The linked list

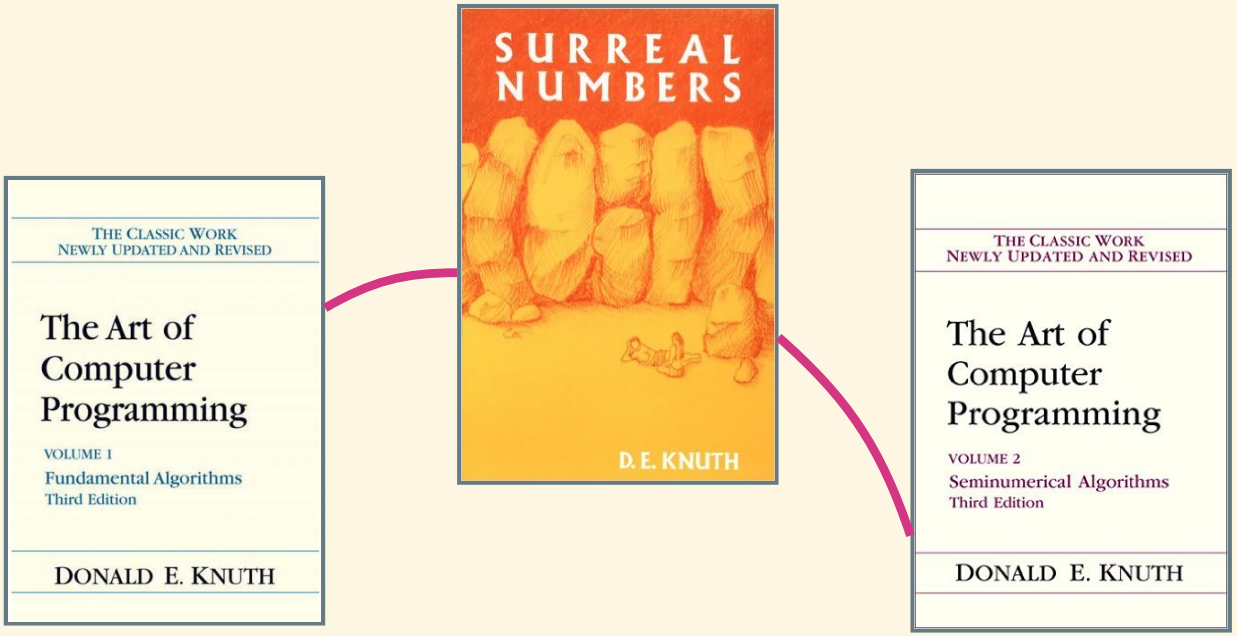
EECS 214, Fall 2018

A problem with vectors

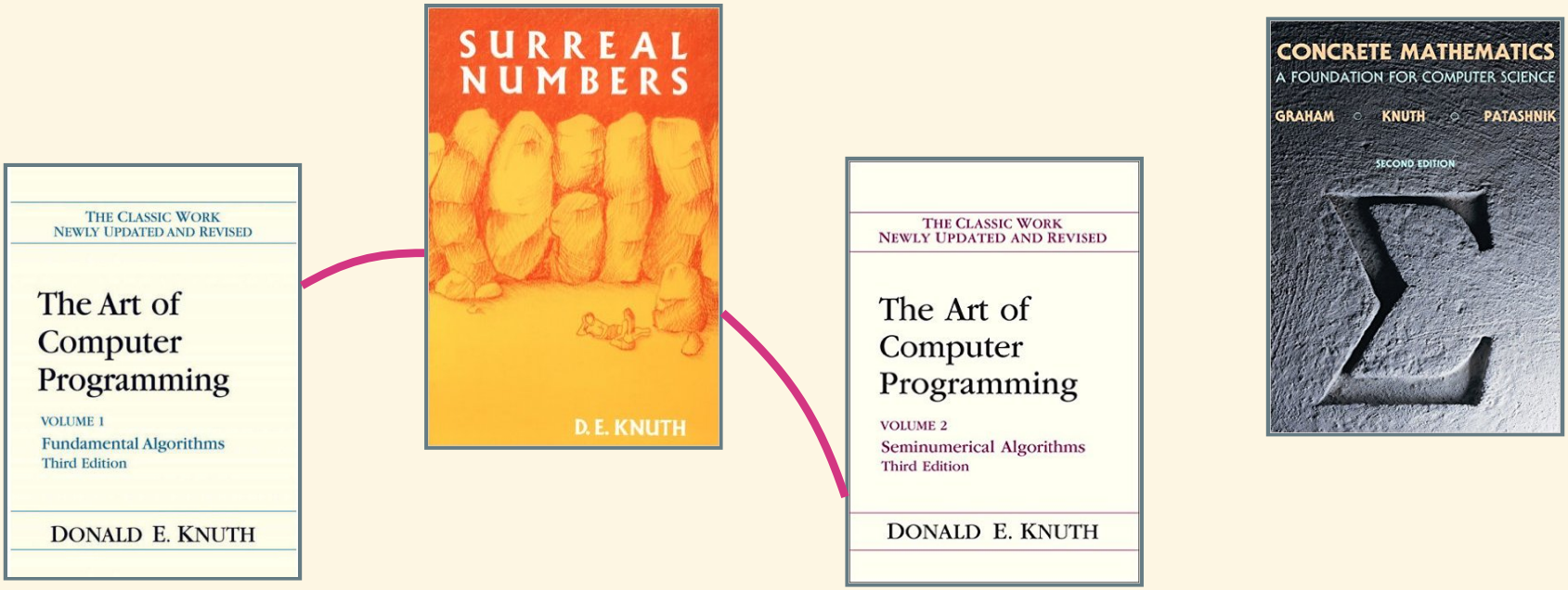
A problem with vectors

What if we want to add 6 between 5 and 7?

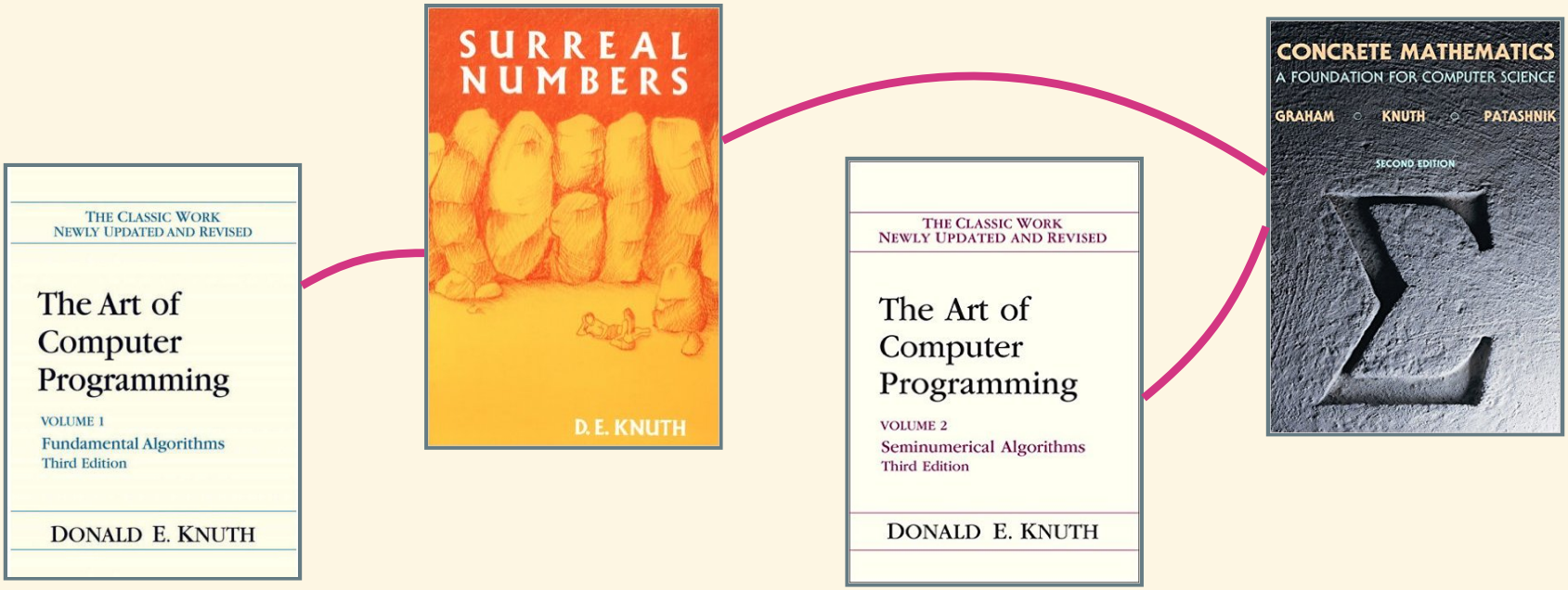
Books on a string



Books on a string

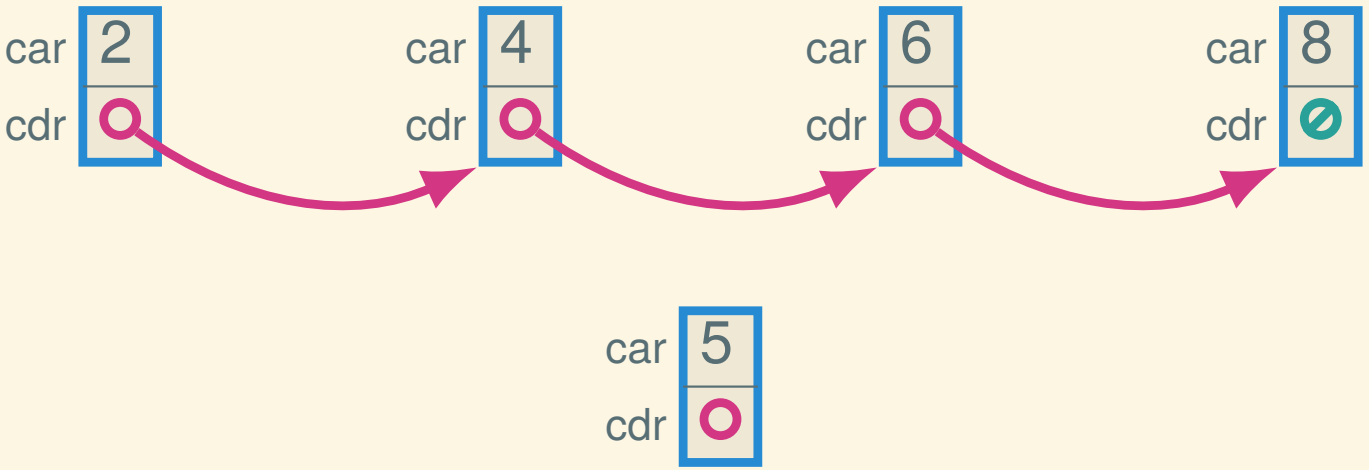


Books on a string

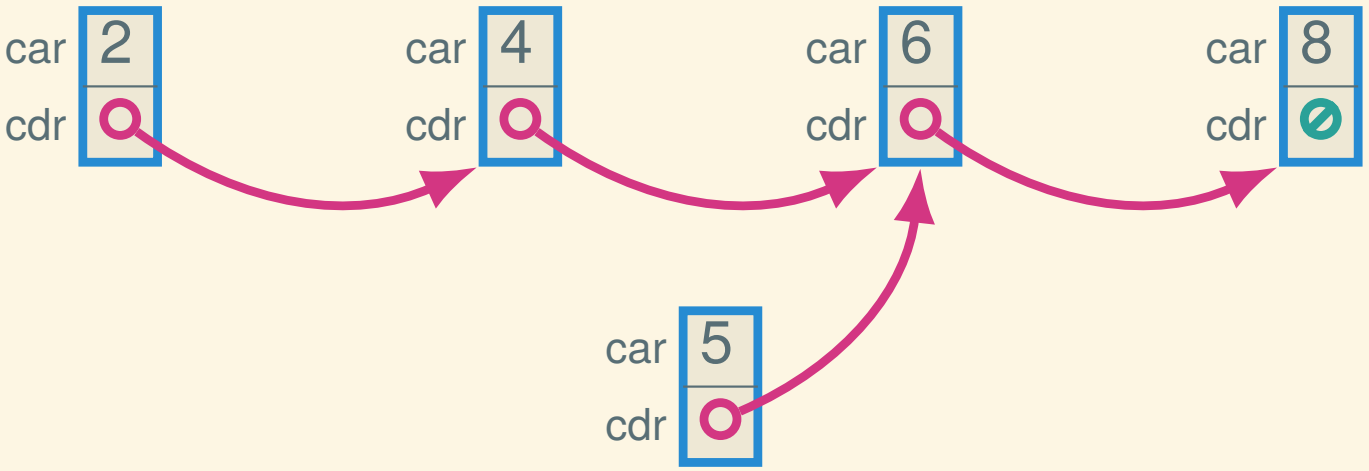


Nodes and pointers

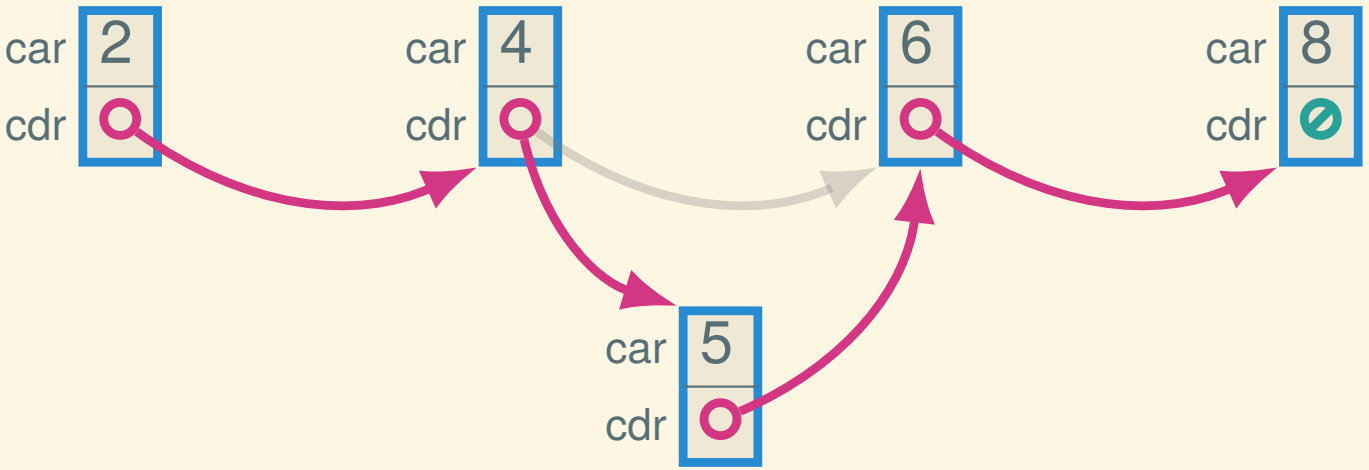
Nodes and pointers



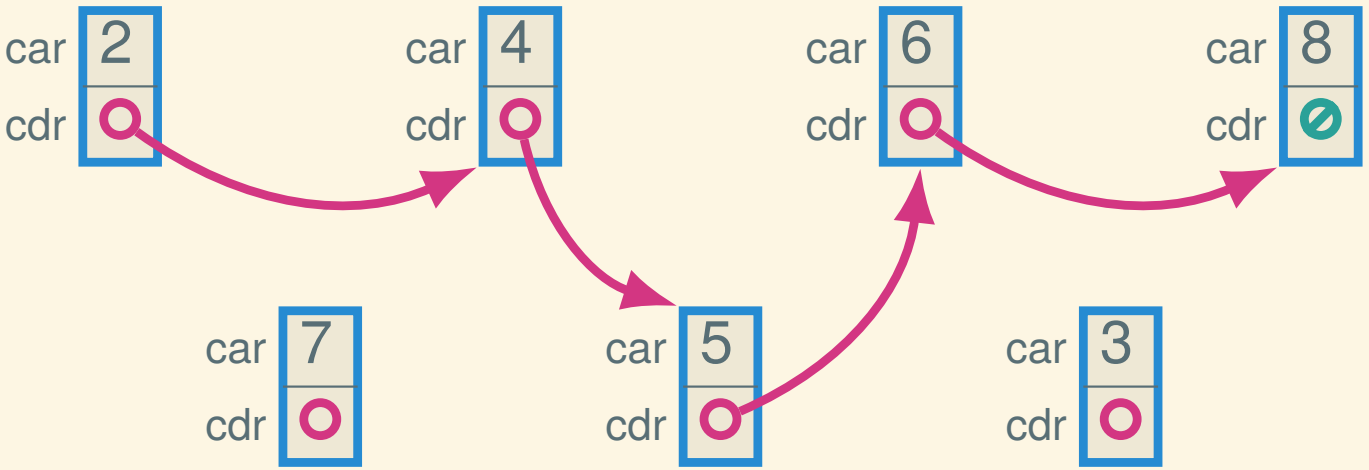
Nodes and pointers



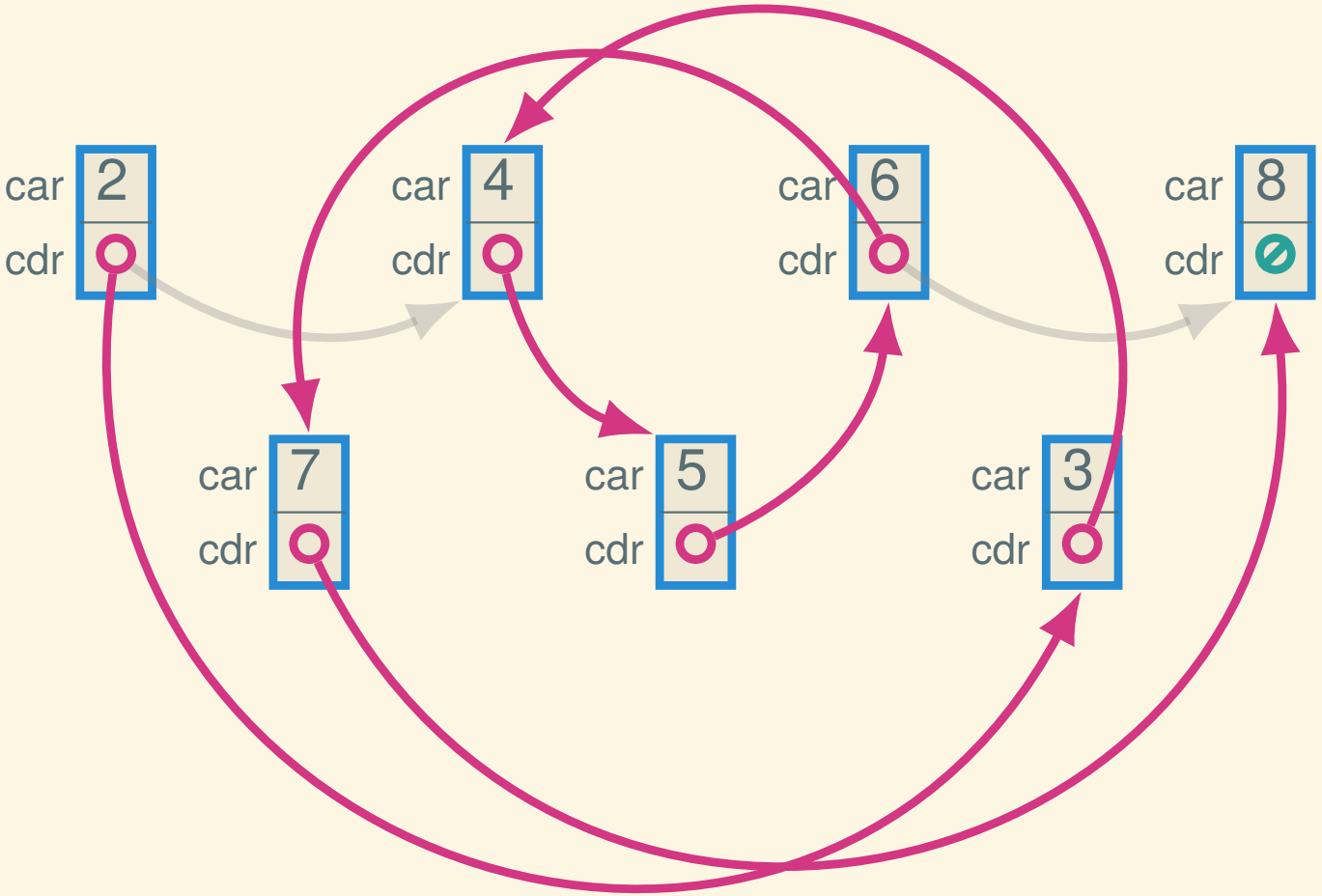
Nodes and pointers



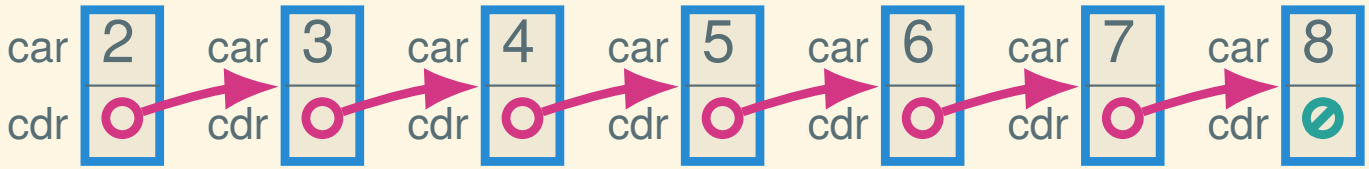
Nodes and pointers



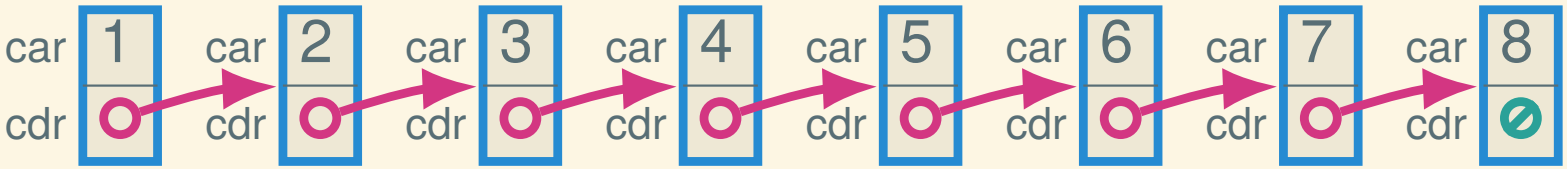
Nodes and pointers



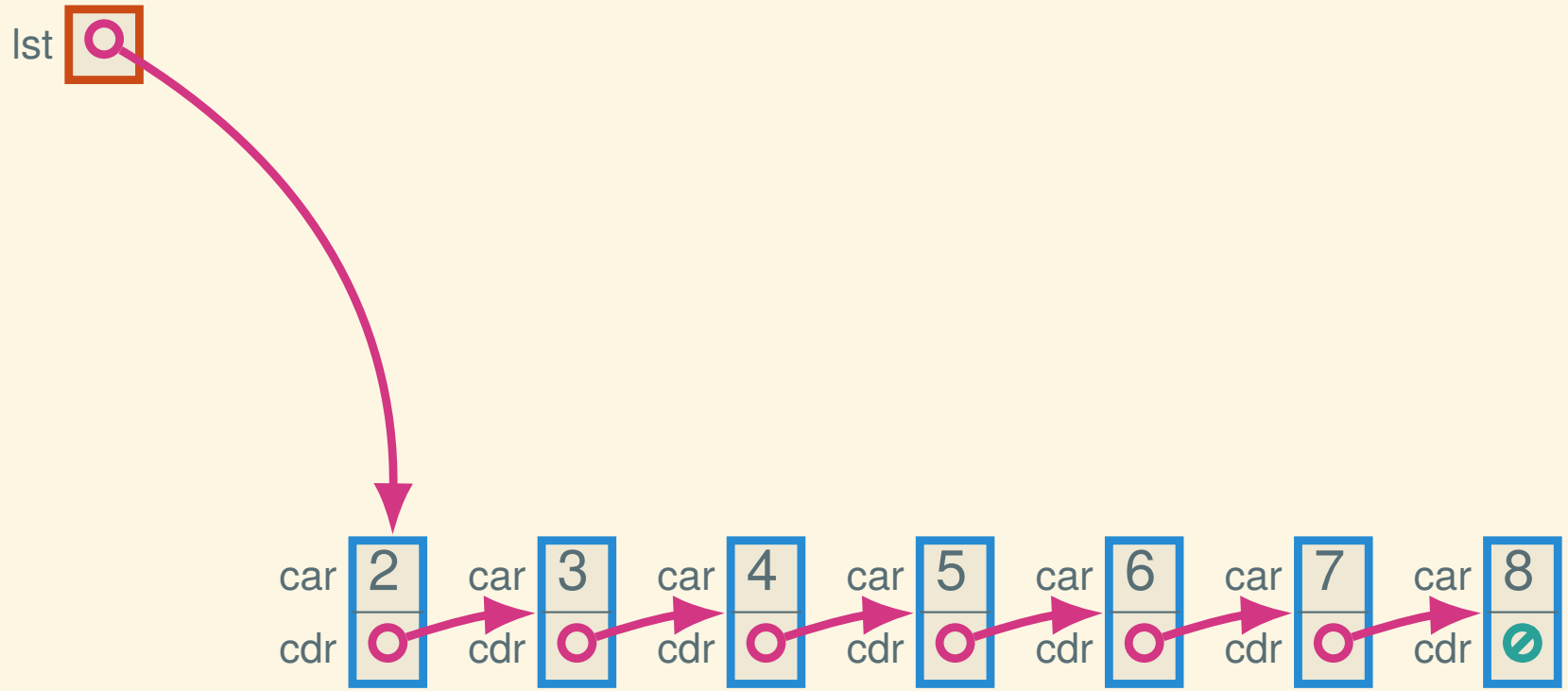
Inserting at the beginning



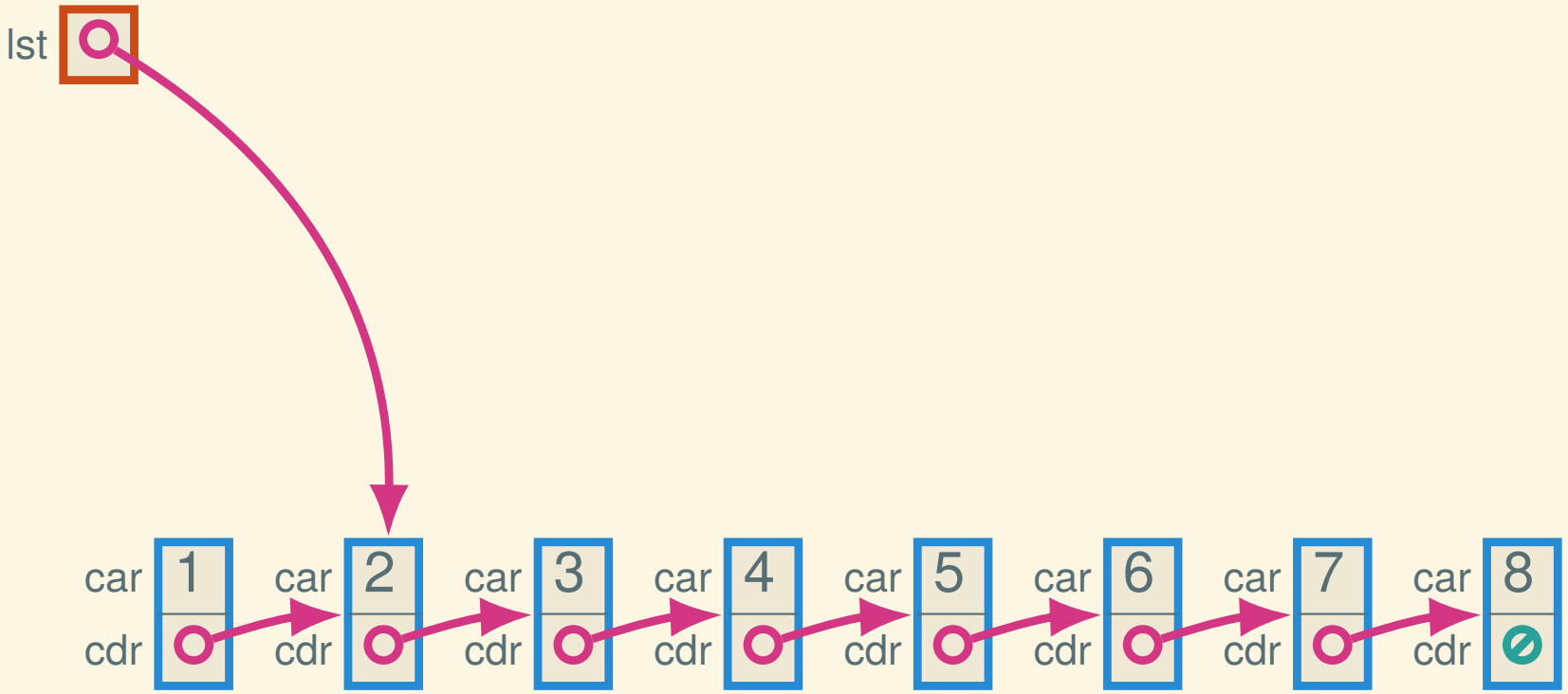
Inserting at the beginning



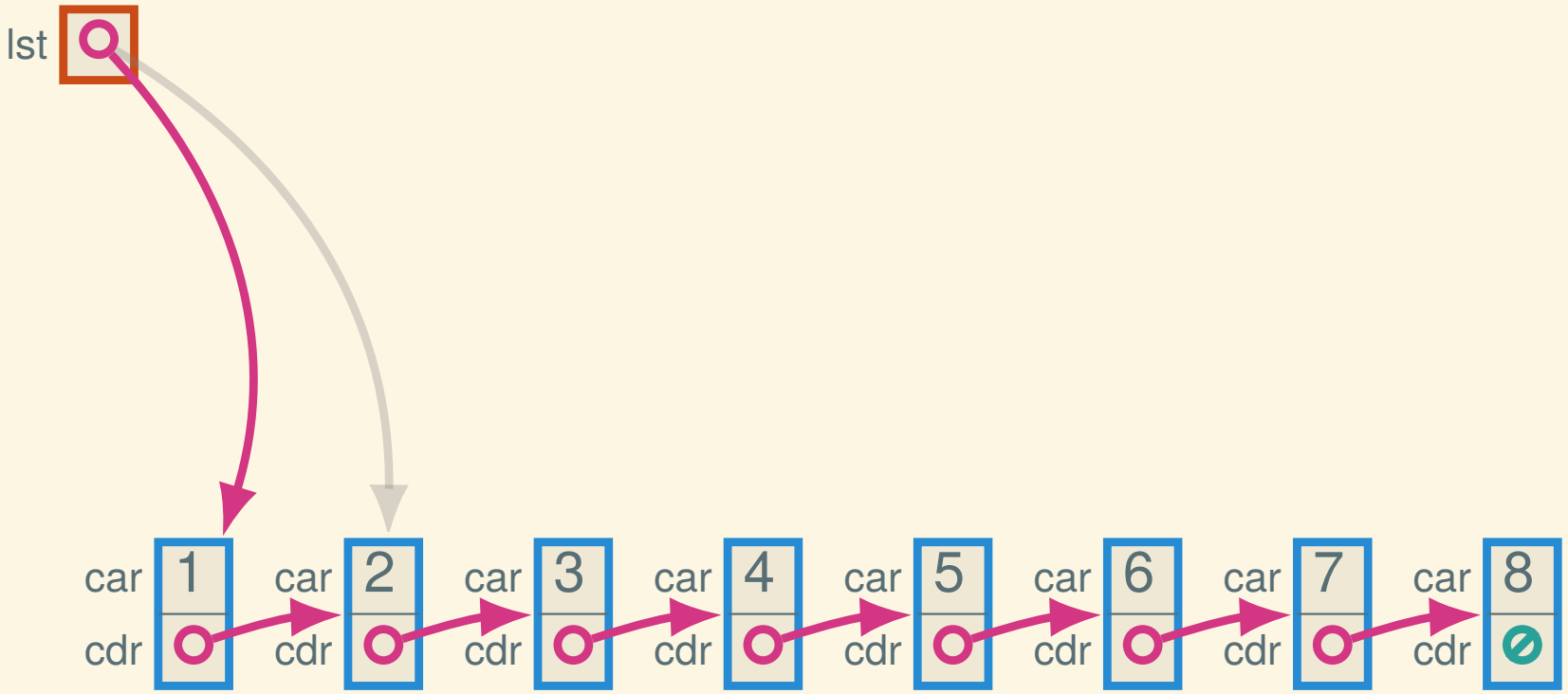
Inserting at the beginning



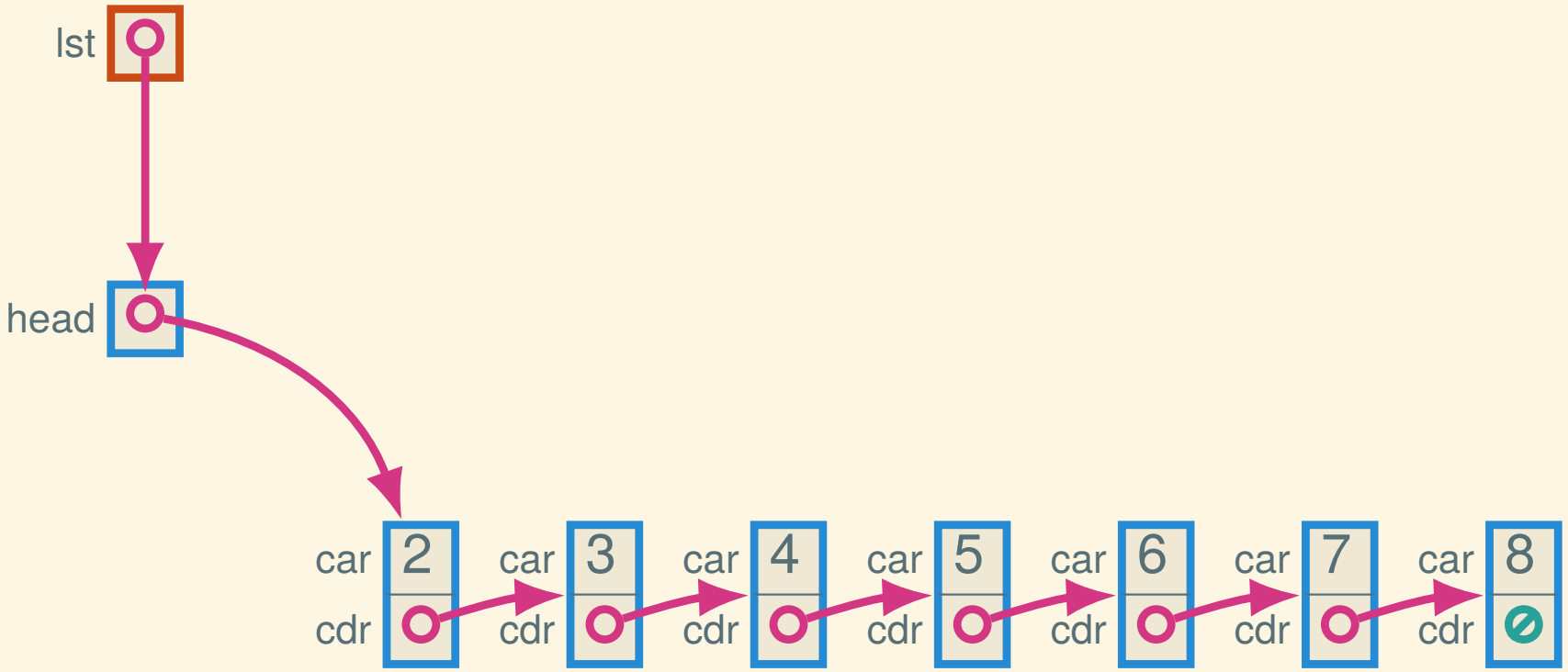
Inserting at the beginning



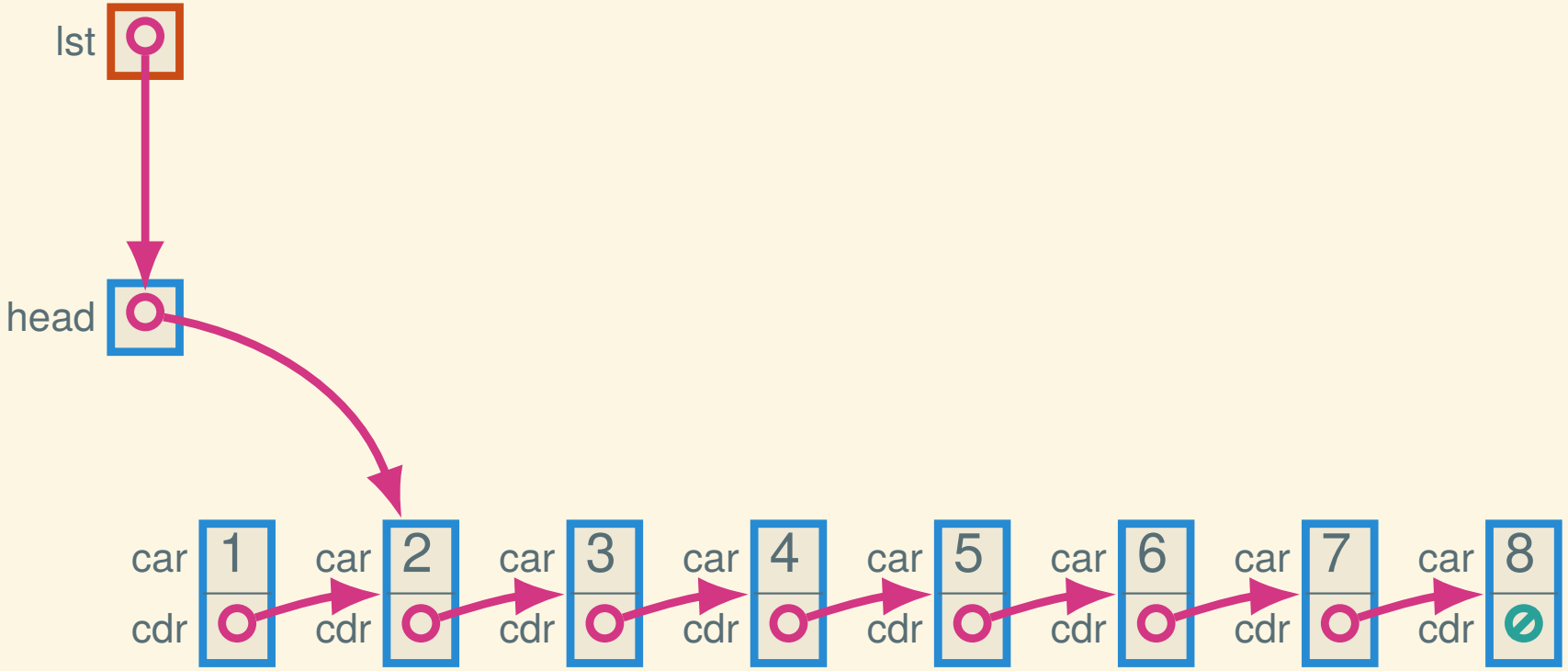
Inserting at the beginning



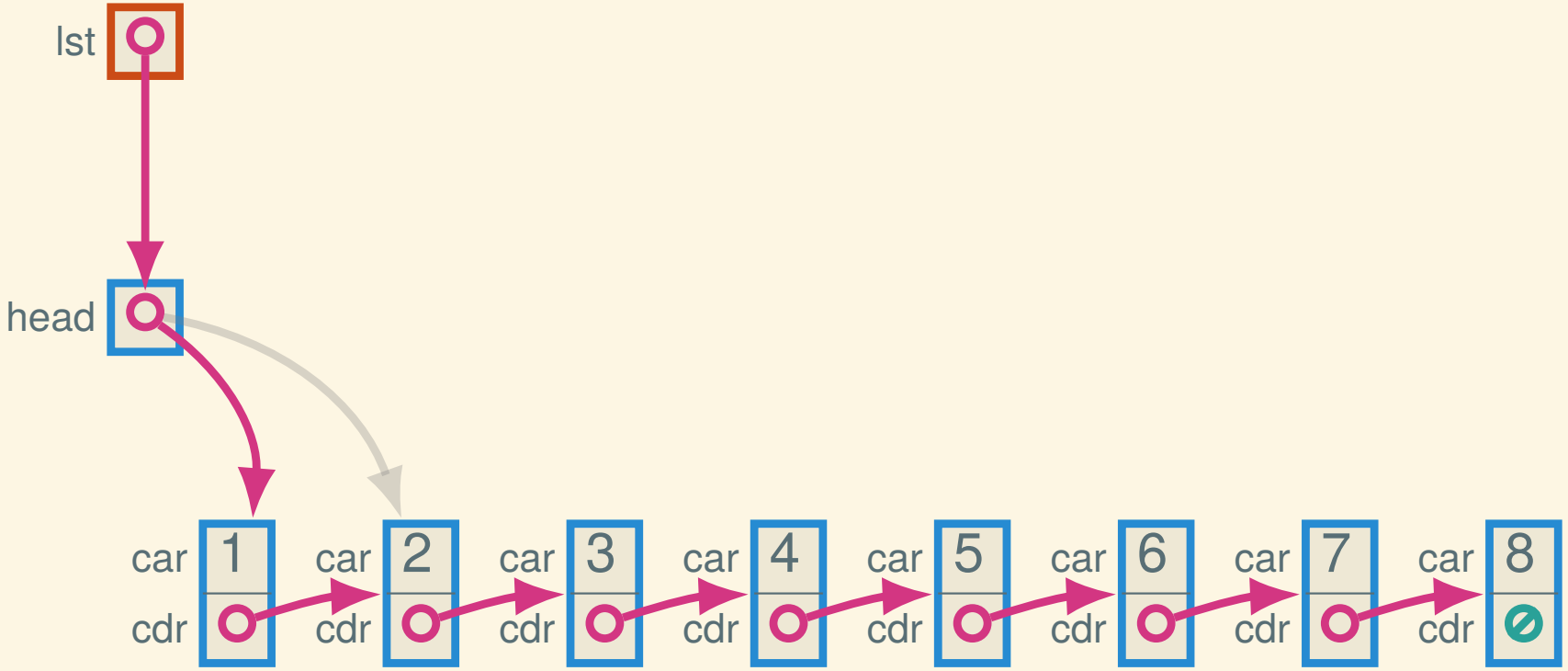
Indirection



Indirection



Indirection



Now in DSSL2

Linked lists in DSSL2

```
# Link is one of:
# - node { data: Number, next: Link }
# - nil()
struct node:
    let data
    let next
struct nil: pass

class LL:
    let head

    def __init__(self):
        self.head = nil()
```

Linked lists in DSSL2

```
# Link is one of:
# - node { data: Number, next: Link }
# - nil()
struct node:
    let data
    let next
struct nil: pass

class LL:
    let head

    def __init__(self):
        self.head = nil()

    def push_front(self, data):
        self.head = node(data, self.head)
```

List operations in DSSL2

```
class LL:
    …

    def get_front(self):
        if node?(self.head): self.head.data
        else: error('LL.get_front: got empty list')
```

List operations in DSSL2

```
class LL:
    …

    def get_front(self):
        if node?(self.head): self.head.data
        else: error('LL.get_front: got empty list')

    def get_nth(self, n):
        let curr = self.head
        while n > 0:
            if nil?(curr):
                error('get_nth: list too short')
            n    = n - 1
            curr = curr.next
        curr.data
```

More DSSL2 list operations

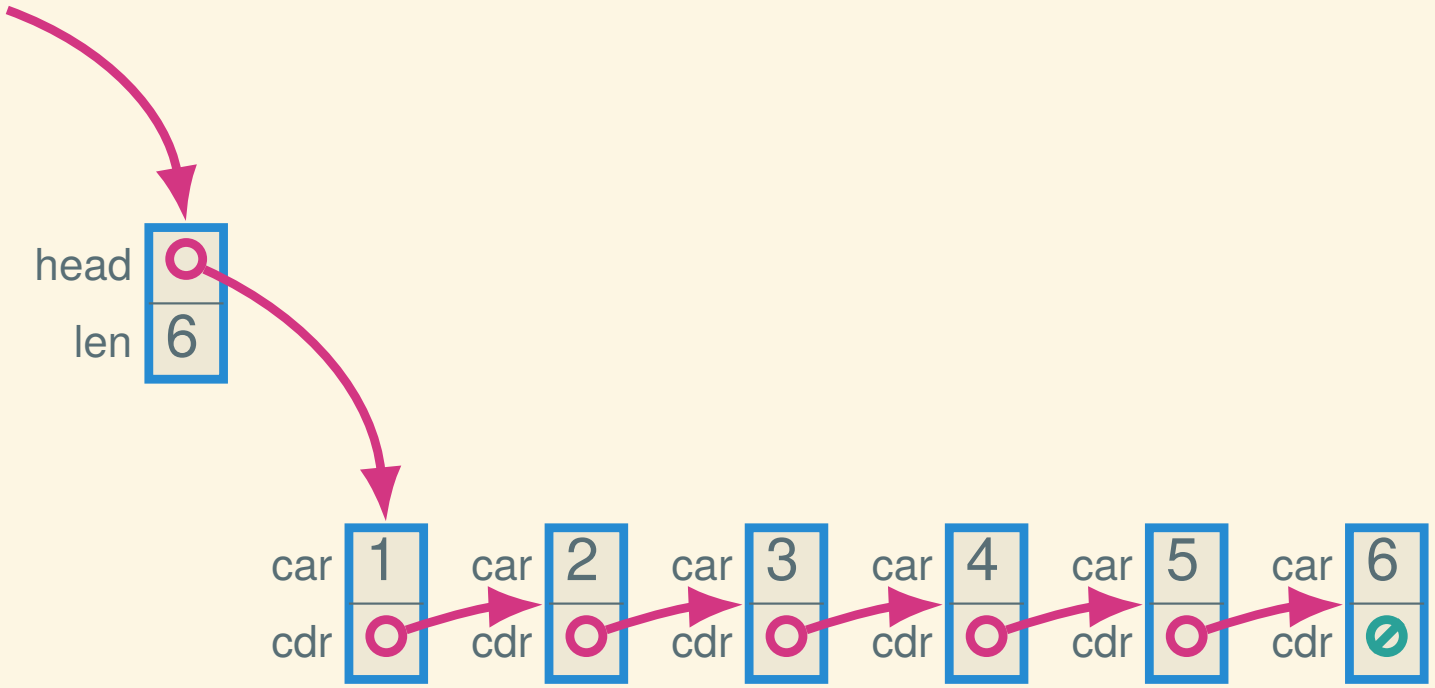
```
class LL:
    …

    def _find_nth_node(self, n):
        let curr = self.head
        while n > 0:
            if nil?(curr): error('list too short')
            n    = n - 1
            curr = curr.next
        curr

    def get_nth(self, n):
        self._find_nth_node(n).data

    def set_nth(self, n, val):
        self._find_nth_node(n).data = val
```

What else might we want to do?

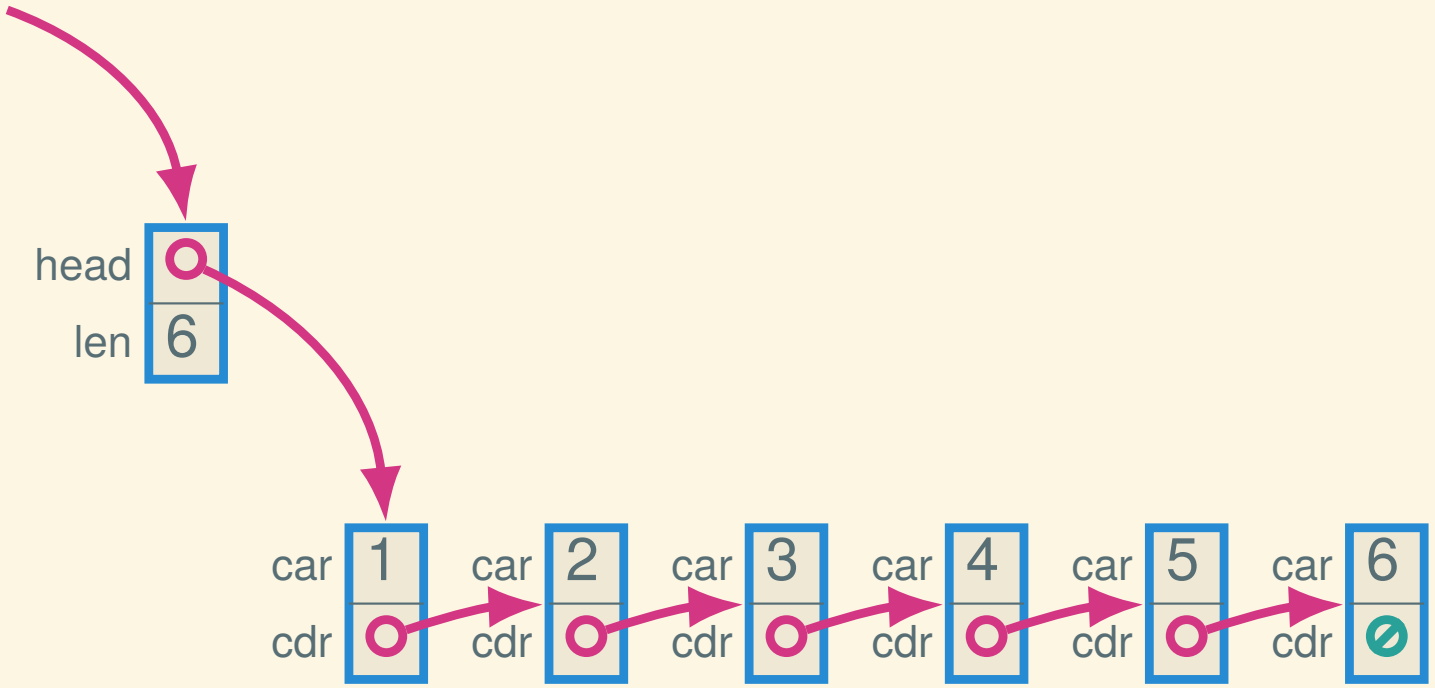
What else might we want to do?

- Insert or remove at the given position or the end.
- Split a list in two or splice two into one.
- Know how long the list is without counting.

Keeping the length

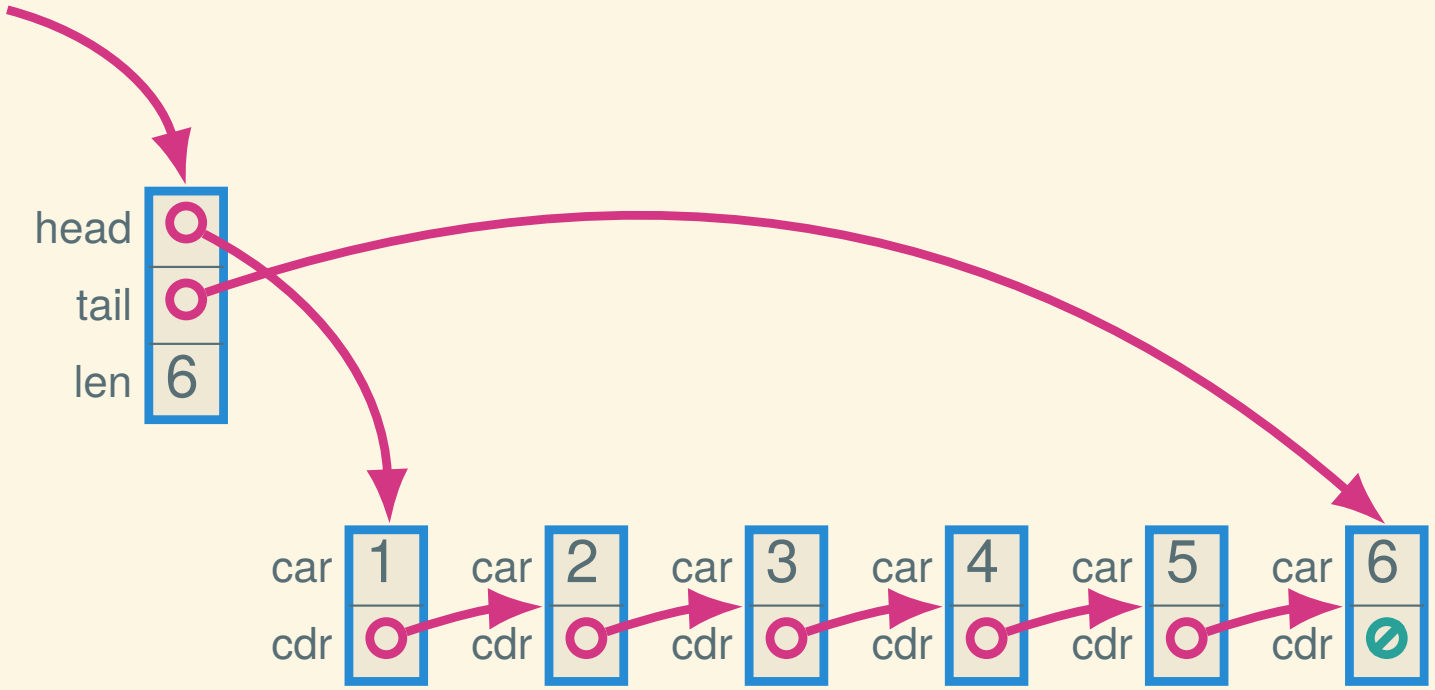


Keeping the length

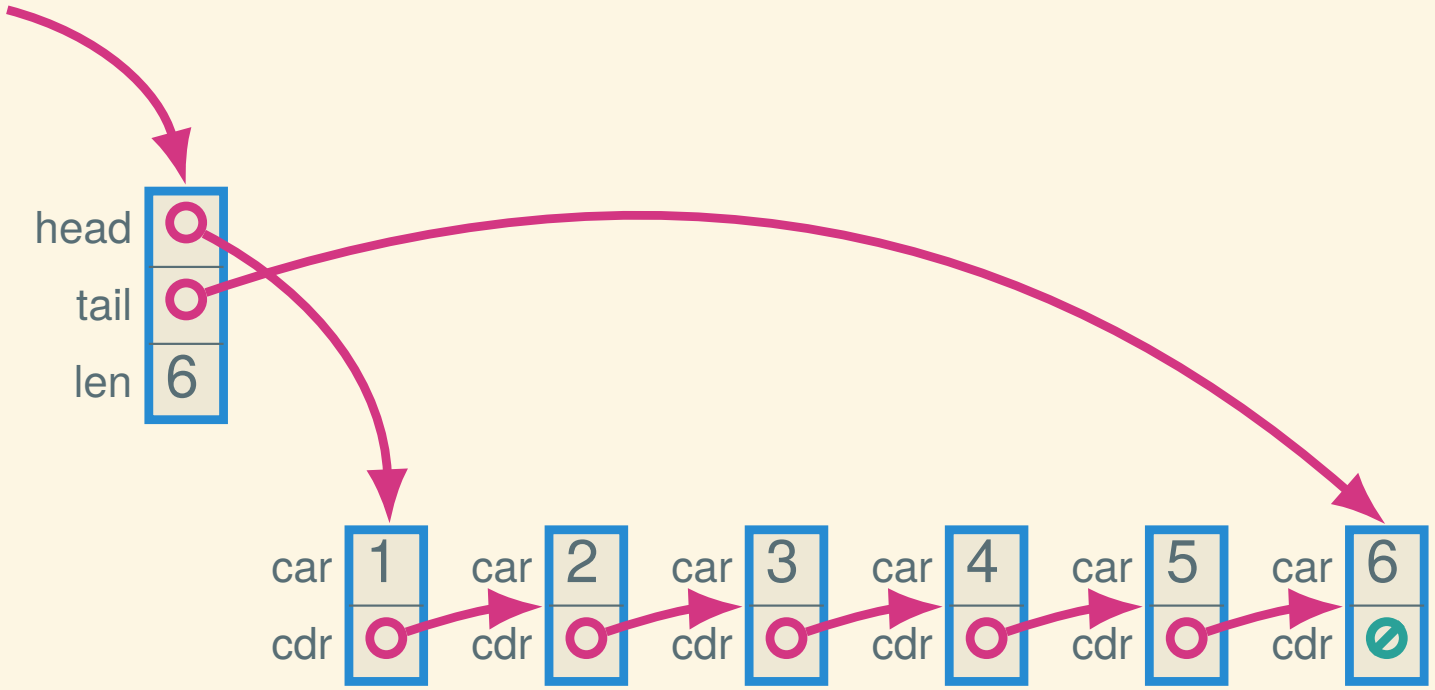


How can we make sure the `len` field is always right?

Quick access to the tail

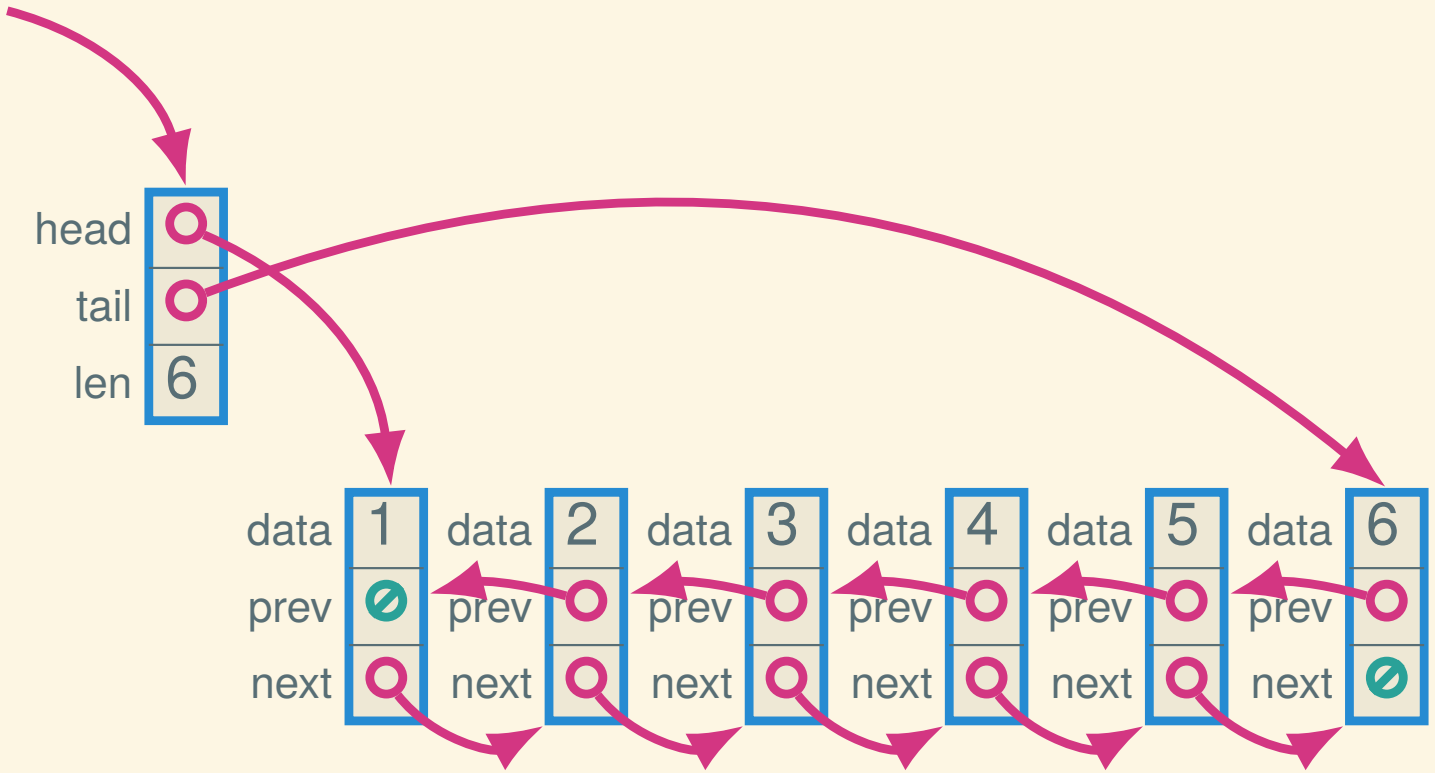


Quick access to the tail

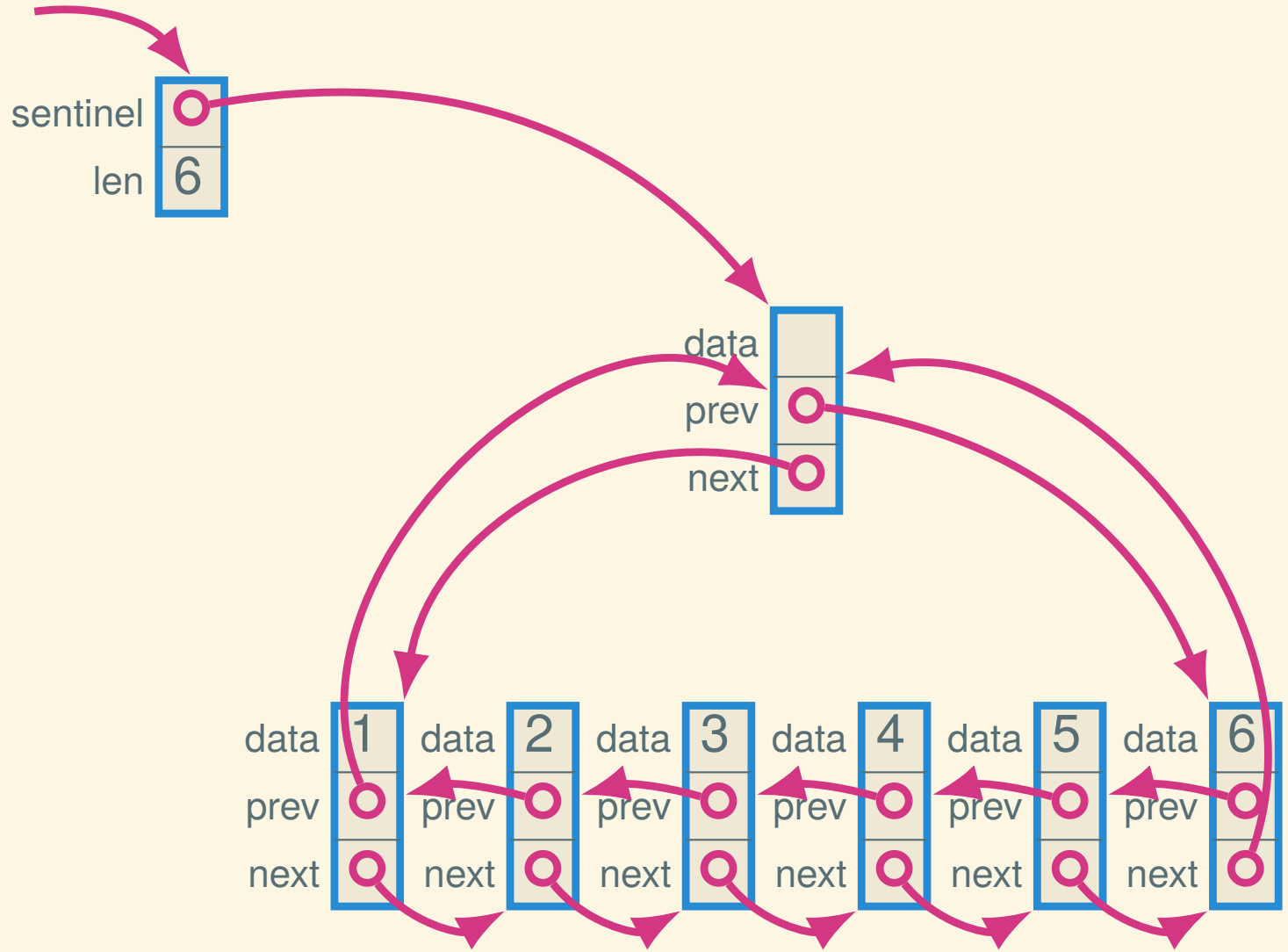


Which operations are simple now? Which are still more work?

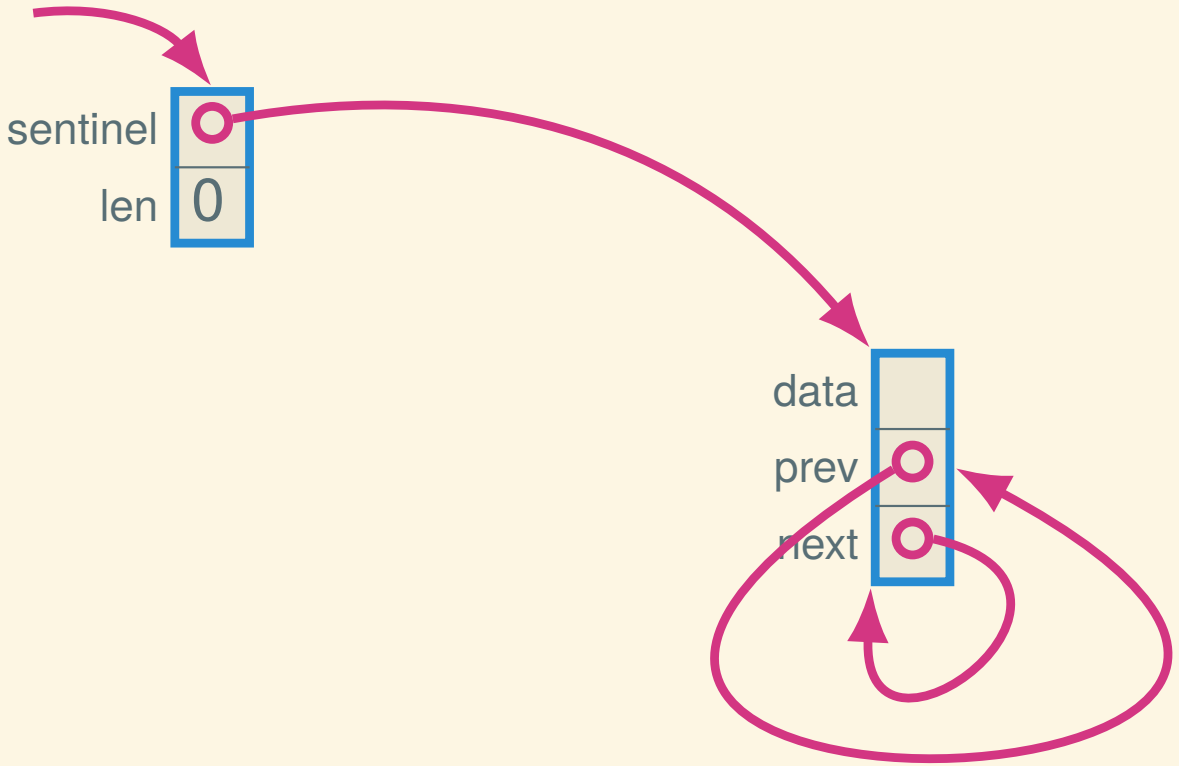
Doubly-linked



Circular, doubly-linked with sentinel



Empty (circular, doubly-linked w/sentinel)



Next time: asymptotic complexity