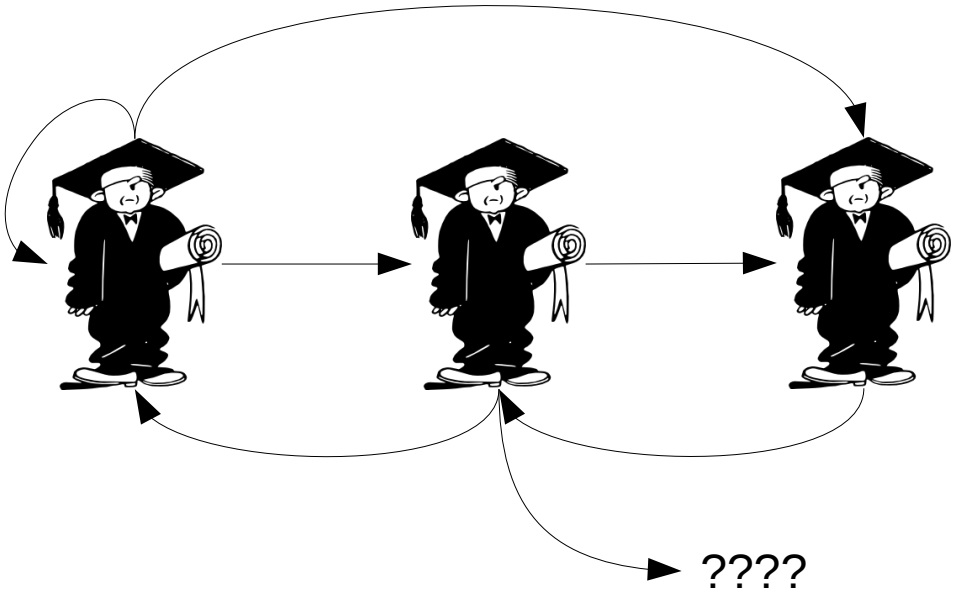
CS240
Fall 2014

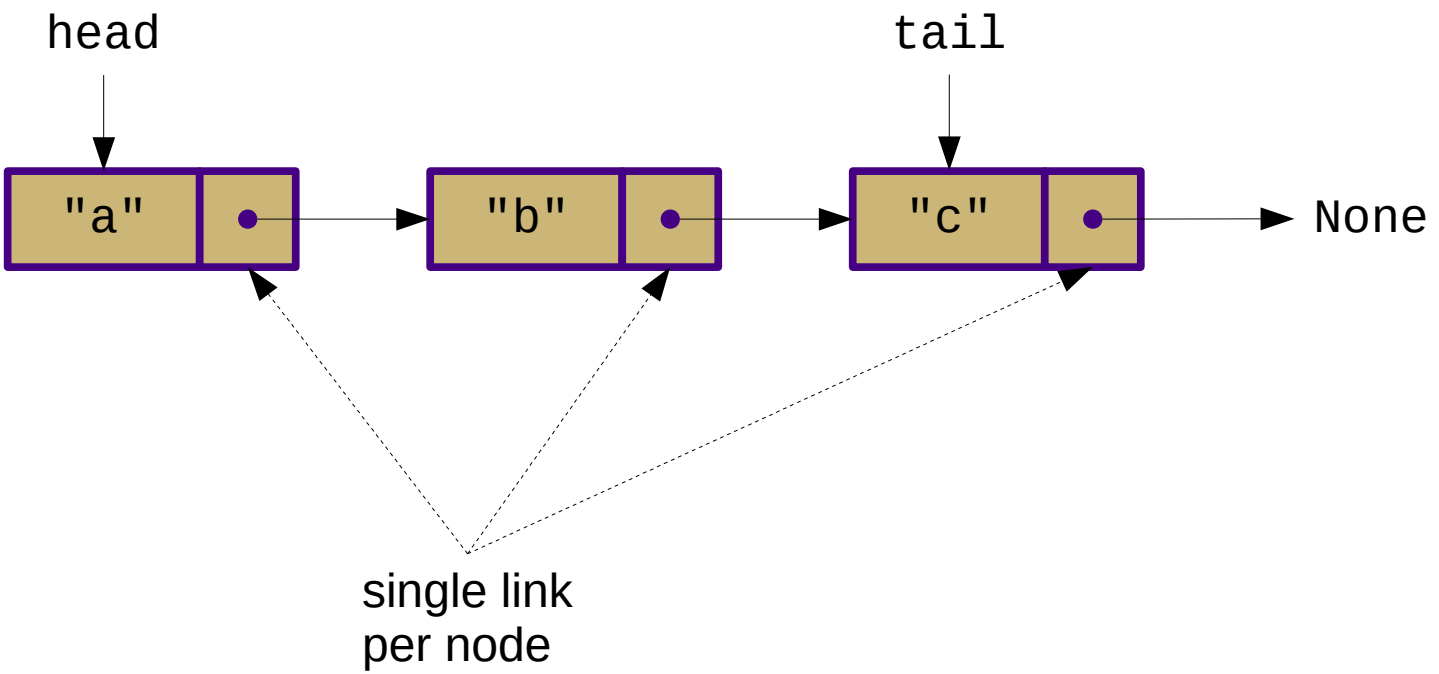
Mike Lam, Professor



Advanced Linked Lists

Review: Singly-Linked Lists

- Singly-linked list:



Circularly-Linked Lists

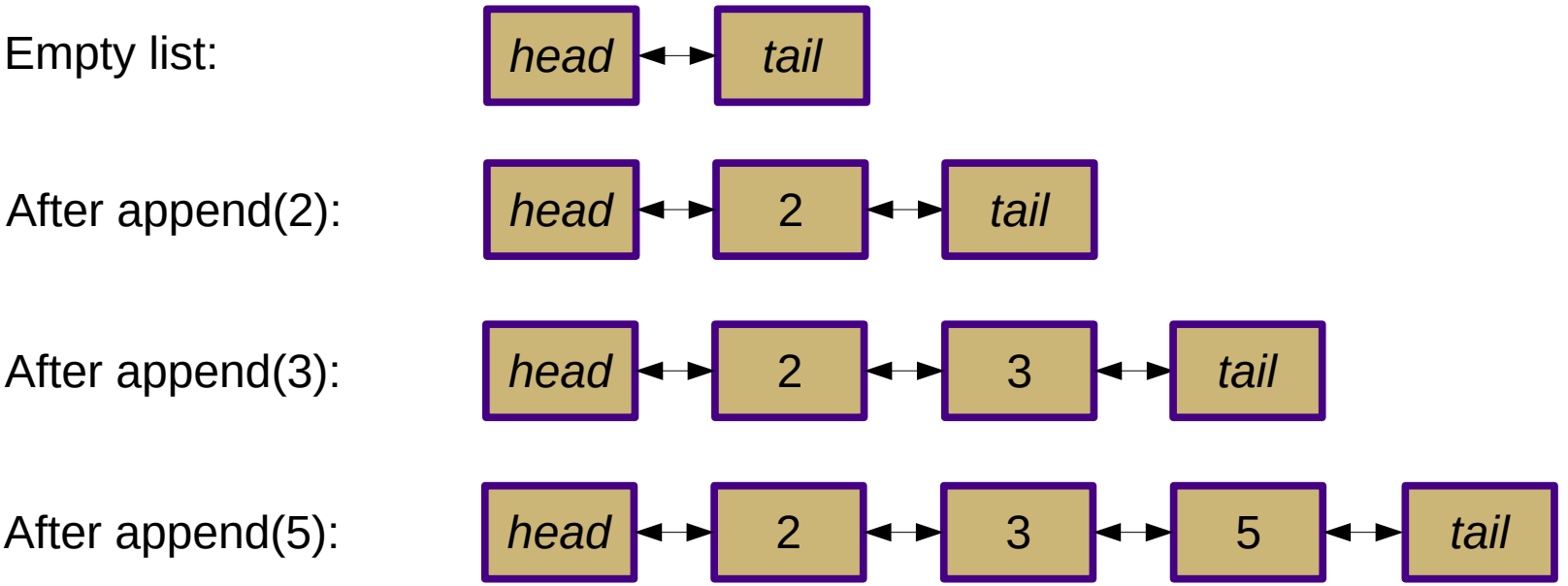
- Keep a single node reference
- Useful for round-robin scheduling
  - New operation: `rotate()`
- Can be used to implement regular list
  - No need to track both head and tail
  - `head = tail.next`

Doubly-Linked Lists

- Two references: `prev` and `next`
  - To predecessor and successor nodes
- Allows insert and remove at both ends
  - Can now implement stacks, queues, and deques

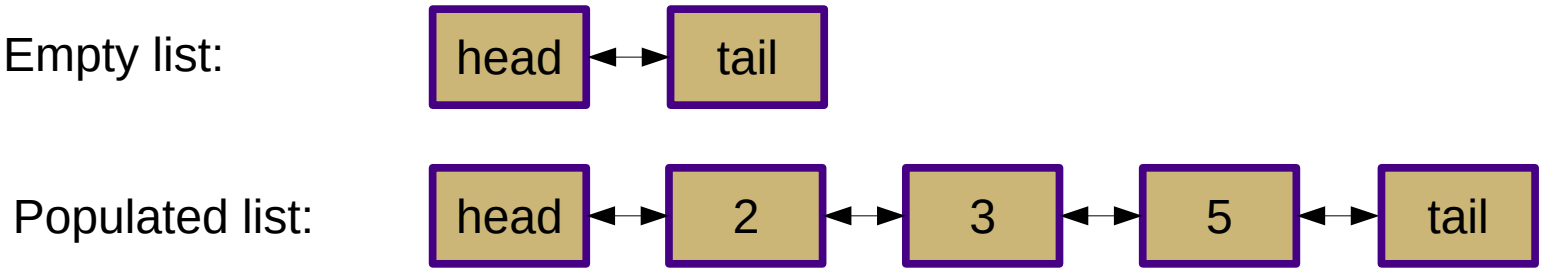
Sentinels

- Placeholder (“fake”) nodes at head and/or tail



Sentinels

- Simplifies logic of insertion and removal




```
def append(self, e):
    new_node = Node(e)
    new_node.prev = self._tail
    new_node.next = None
    if self.is_empty():
        self._head = new_node
        self._tail = new_node
    else:
        self._tail.next = new_node
    self._tail = new_node
```

```
def append(self, e):
    new_node = Node(e)
    new_node.prev = self._tail.prev
    new_node.next = self._tail
    self._tail.prev.next = new_node
    self._tail.prev = new_node
```

Deques

- Double-ended queue
- Two sets of insert/remove methods:
  - `insert_first` and `delete_first`
  - `insert_last` and `delete_last`
- Implementation using doubly-linked list w/ sentinels

Tradeoffs

- Advantages of Arrays
  - O(1) access to elements by index
  - Proportionally fewer actual operations
    - Calculation and dereference vs. memory allocation and reference re-arranging
  - Proportionally less memory usage
    - Both arrays and linked lists can be referential
    - Arrays require at most 2n space overhead, while linked lists are at least 2n (or 3n for doubly-linked lists)

Tradeoffs

- Advantages of linked lists
  - Worst-case O(1) bounds
    - No amortized bounds
  - O(1) insertions and removals at arbitrary positions
    - No need to shift elements
    - This is a HUGE advantage!