Hashing

Exam Logistics

- Monday, June 3, 12:30pm, KNE130 and KNE110
- In advance of the exam we will release a seating chart. Please show up to your assigned room and sit in your assigned seat.
  - Check Ed if you need a left-handed desk!
- Materials allowed:
  - The exam page (includes a reference sheet)
  - Your own reference sheet (1 page front and back, written or typed)
  - A writing implement
- Not allowed:
  - Anything electronic (laptop, phone, tablet, earbuds, etc.)

List Data Structures

- Goal:
    - Store a sequence of things
        - Sequences have order (indexing, next)
        - Sequences can have repeats
- Operations:
    - Add
        - To beginning
        - To end
        - At an index
    - Remove
    - Get
        - At an index

Linked Lists vs Array Lists

Operation	ArrayList	LinkedList
`add(index, value)`	For each item at or after `index`, shift it to the right by one. Put `value` at `index` Time:	Create a new node whose data field is `value` If `index==0`, `newNode.next=front`, `front=newNode` Otherwise follow `.next` `index-1` times, `newNode.next=curr.next`, `curr.next=newNode` Time:
`remove(index)`	For each item at or after `index`, shift it to the left by one Time:	If `index==0`, `front=front.next` Otherwise follow `.next` `index-1` times, `curr.next = curr.next.next` Time:
`remove(value)`	For each index, check if the item matches `value`. If so, shift everything after it to the left. Time:	Follow `.next` until `curr.next.data` matches `value`. `curr.next = curr.next.next` Time:
`get(index)`	Return the thing at `index` of the array Time:	Follow `.next` `index` times, return `curr.data` Time:

Set Structures

- Goal:
  - Store a Collection with no order, no duplicates
- Operations:
  - Add
  - Remove
  - Contains
- Ideas:

ReallyBigArray

- Have a really big array of booleans
  - Every possible int gets its own index
  - Length is `Integer.MAX_VALUE`
  - If `bigArray[x]` is `true`, then `x` is in the set
- What’s wrong with this?

Better Ideas

- Use Binary Search Trees!
  - When calling add, remove, contains we only need to go left or right at each level
    - Each level you cut the number of items in half! (ideally…)
- Use HashSets!
  - Use a small array to store items
  - Use a hash function to select an index in that small array
    - Selected index should be hard to predict so that the small array behaves similarly to the big array
  - If two different items select the same index, deal with it…