COSC160: Data Structures
Linked Lists

Jeremy Bolton, PhD
Assistant Teaching Professor

GEORGETOWN UNIVERSITY

Outline

I. Data Structure (lower-level) Implementations
   I. Chaining (Linked Lists)
      I. Design Decisions and Implications on Complexity
         I. Example: Tail Pointer
   II. Arrays
      I. Design Decisions and Implications on Complexity

II. Design Decisions
   I. Arrays vs. Chaining

III. Interesting Applications
   I. Polynomial Arithmetic and Evaluation

GEORGETOWN UNIVERSITY

Linear Structures

- Linear structures are generally implemented using arrays or chaining
  - Contiguous vs. non-contiguous in memory
    - Static size vs. dynamic size
  - When designing a data structure, one must consider these low-level designs
    - One major decision: contiguous vs non-contiguous (ie array vs chain)
    - This design decision will have implications with respect to usage and efficiency
  - There are many design decisions that a programmer must consider
    - Some are standard, others may be tailor-made for the problem/solution on-hand.

List Class Implemented by Chaining (Linked Lists)

- Design Decisions
  - Single Link, next
  - Head pointer maintained

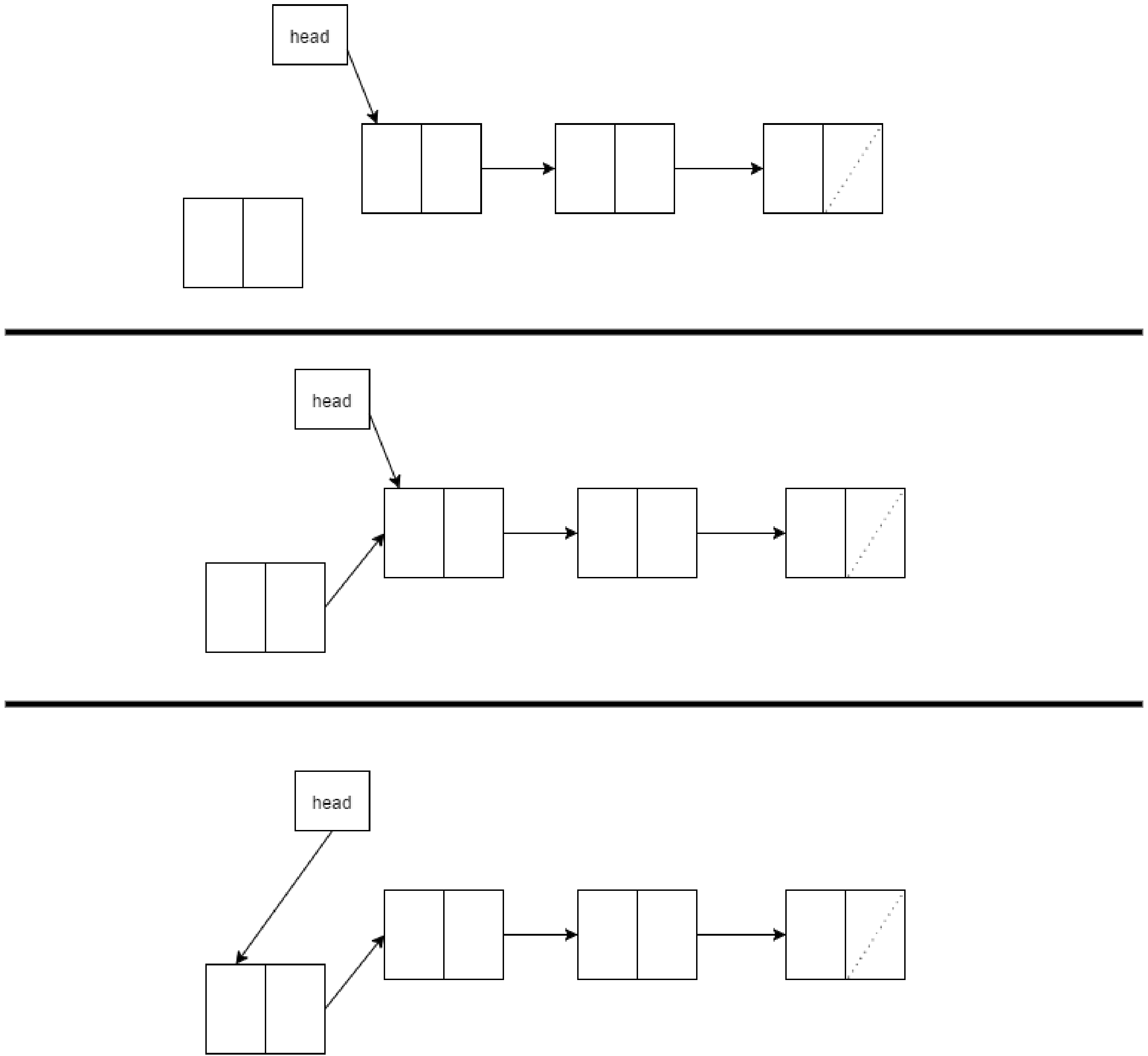
- Standard Operations
    - Insert(item, index)
    - Remove(index)
    - Retrieve(index)

Inserting into a Linked List

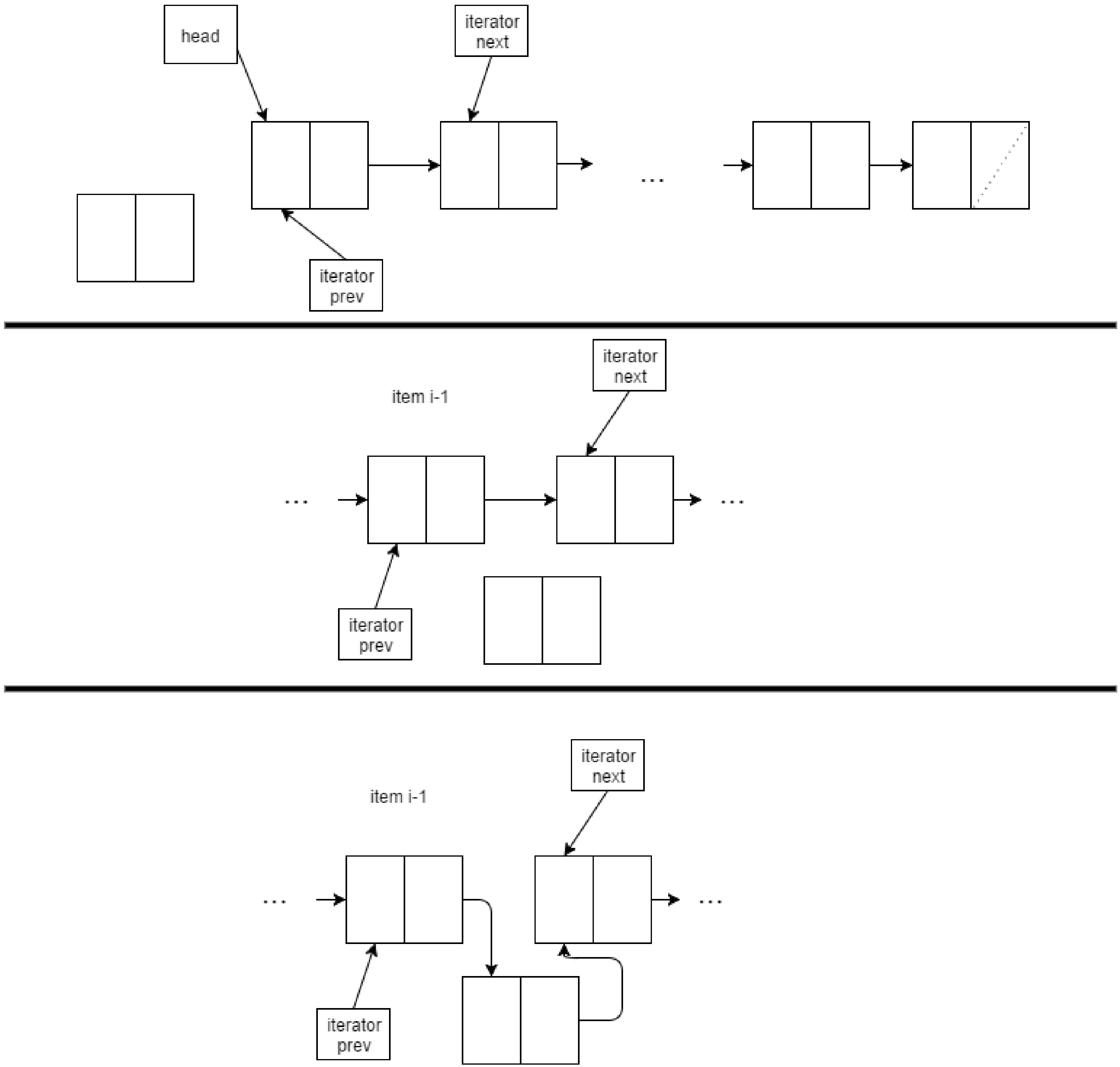
- Lets investigate three cases of the insertion operation
  - Insert to front of list
  - Insert to middle of list
  - Insert to end of list

Inserting into front of a Linked List

- Insert to front of list
- Approximately how many computational steps?
- Does the number of computational steps depend on the size of the list?
- Time complexity?



Inserting into middle of Linked List



Approximately how many computational steps?

Does the number of computational steps depend on the size of the list?

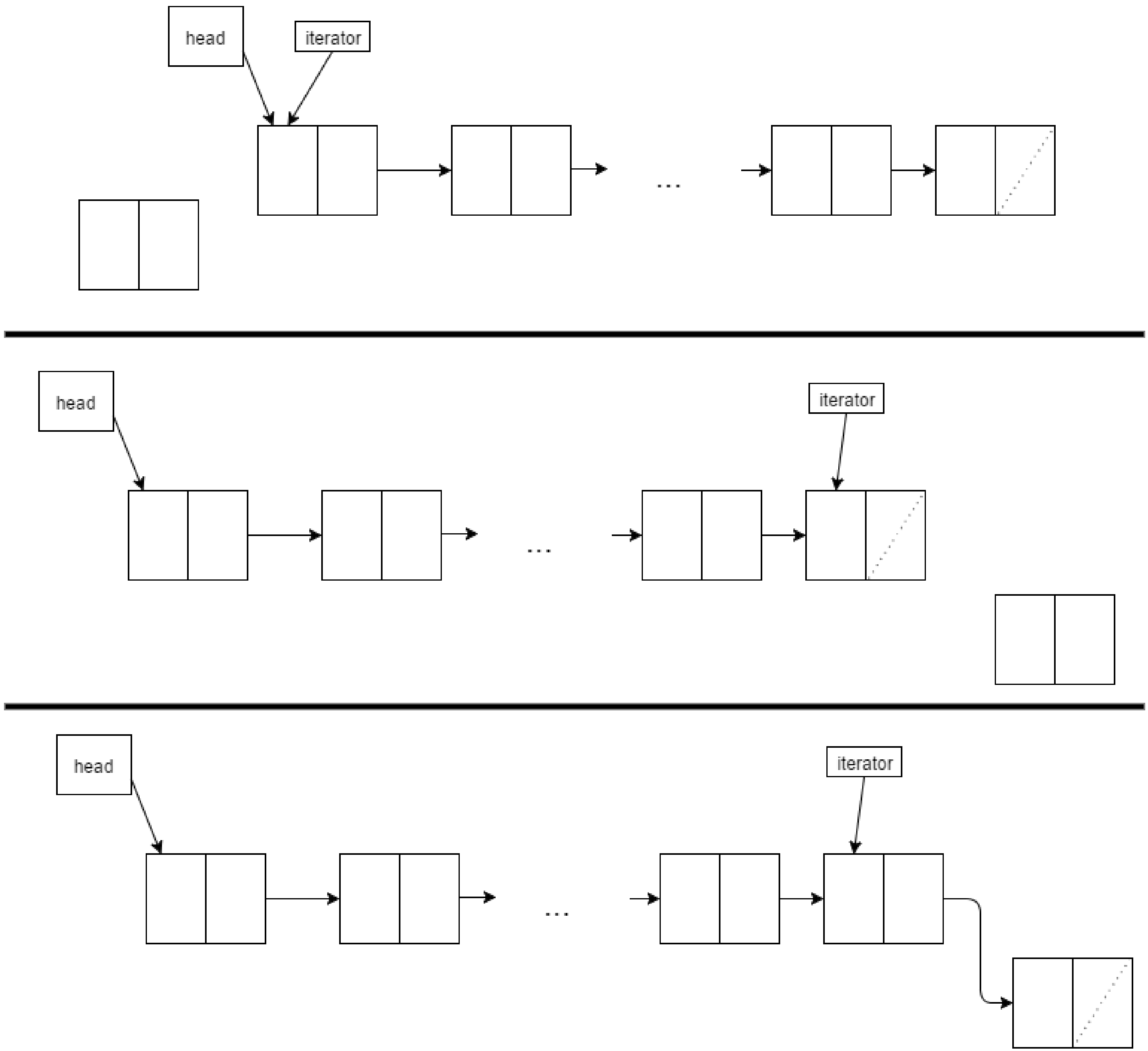
Time complexity?

Inserting to end of Linked List

- Approximately how many computational steps?

- Does the number of computational steps depend on the size of the list?

- Time complexity?



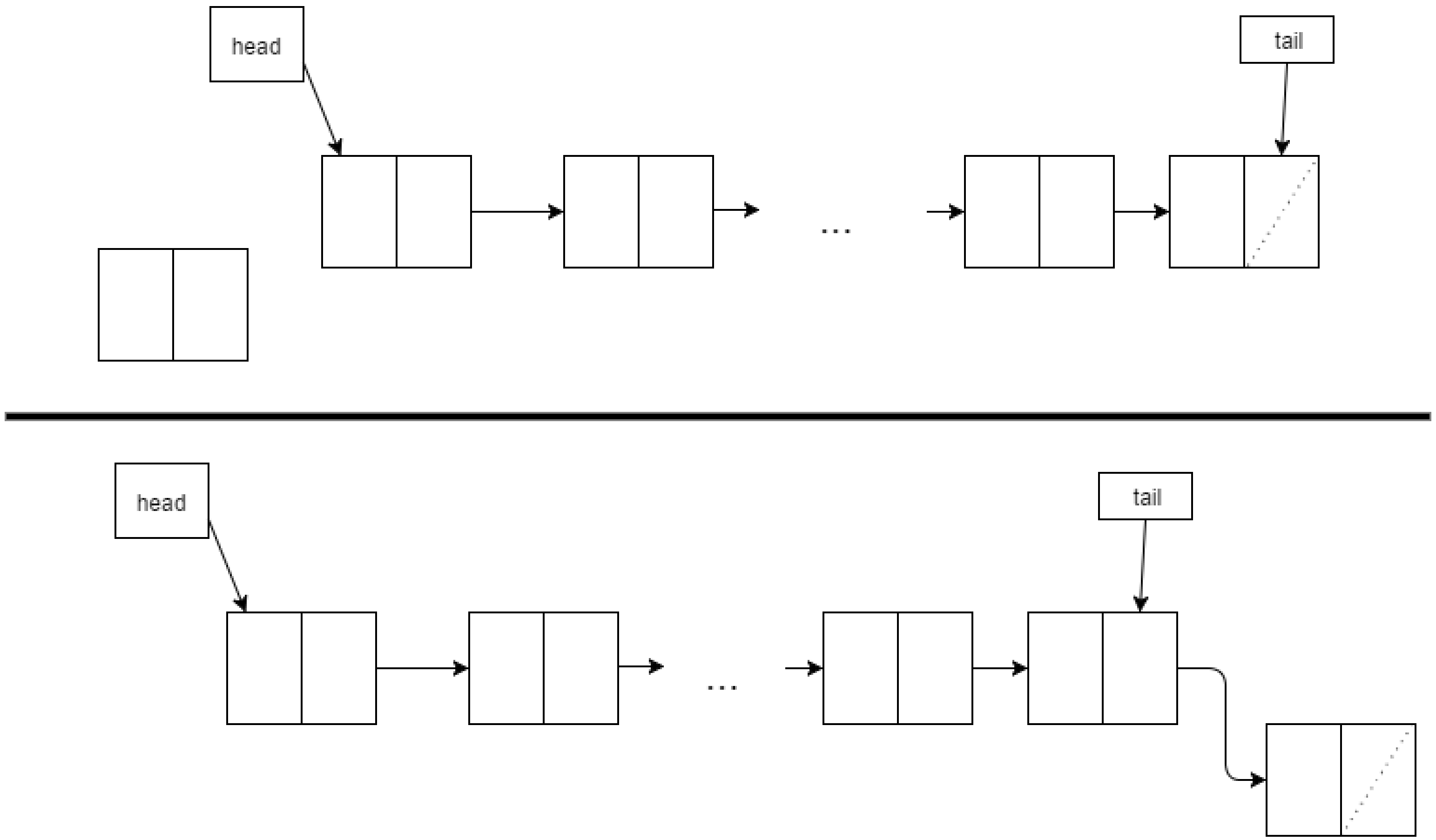
Design Decisions

- One major goal of a data structure is to provide an efficient way to manage data.

- Can we improve the efficiency of any of these operations -- YES

Insert to End (List with Tail Pointer)

- Significant time reduction.
- What is the time complexity now?



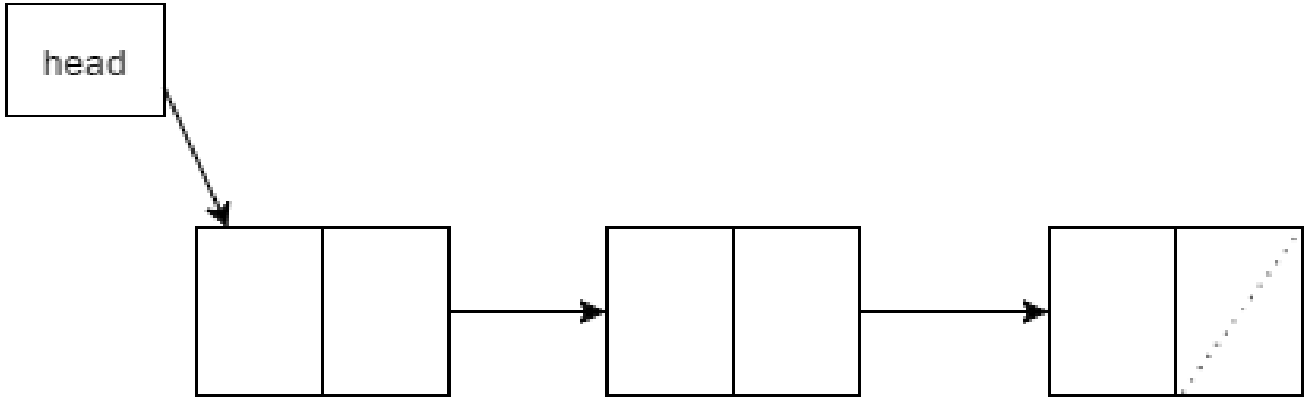
GEORGETOWN UNIVERSITY

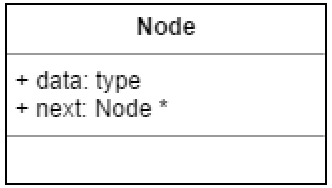
List Variations: Singly Linked vs. Doubly Linked Lists

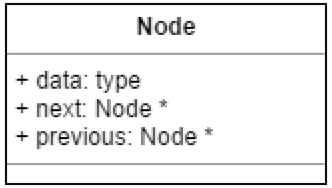
- Differences
  - Doubly Linked List
    - Can traverse a list in both directions at added cost of 1 pointer per node
    - Slightly simplify iteration scheme for insertions and removals

Singly Linked vs. Doubly Linked Lists

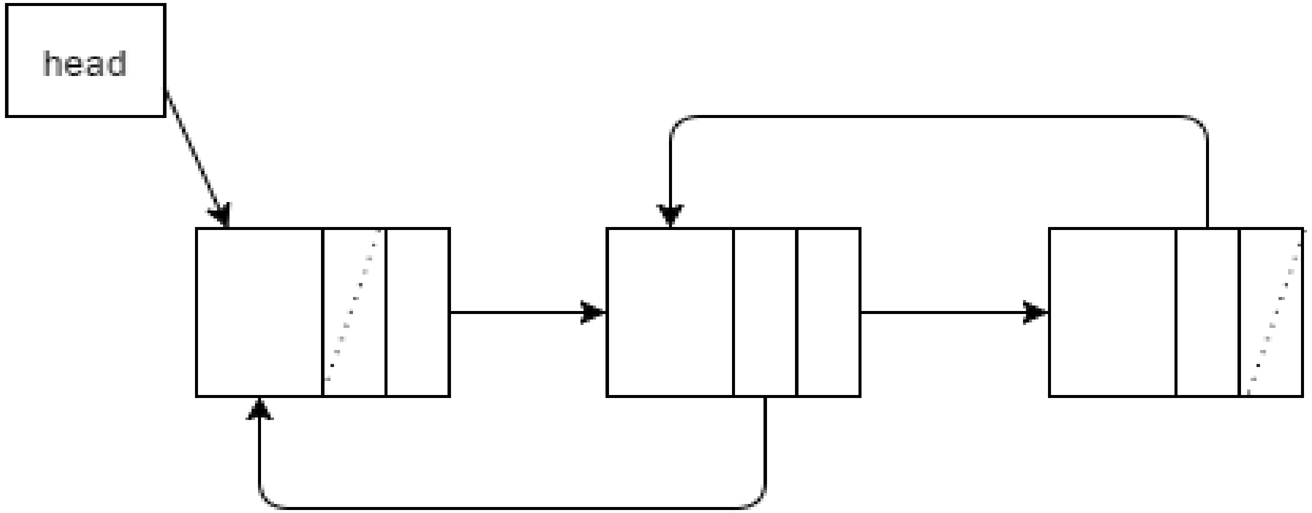
- Singly Linked List







- Doubly Linked List



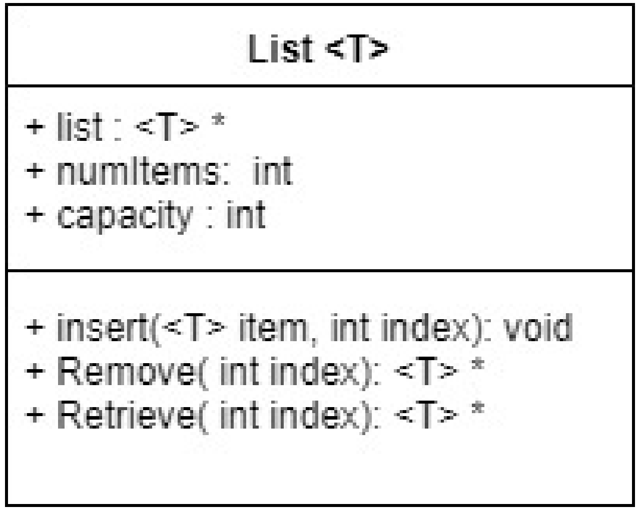
Why a double link?

- May be useful in specific situations.
  - ***Remove last node***

- A prev pointer, may reduce retrieval and insertion times at the cost (space) of 1 extra pointer per node.
  - Create an efficient algorithm**:** **Node*** get(int index)

List Class Implemented as Array

- Assume the following design decisions
  - Array initialized to “large” capacity, chosen appropriately based on application
  - No gaps in the list (items shift after a removal)
  - Assumes index = 0 is beginning of list
  - Should maintain index for end (size) of list or numItems
  - Operations
    - Insert(item, index)
    - Remove(index)
    - Retrieve(index)
  - Concerns: What happens if insertion exceeds capacity?

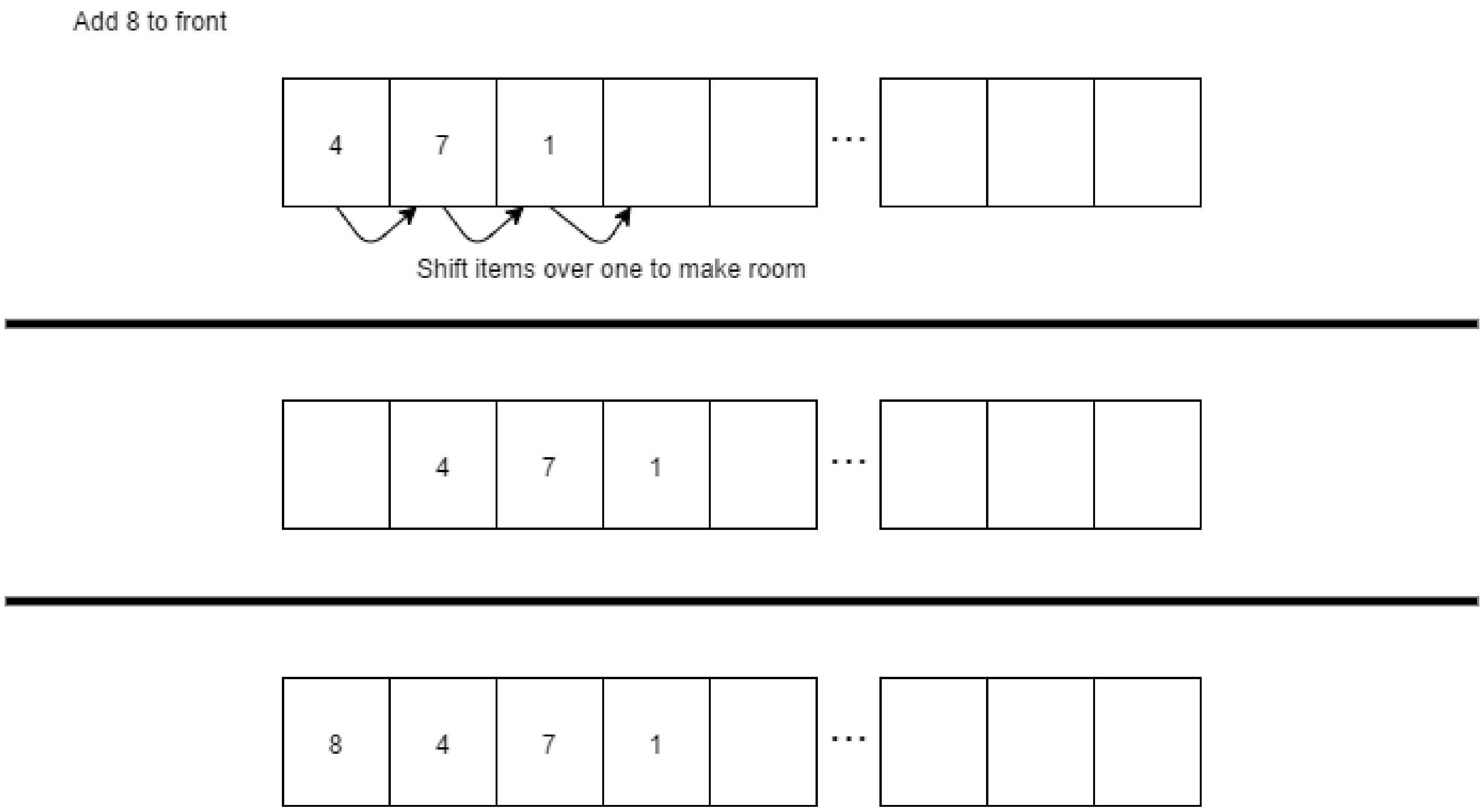


Inserting into an Array

- Lets investigate three cases of the insertion operation
  - Insert to front of array
  - Insert to middle of array
  - Insert to end of array

Insertion to Front of Array

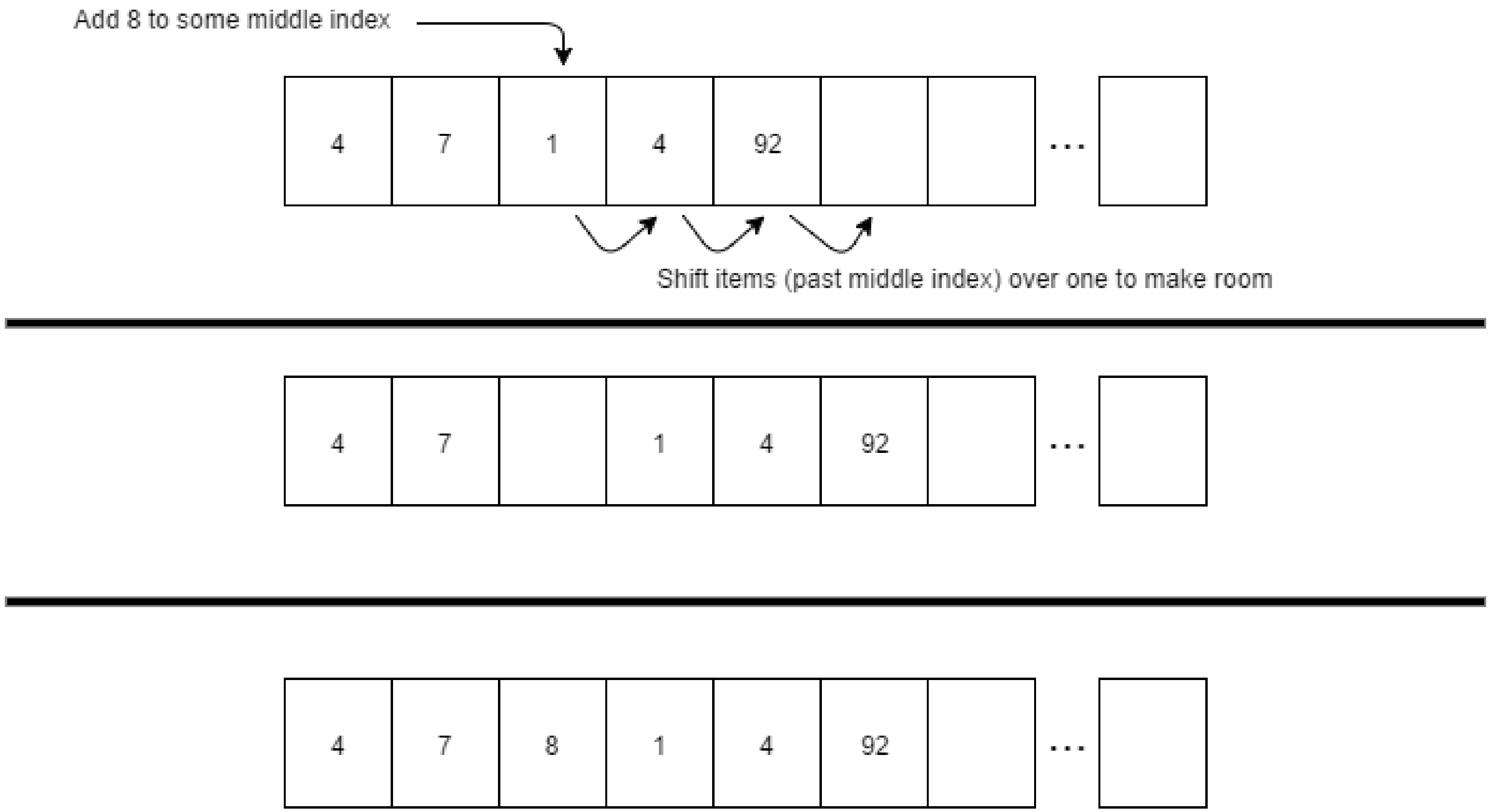
- Approximately how many computational steps?
- Does the number of computational steps depend on the size of the list?
- Time complexity?



GEORGETOWN UNIVERSITY

Insert to Middle of Array

- Approximately how many computational steps?
- Does the number of computational steps depend on the size of the list?
- Time complexity?



Insert to End of Array

- Approximately how many computational steps?

- Does the number of computational steps depend on the size of the list? Note: Assumes there is enough allocated space.

- Time complexity?

GEORGETOWN UNIVERSITY

"Worst" Case Time Complexity for a List Class

- Exercise: try to fill out table for best case and average case.

IMPLEMENTATION		CHAINING	ARRAYS
INSERT FRONT		$\Theta(1)$	$\Theta(n)$
INSERT MIDDLE		$\Theta(n)$	$\Theta(n)$
INSERT BACK		$\Theta(1)$*	$\Theta(1)$**
RETRIEVE FRONT		$\Theta(1)$	$\Theta(1)$
RETRIEVE MIDDLE		$\Theta(n)$	$\Theta(1)$
RETRIEVE BACK		$\Theta(1)$*	$\Theta(1)$
REMOVE FRONT		$\Theta(1)$	$\Theta(n)$
REMOVE MIDDLE		$\Theta(n)$	$\Theta(n)$
REMOVE BACK		$\Theta(1)$	$\Theta(1)$
* Assumes tail ptr			
** Assumes alloc			

Basic Lists: Array vs Linked List Implementation

- Example: Design a list class for use to store information for 250 tenants in apt. building.

- Design an aptList Class with the following operations
  - Insert(item, location): bool
  - Remove(item): bool
  - Find(item): int

- How would you implement this structure as a class?
  - Array or Chain? Why?
  - What member variables would you include? Why?

- We could keep the data in the list in some order …