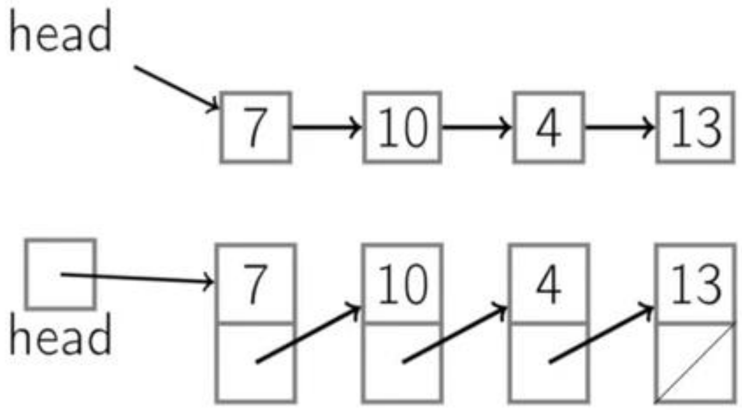
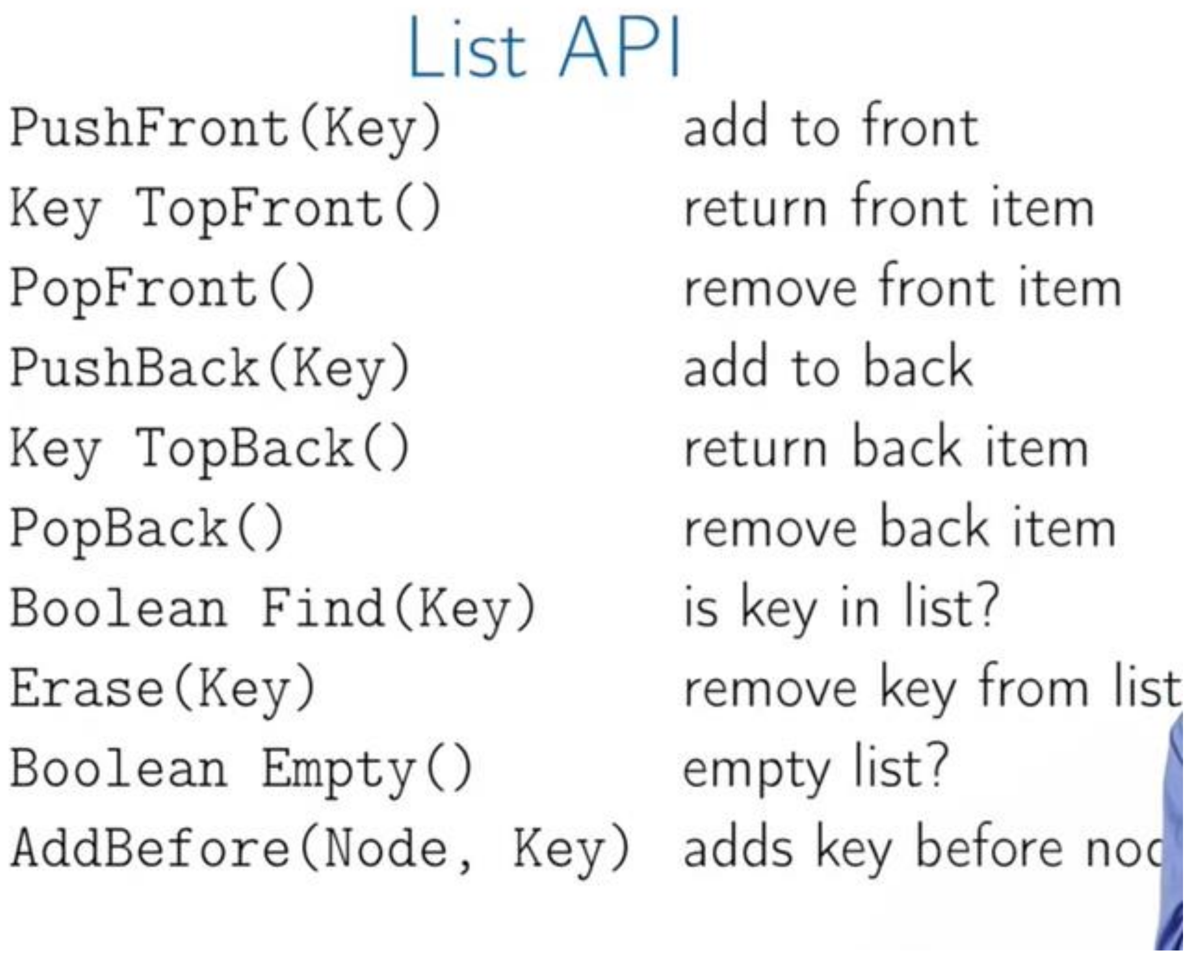
Singly-Linked Lists

Singly-Linked List



Node contains:

- key
- next pointer



List API

`PushFront(Key)`	add to front
`Key TopFront()`	return front item
`PopFront()`	remove front item
`PushBack(Key)`	add to back
`Key TopBack()`	return back item
`PopBack()`	remove back item
`Boolean Find(Key)`	is key in list?
`Erase(Key)`	remove key from list
`Boolean Empty()`	empty list?
`AddBefore(Node, Key)`	adds key before noc

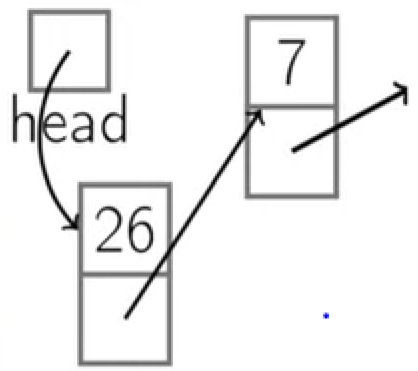
Time Complexity Analysis of Singly Linked List

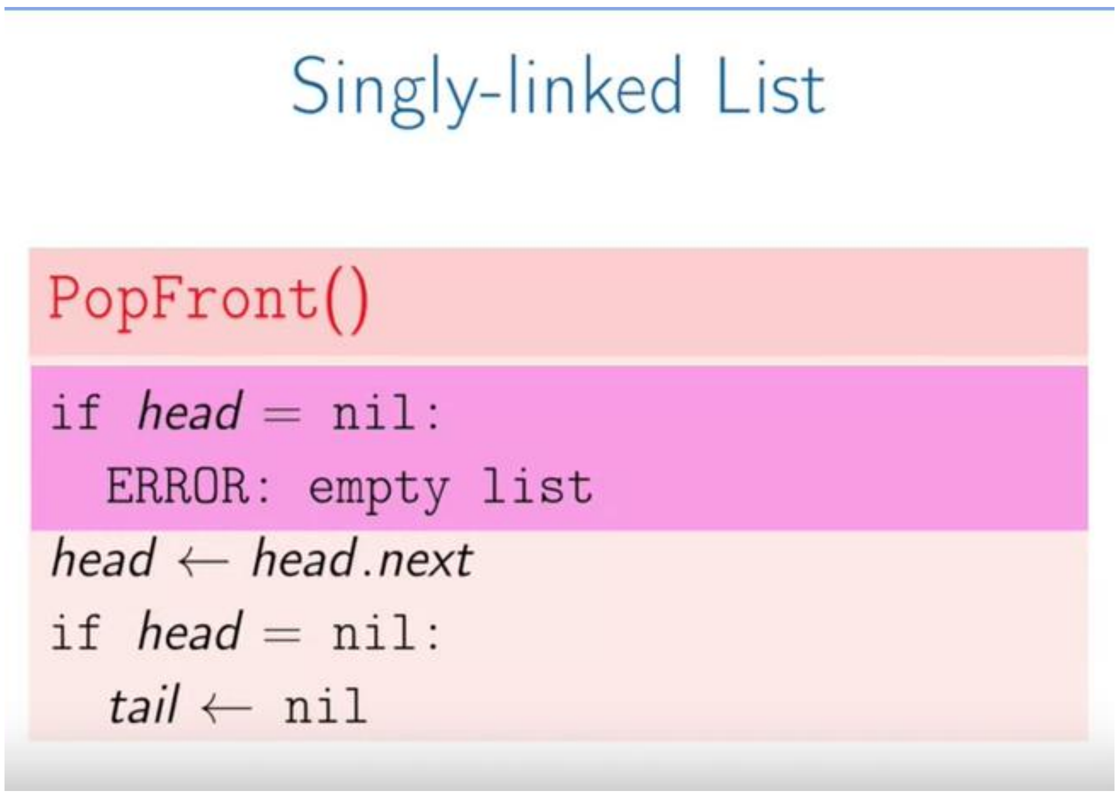
- PushFront O(1)
- PopFront O(1)
- PushBack (no tail) O(n)
- PopBack (no tail) O(n)
- PushBack (with tail) O(1)
- PopBack (with tail) O(n)

Singly-linked List

PushFront(*key*)

```
node ← new node
node.key ← key
node.next ← head
head ← node
if tail = nil:
    tail ← head
```

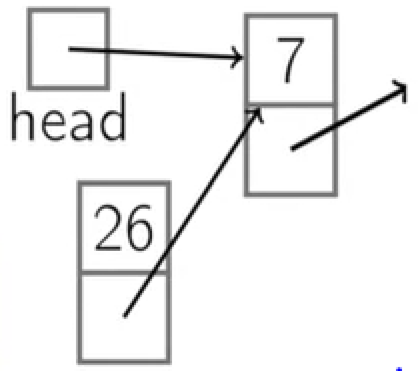




Singly-linked List

PopFront()

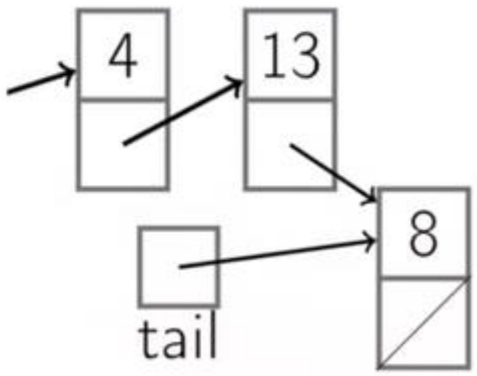
```
if $head$ = nil:
    ERROR: empty list
$head \leftarrow head.next$
if $head$ = nil:
    $tail \leftarrow$ nil
```



Singly-linked List

PushBack(*key*)

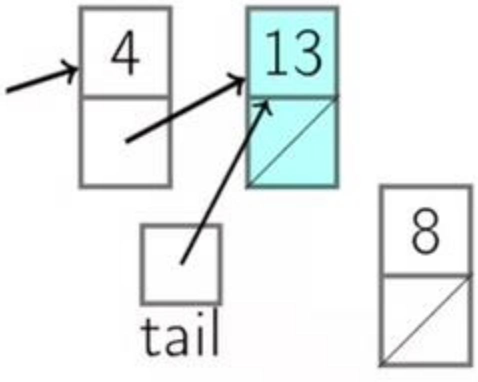
```
$node \leftarrow$ new node
$node.key \leftarrow key$
$node.next =$ nil
if $tail =$ nil:
    $head \leftarrow tail \leftarrow node$
else:
    $tail.next \leftarrow node$
    $tail \leftarrow node$
```



Singly-linked List

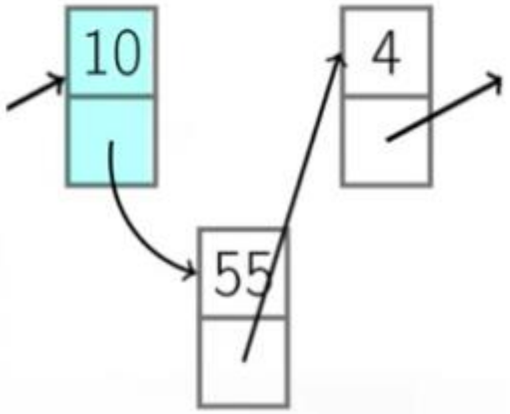
PopBack()

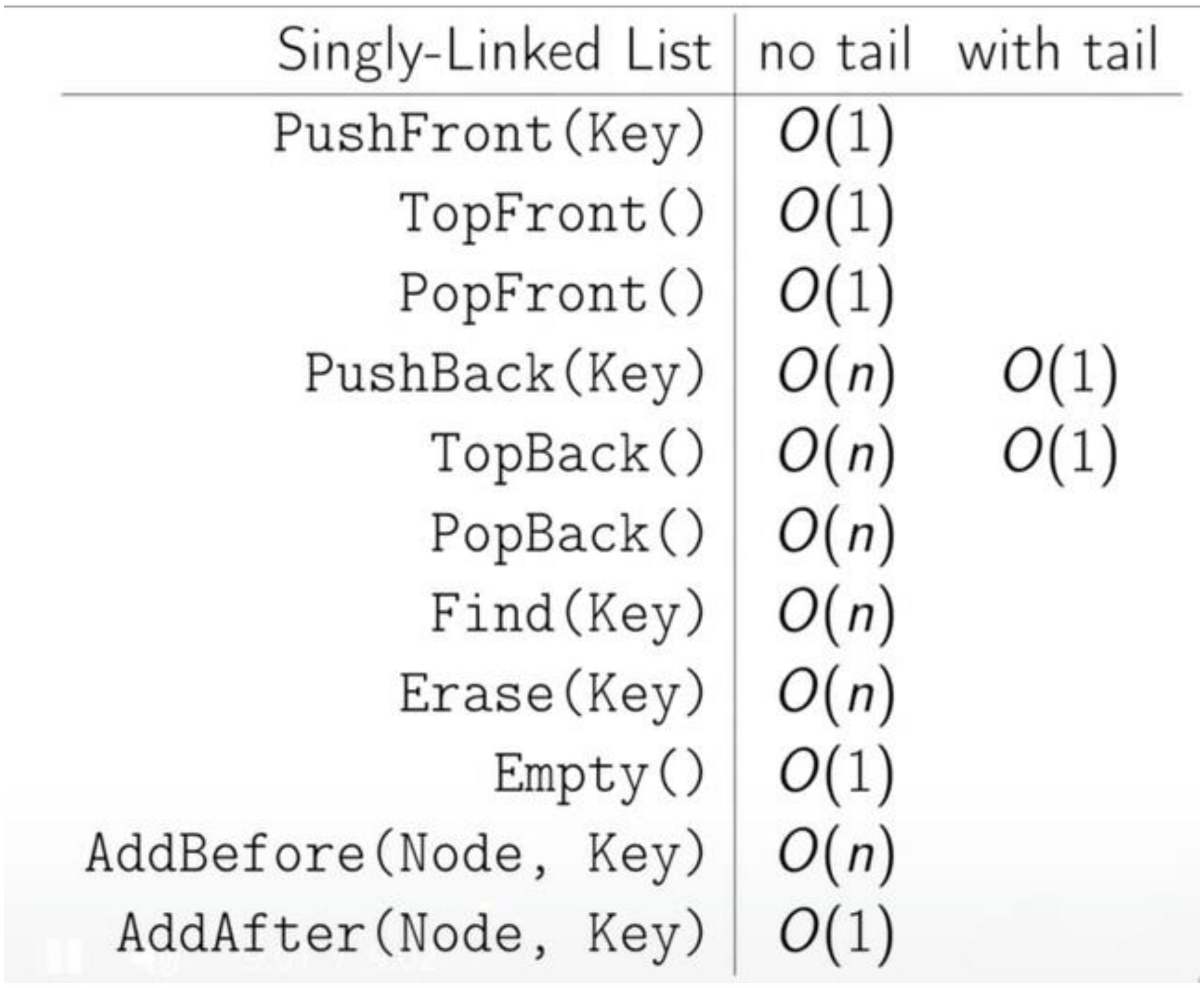
```
if $head$ = nil:  ERROR: empty list
if $head = tail$:
  $head \leftarrow tail \leftarrow$nil
else:
  $p \leftarrow head$
  while $p.next.next \neq$ nil:
    $p \leftarrow p.next$
  $p.next \leftarrow$ nil; $tail \leftarrow p$
```



Singly-linked List

```
AddAfter($node$, $key$)
$node2 \leftarrow$ new node
$node2.key \leftarrow key$
$node2.next = node.next$
$node.next = node2$
if $tail = node$:
    $tail \leftarrow node2$
```





Singly-Linked List	no tail	with tail
`PushFront(Key)`	$O(1)$	
`TopFront()`	$O(1)$	
`PopFront()`	$O(1)$	
`PushBack(Key)`	$O(n)$	$O(1)$
`TopBack()`	$O(n)$	$O(1)$
`PopBack()`	$O(n)$	
`Find(Key)`	$O(n)$	
`Erase(Key)`	$O(n)$	
`Empty()`	$O(1)$	
`AddBefore(Node, Key)`	$O(n)$	
`AddAfter(Node, Key)`	$O(1)$	