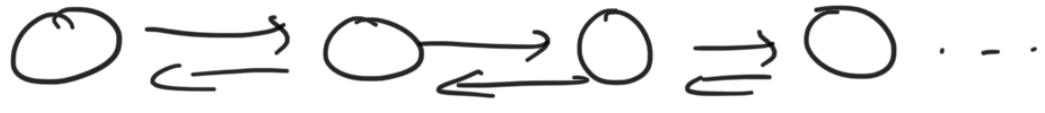
Linked lists

Linked list

Doubly linked list



to find k^{th} element in list → O(K)

but removing items from beg of list

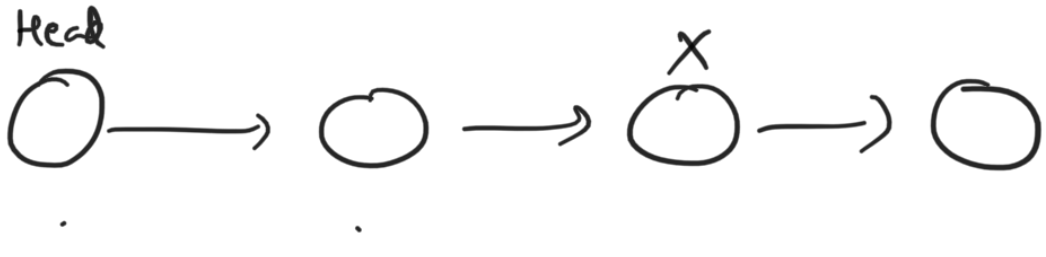
- O(1)

```
class ListNode:

    def __init__(self, val=0,
                 next=None):

        self.val = val
        self.next = next
```

Delete a node



scan through nodes

if node.next = X,

node.next = X.next

Runner technique

Iterate through linked list with 2 pointers simultaneously, one ahead of other (fixed amount/diff speed :-))

(ex) even length linked list

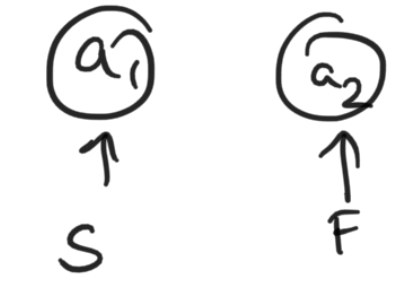
$a_1 \to a_2 \to \cdots \to a_n \to b_1 \to b_2 \to \cdots \to b_n$

to rearrange to

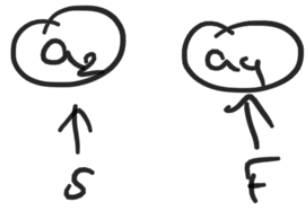
$a_1 \to b_1 \to a_2 \to b_2 \cdots$

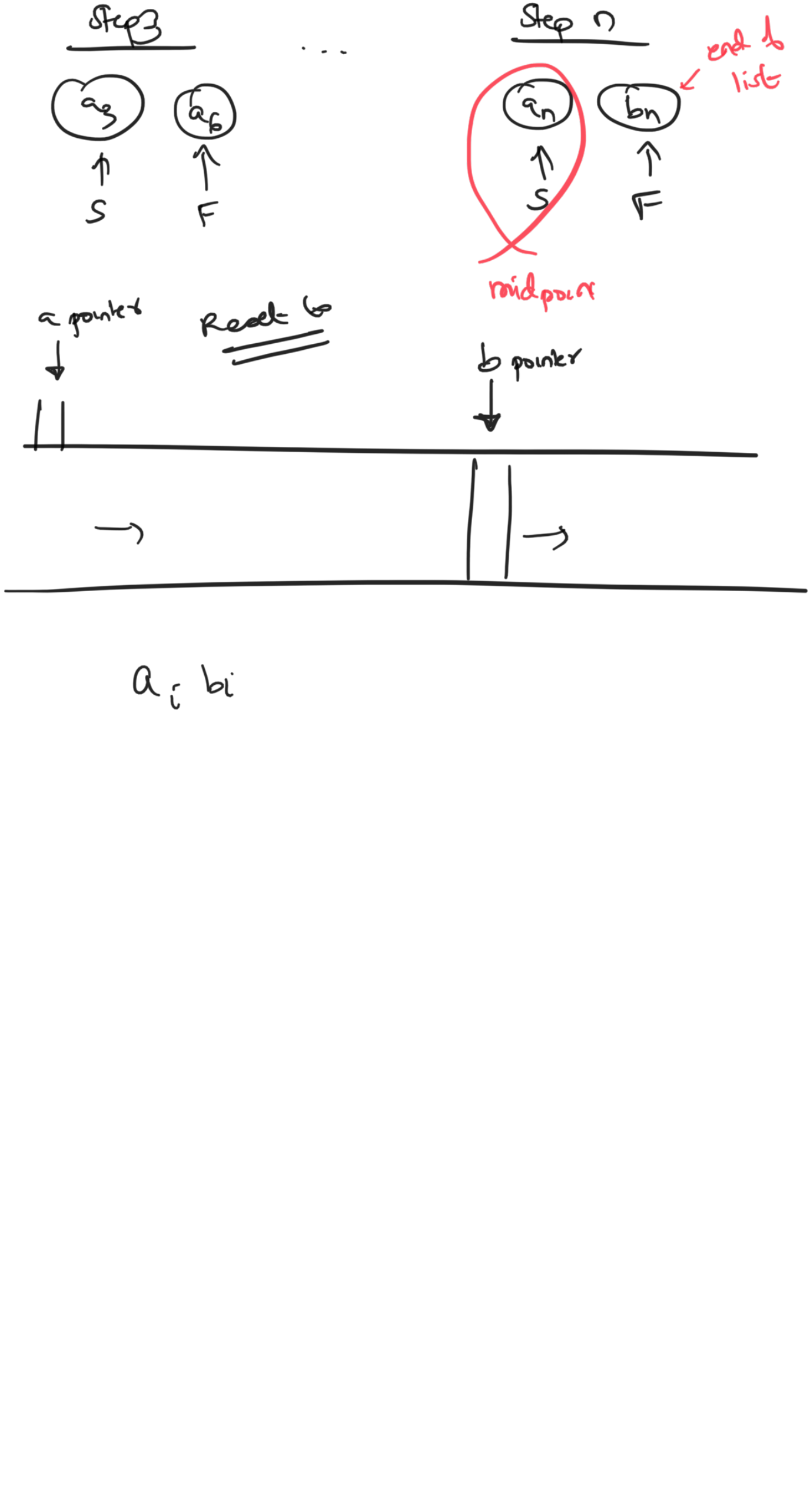
fast pointer - moves 2 elements for 1 move of slow pointer

Step 1



Step 2




Step 3
Step n
end of list
S
F
S
F
midpoint
a pointer
Reset b
b pointer

a_i b_i