2'' \cdots w_\ell'' \underbrace{\lfloor z_1 \cdots\cdots ee\cdots e /}_{\text{2nd carrier}} \underbrace{\lfloor c_1 c_2 \cdots c_i ee\cdots e /}_{\text{first carrier}} \underbrace{ee\cdots e}_{i} \underbrace{ee\cdots e}_{i}$$

where $c_1, c_2, \ldots, c_i \notin \{e\}$

$w''_{\ell+1} = z_1$ $\quad c_1 \quad e$

$$w_1'' w_2'' \cdots w_\ell'' \underbrace{\lfloor c_1 \cdots\cdots ee\cdots e /}_{\text{2nd carrier}} \underbrace{\lfloor c_2 \cdots c_i \quad ee\cdots e /}_{\text{first carrier}} \underbrace{ee\cdots e}_{i-1} \underbrace{ee\cdots e}_{i}$$

Insert e into the first carrier
and remove c_2 from the first carrier, then
insert c_2 into the second carrier
and move $w''_{\ell+2}$ from the second carrier to the left of the second carrier.

Continue in this way until we have inserted e into the 1st carrier and removed c_i from the first carrier, then

inserted c_i into the second carrier

and move $w''_{\ell+i} = w''_{\ell'}$ from the second carrier to the left of 2nd carrier:

$$w_1'' w_2'' \cdots w_\ell'' \cdots w_{\ell'}'' \underbrace{c_1 c_2 \cdots c_i\, ee\cdots e}_{\text{2nd carrier}} \underbrace{\qquad ee\cdots e}_{\text{first carrier}} \underbrace{ee\cdots e}_{i}$$

The final $i(w)$ steps insert e into the 1st carrier and remove e from the 1st carrier, then } the $\ell'+k$-th load/unload of Process I

insert e to the second carrier and move $c_k = w''_{\ell'+k}$ from the 2nd carrier to its left } the $\ell'+k$-th load/unload of Process II

(for $k = 1, 2, \ldots, i$).

So the end result is $w_1'' w_2'' \cdots w_\ell''$.

Questions

True/false?

- If $\pi \in S_n$, then the number M of solitons of π (i.e height of BBS soliton partition, i.e $des^*(\pi) = des(\pi)+1$)

  is an upper bound for the minimum time required for π to decompose into solitons.

  That is, at $t=M$, the BBS configuration has decomposed into solitons.

Reason: After one carrier algorithm, we put the first soliton in the first carrier.

The second carrier algorithm has no effect on the first soliton (other than moving it forward by $i(\pi)$).

After n steps using the train (or M-carrier) algorithm, we would be in a state where the first carrier has the first soliton.

(This follows from the def of M-carrier algorithm)

In general, the 2nd carrier may be empty or not empty.

Question True or false? (If true, this would prove above)

If $M = des^*(\pi)$,

is it true that the second soliton is in some carrier farther left,

the third soliton is in some carrier even farther left,

...

the penultimate soliton is in the M-th carrier

and the final soliton is *not* in a carrier.