Administrivia

- We're losing our TA Andrew 😊
  - Should get a new TA by next week, hopefully

Agenda

- Linked lists, cont'd.

Linked List structure

- Two basic objects:
  - The list “parent” itself
  - An “element” (book calls “link”), with data
  - Technically, we don’t need both
- Parent contains reference to the first element
- Each element contains a reference to the next element
- Last element’s “next” is set to null

Basic Linked List operations

- How to tell if empty?
- Insertions
  - insertFirst()
  - deleteFirst()
  - displayList()
  - insertLast()
- More complex operations
  - How to find an arbitrary element?
  - How to delete arbitrary element?

Double-ended list

- Contains pointer to last element
- Makes insertLast() much faster (how much?)

Linked list complexity?

- Similar to arrays
- O(1) insert/delete at beginning (or end of list for double-ended)
- Other operations take O(N), but faster than array if “sliding” is needed in array
- Memory?
  - Linked list more efficient, although it has to keep lots of references

Revisit abstraction

- Book finally covers abstraction here
- We can redo all of our previous data structures, previously array-backed, as linked list-backed
• *Interface* – high-level contract, while the dirty details are hidden
• How to do a stack?
• How to do a queue?
• You should read through this section

9 □ Next time…
• Finish Linked Lists
• Start Recursion