A02 List Implementations

An array-based implementation
A linked-list-based implementation

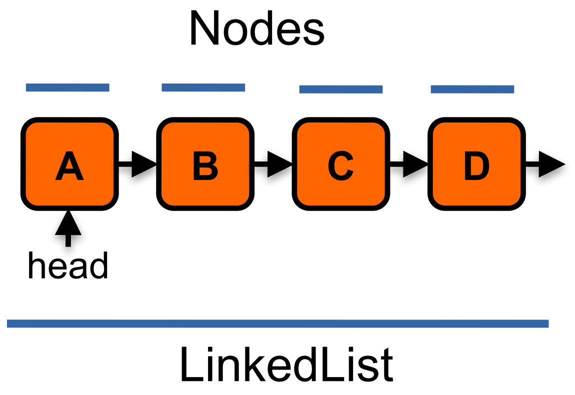
ANU – School of Computing – Structured Programming 1110 / 1140 / 6710

List Implementation Options

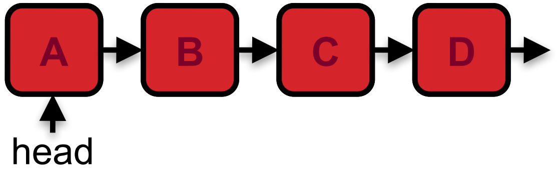
- Arrays
  - Fast lookup of any element
  - A little messy to grow and contract
- Linked list
  - Logical fit to a list, easy to grow, contract
  - Need to traverse list to access elements

Linked Lists: Singly Linked List

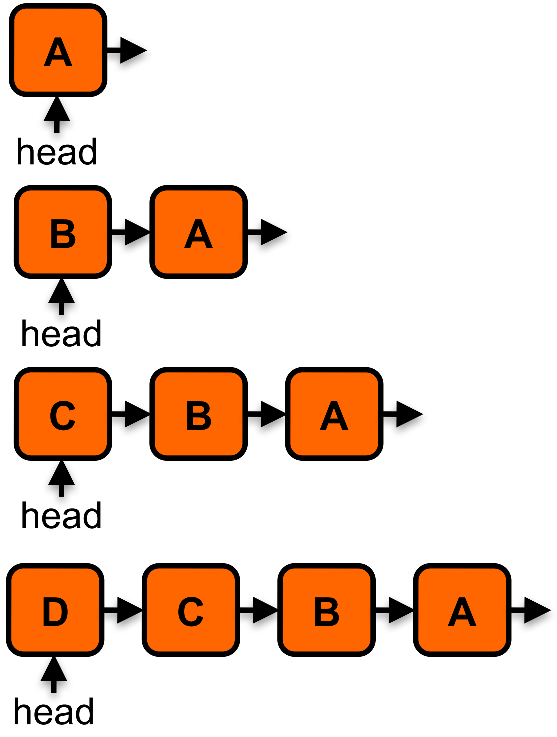
```
public class LinkedList<T> {
  private class Node<T> {
    T value;
    Node<T> next;
  }
  Node<T> head;
}
```



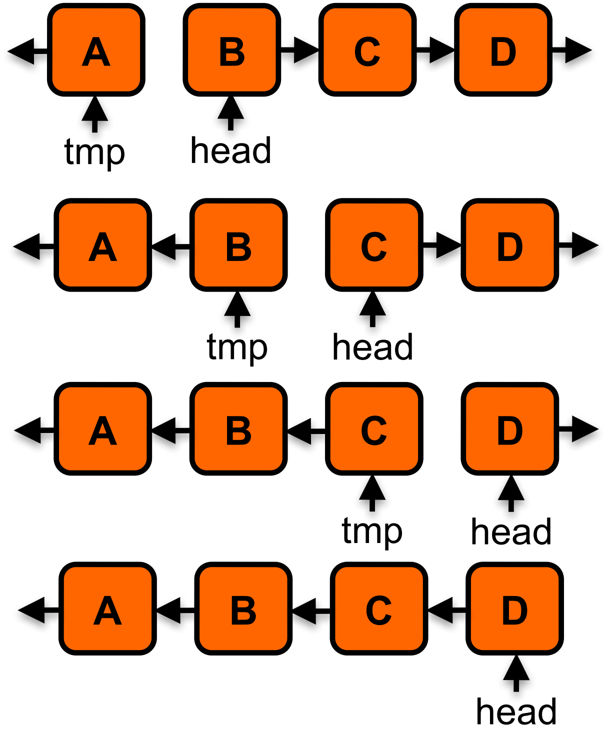
Linked List Reversal: Two Approaches



- Add each item to start:



- Pointer reversal:



Complexity

```
void add(T value);
T get(int index);
int size();
T remove(int index);
void reverse();
```

ArrayList

- `add` – **Time O(1) amortized, O(n) worst**
- `get` – **Time O(1)**
- `size` – **Time O(1)**
- `remove` – **Time O(n)**
- `reverse` – **Time O(n)**

Space O(n)

LinkedList

- `add` – **Time O(1)**
  - if explicitly tracking last node
- `get` – **Time O(n)**
- `size` – **Time O(1)**
  - if explicitly tracked
- `remove` – **Time O(n)**
- `reverse` – **Time O(n)**

Space O(n)