CSSE 220

Data Structures + BIG-O Notation

Understanding the engineering trade-offs when storing data

Checkout *LinkedListSimple* project from public git

Checkout *SinglyLinkedList* homework from git

Data Structures

- Efficient ways to store data **based on how we'll use it**

- The main theme for the rest of the course

- So far we've seen `ArrayList`s
  - Fast addition **to end of list**
  - Fast access to any existing position
  - Slow inserts to and deletes from middle of list

Big-O Notation

- Describes the limiting behavior
  - How slow it can possibly run?
  - Describes the worst case
- Used for Classifying Algorithm Efficiency
- “O” for “Order”
  - O(n) → said as “Order n”
  - O(n^2) → said as “Order n-squared”

Big-O Notation (continued)

- Don’t Care About Constants
  - O(2n + 7) → O(n)
- Don’t Care About Smaller Powers
  - O(6n^2 + 7n) →O(n^2)
  - Algorithm grows asymptotically no faster than n^2
- If constant value, we say O(1) → “Order 1”
  - O(48) → O(1)

Q2

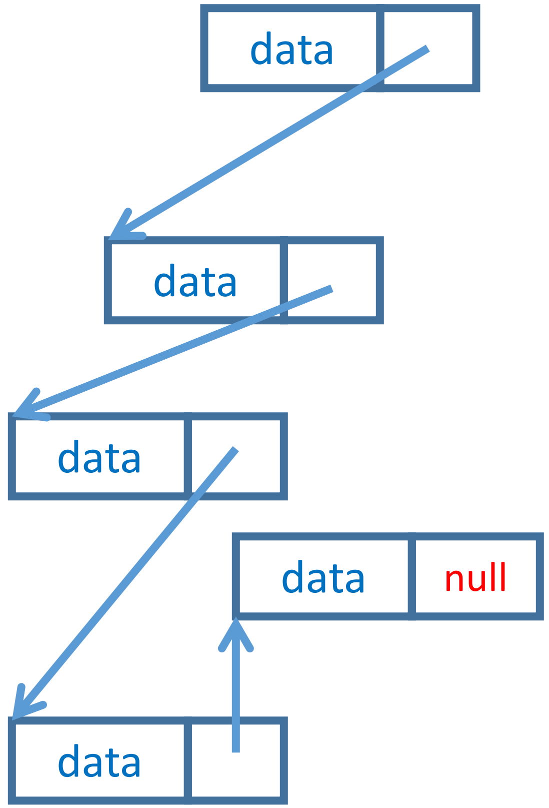
ArrayList Performance (Revisited)

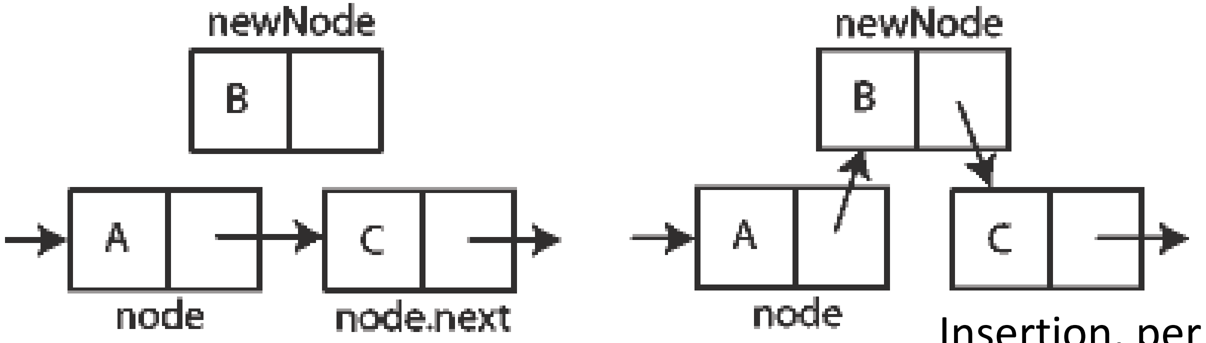
- Fast addition to **end of list**:
  - Fast access to any existing position – O(1) (like array)
  - Keep extra *capacity* for list growth
    - Fast access includes items in capacity not yet filled – O(1)
  - Capacity management is best left for CSSE230
- Slow inserts to and deletes from middle of list
  - Can get to insert/delete location quickly
  - For insert, shift all items right to accommodate -O(n)
  - For delete, shift all items left to fill gap – O(n)

Q3

Another List Data Structure

- What if we have to add/remove data from a list frequently?
- `LinkedLists` support this:
  - Fast insertion and removal of elements
    - Once we know where they go
  - Slow access to arbitrary elements





Insertion, per Wikipedia

Q4-5

LinkedList<E> Methods

- **void addFirst(E element)**
- **void addLast(E element)**
- **E getFirst()**
- **E getLast()**
- **E removeFirst()**
- **E removeLast()**

Complete Quiz

- Turn in quiz today

Homework

- SinglyLinkedList
  - Requires you to implement a SinglyLinkedList
  - Additional algorithm questions which make use of the SinglyLinkedList
  - Will give you remaining class time to work on it
  - If you complete it, work on the project!