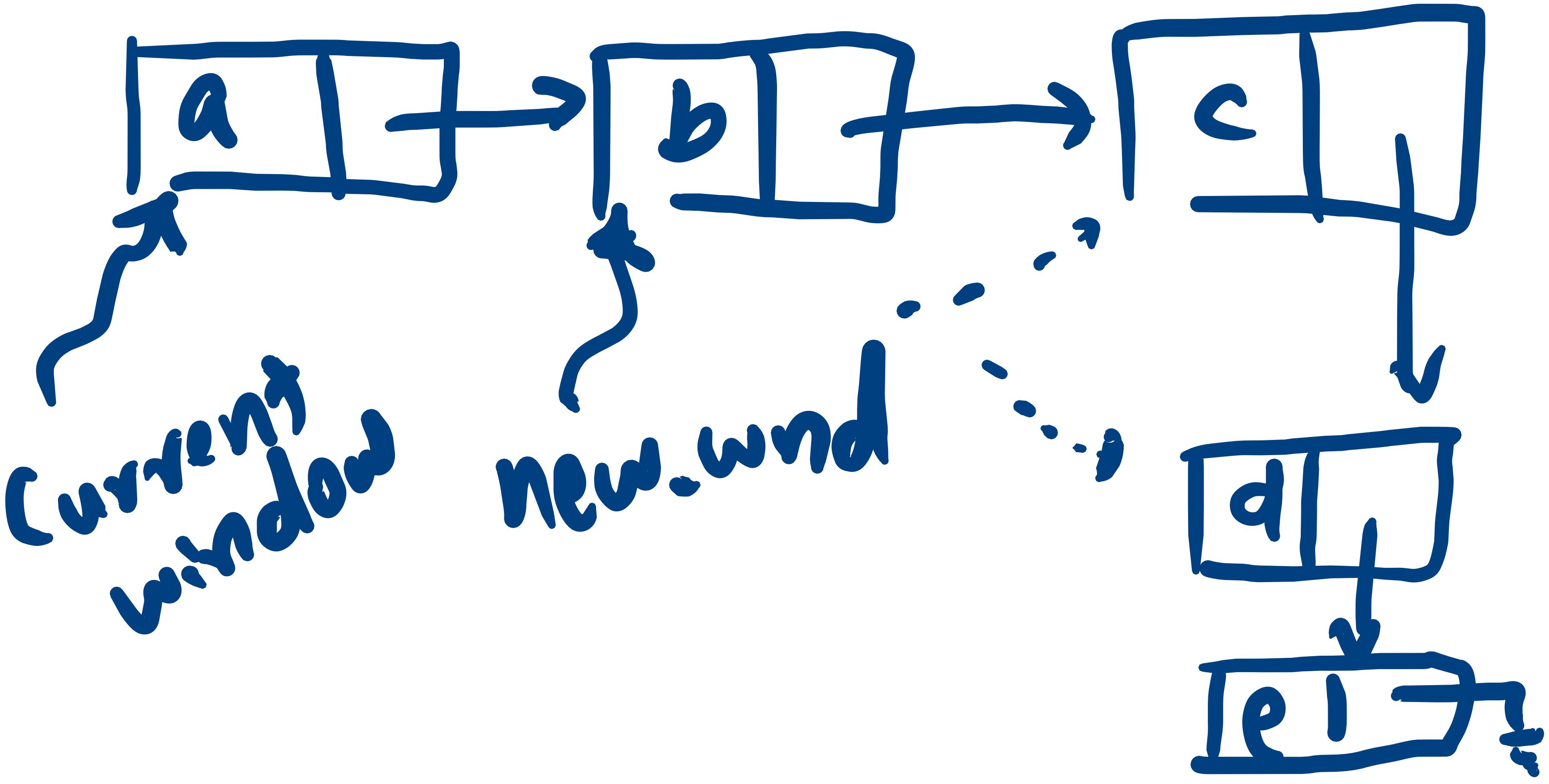
An application of linked lists.

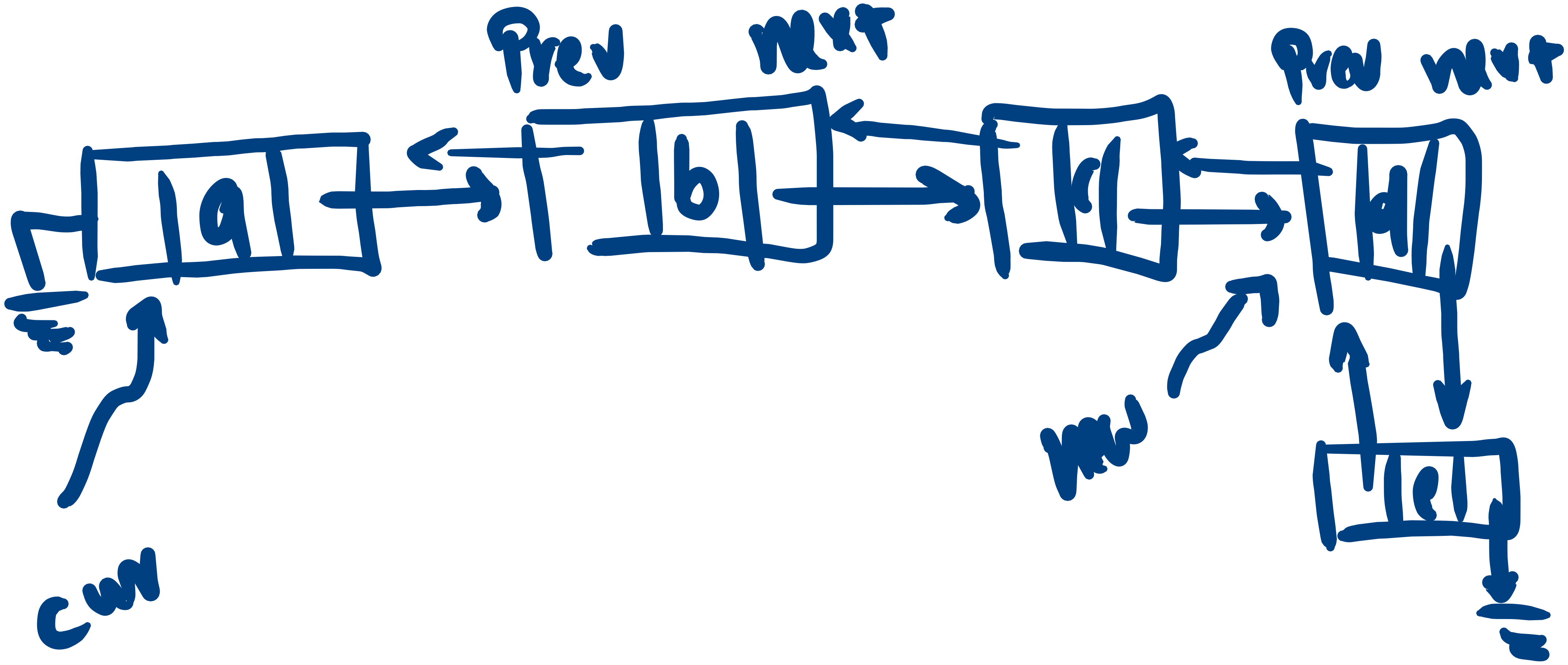


Each tab moves new-wnd to next.

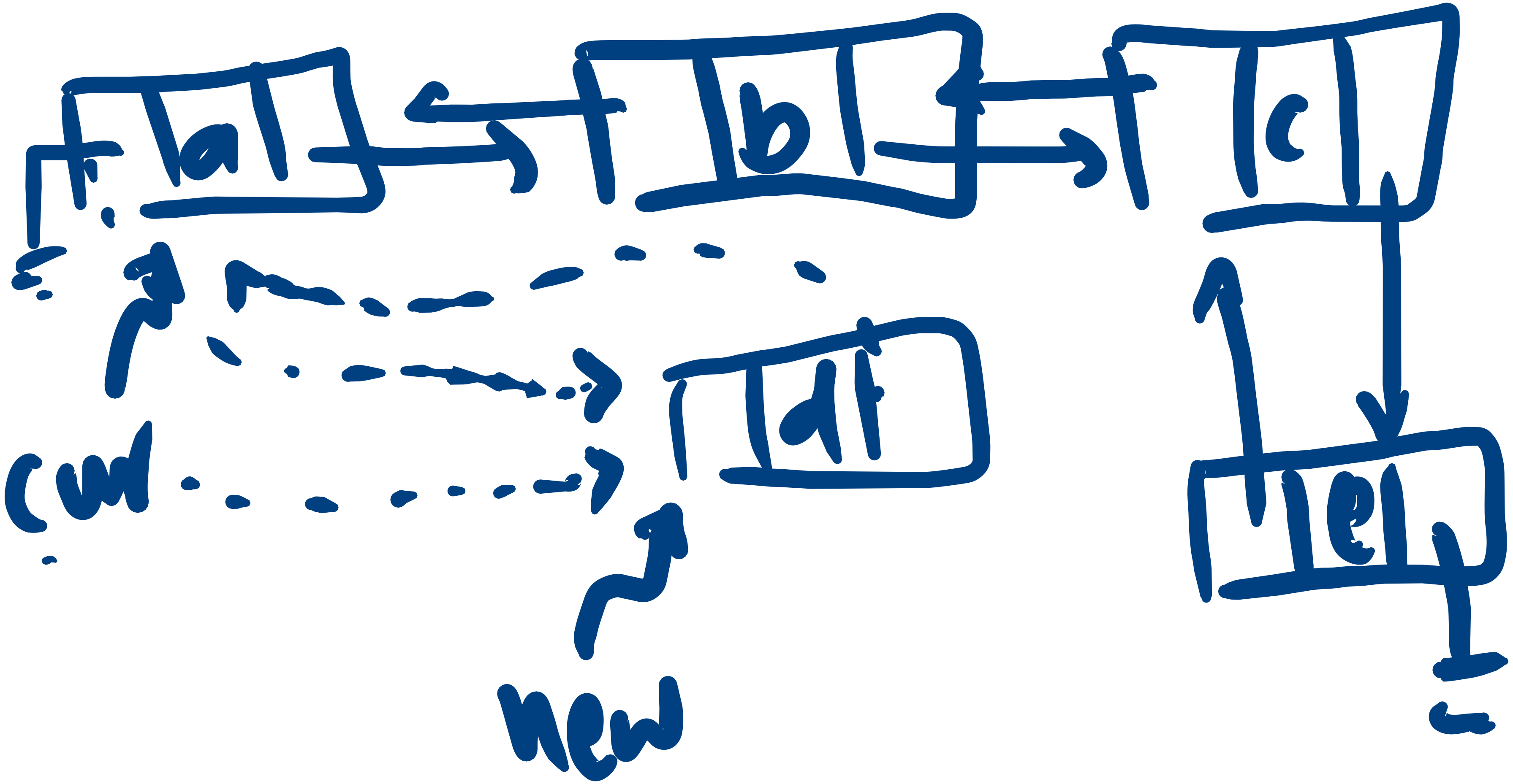
On release:

- Unlink new_wnd
- Insert it at the head of the list

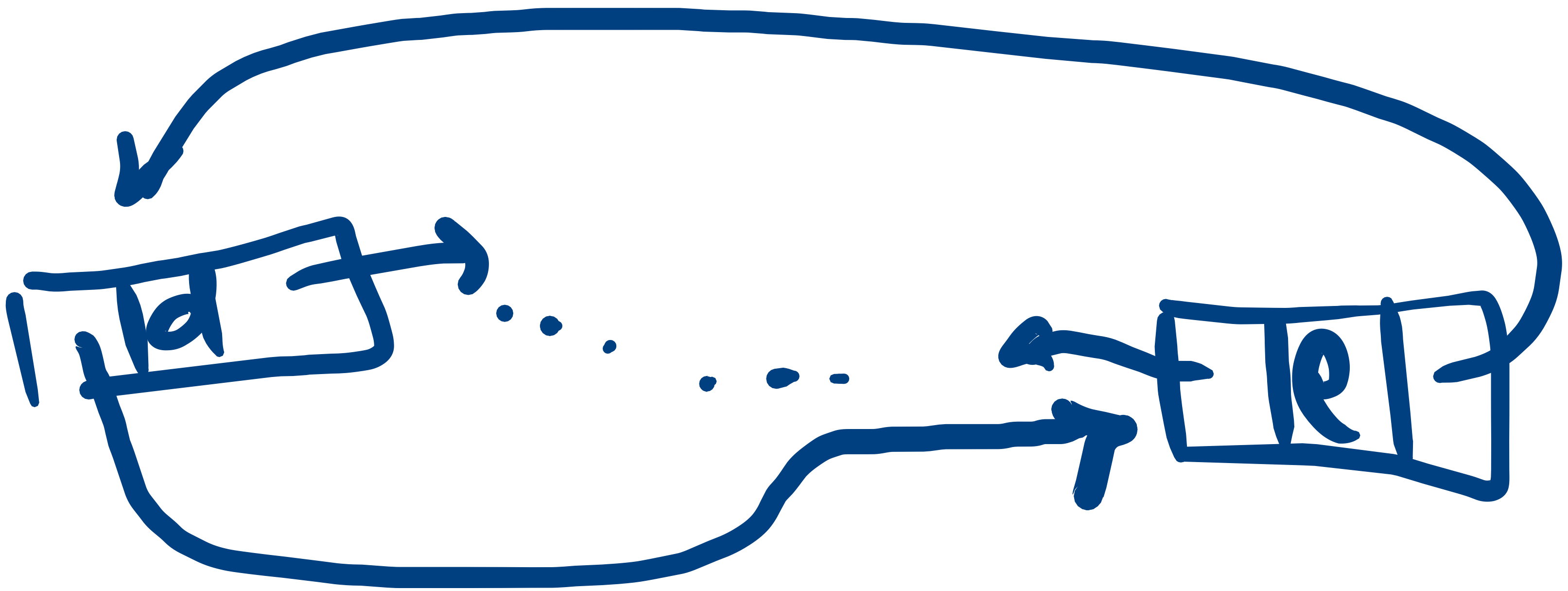
Problem. How to unlink?.

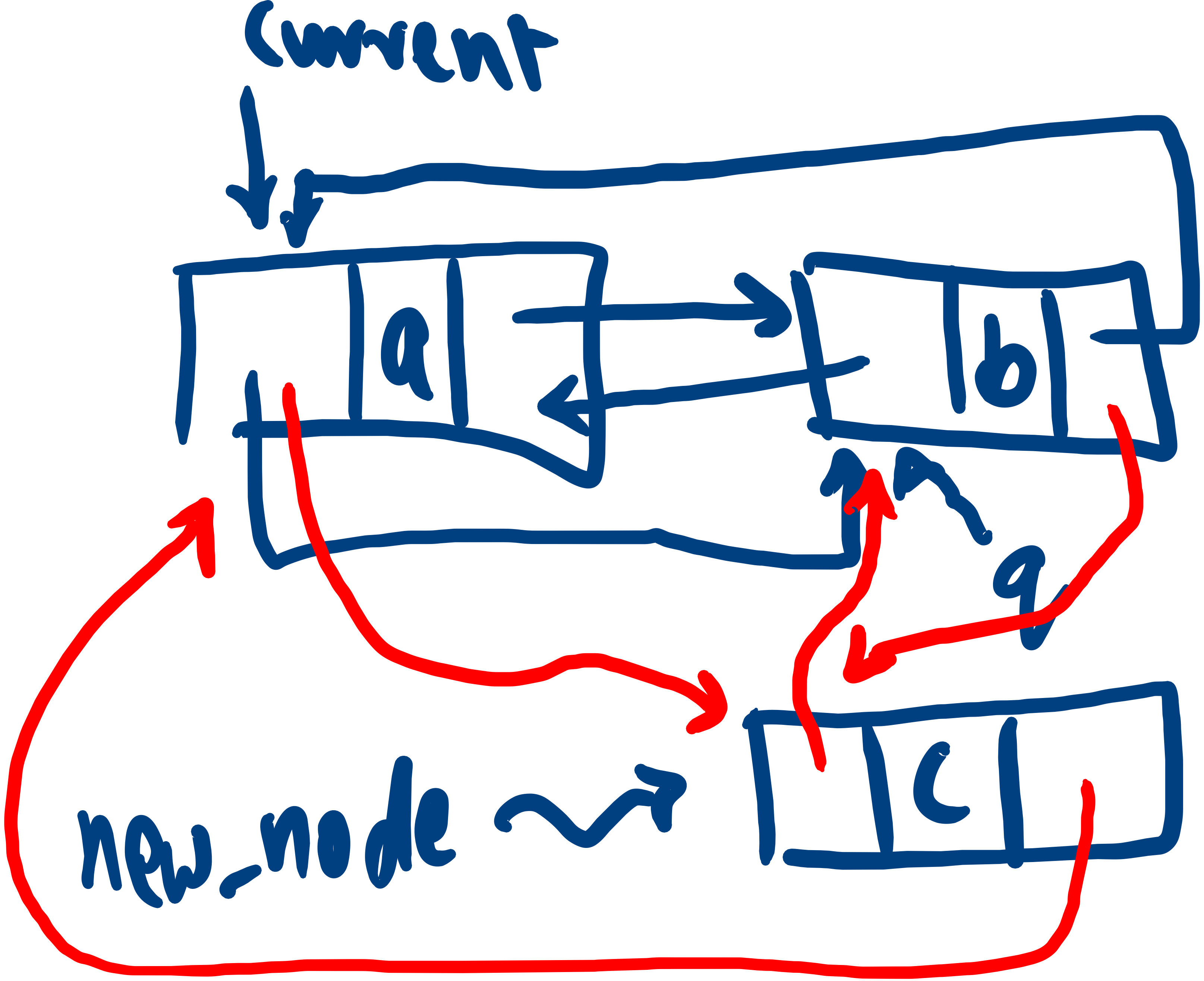

prev
next
prev
next
a
b
c
d
e
curr
new

After unlinking d



The actual ds is circular!

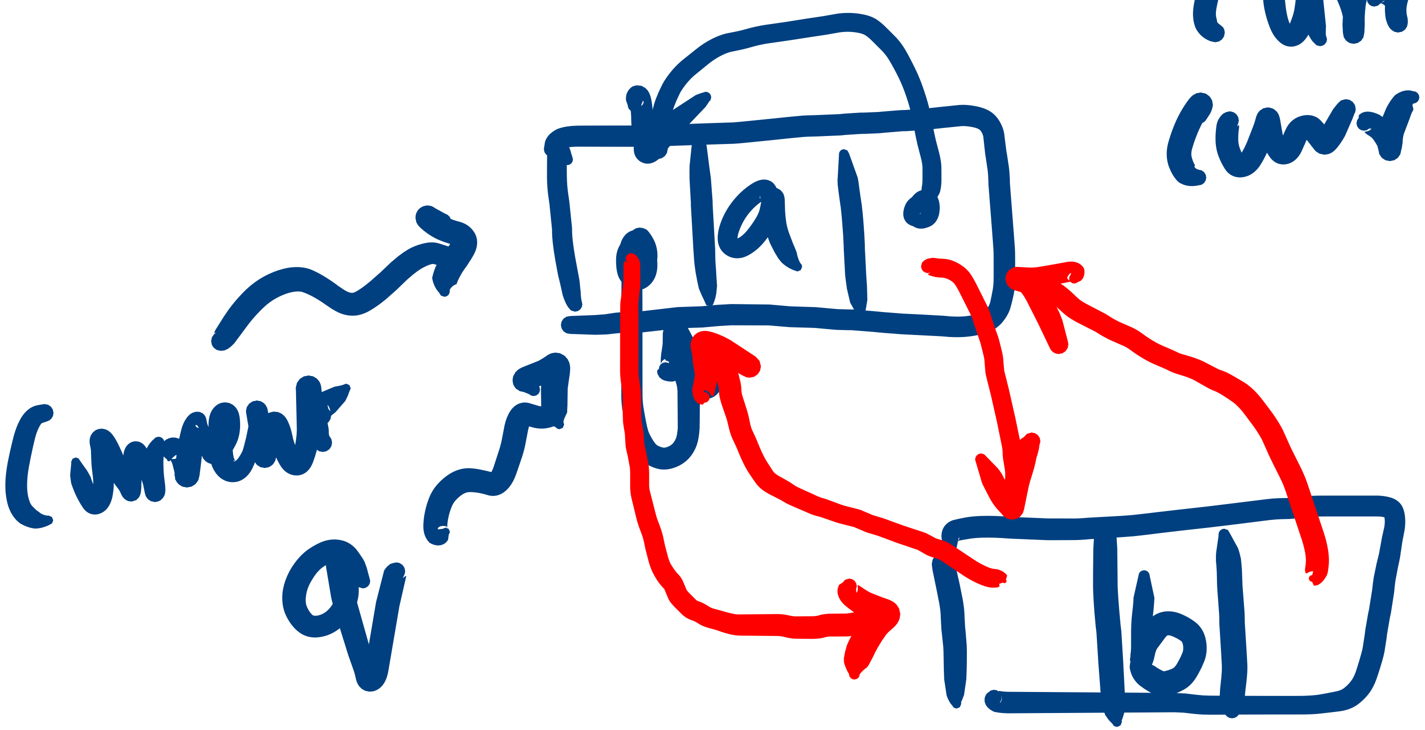



current
a
b
c
new_node

What about the empty list?.

list with 1 elt?.

List with 1 elt



current->next=q curr
curr->prev=curr

Avoiding special code for empty lists

- Sentinel node technique.
  - ↳ use a node with an invalid data valu as a node that will always be present in the list.

Analysis of rehash()

↳ Time Complexity

$O(n)$

Say we start with a table of 11 slots. First 21 insertions ⇒ ~1 unit

22nd insertion ⇒ ~22 units

next prime 23 $\underset{\sim}{23}$ insertions

all insertions until 45th ~ 1 unit of time

46th => ~ 46 units

Next prime. 47

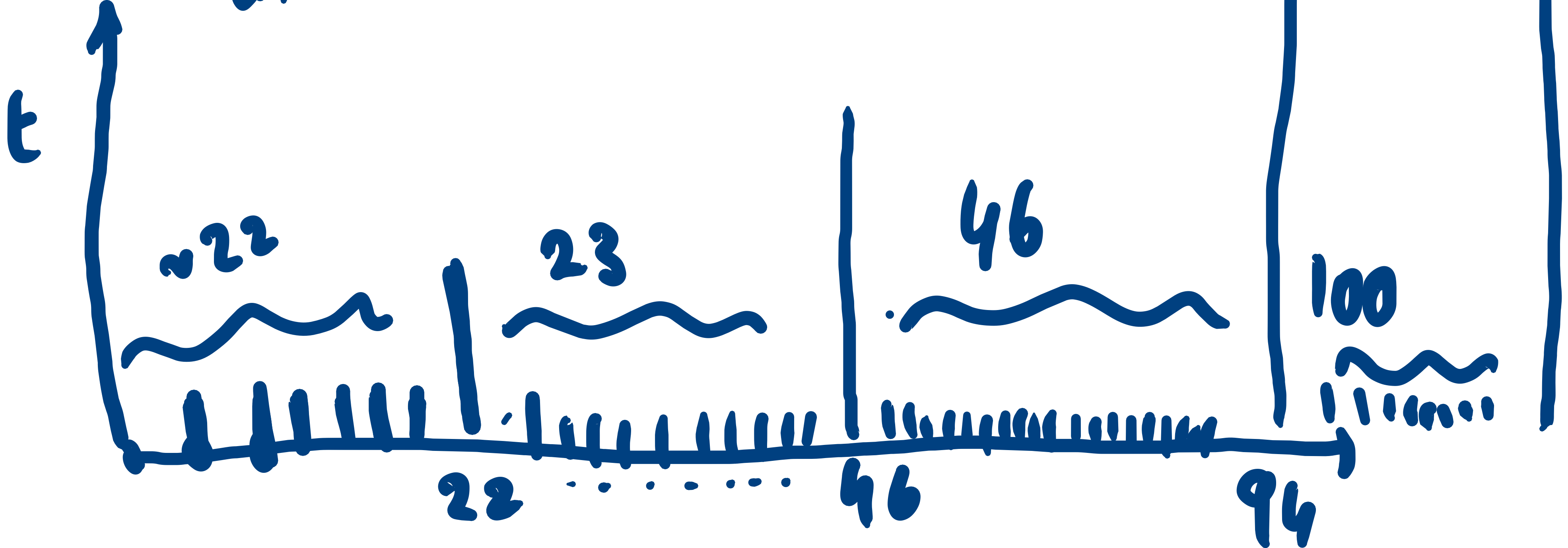
next 47 insertions ~ 1 unit

next ~ 94 units

next prime ~ 97

~~nxt~~ 100 insertions ~1 unit of time

next ~ 194 units of time



Key Points

- The i^{th} rehash takes about 2x time as the $(i-1)^{st}$ rehash()
- The # insertions b/w i^{th} and $(i+1)^{st}$ rehash is 2x the # insertions b/w $(i-1)^{st}$ & i^{th} rehash()

average of any prefix of this infinite list

1, 2, 1, 1, 4, 1, 1, 1, 1, 8, 1, 1, 1, 1, 1, 1,
1, 1, 16, . . .