1 CS3134 #9

9/30/03

Janak J Parekh

2 Administrivia

- None, for a change
- Questions?

3 Agenda

- Circular queues
- Priority queues
- Linked lists

4 Queues: Review

- FIFO, instead of LIFO
- Insert, Remove, Peek
- Book's convention: front is at bottom, near beginning of array – doesn't matter as long as you're consistent
- Problem: how to represent in array?
  - We can't stick it at one end or the other, unless we slide all the elements around
  - There's a better approach

5 Circular queue

- Don't move elements around, keep front and back pointers
- Yes, back/front can wrap around: "broken sequence"
- Keep track of number of elements – i.e., full/empty
- Convention: initialize rear to -1, front to 0

6 Circular queue operations

- Be very careful of keeping pointers consistent
  - Pointers should not "cross" unless empty
- Insert
  - If rear at last element (length-1), reset to -1
  - Increment rear, and then place the object in the new rear
  - Increment # of items
- Remove
  - Grab element at front, and then increment it
  - If front is off the end (== length), reset to 0
  - Decrement # of items
- Why -1?
  - Convention so that rear actually points to the newest-added element
  - You can program with 0 if you're careful
- Efficiency of operations?

7 Circular queue: miscellany

- Having to keep count is a little extra work
- Book has sample code to deal with "no-count" implementation, but more complex
  - Basic problem: how to tell queue empty vs. full
  - Trick: if full, leave an empty space (i.e., make array one cell larger than maximum # of items), and check for the empty space
    - One apart => empty; two apart => full

    - Two cases for each:
        - If front is “ahead” of rear
        - If front is “behind” rear

8 Other queues

- Deque: “double-ended” queue – essentially a stack and queue combined: insert/remove left/right
- Priority queue
    - The idea is that the object of “highest priority” will be next to be dequeued
    - Typically, process array during insert such that front is pointing to highest-priority element
    - Book’s implementation does insertion sort: starts at end, and moves elements up until it’s in the right position
    - No benefit to using circular constructs, so very similar to naïve queue approach
    - Complexity? (Heaps are better, but later)

9 Linked lists

- Arrays are rather limited, cumbersome data structures – cells are “fixed” together, limited length
- What if we could break apart the cells?
- We *can*!
- In fact, linked list-style structures are used more frequently unless you need very fast random index-based access
- Trees, graphs, etc. are generalizations of linked lists

10 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

11 Next time…

- Finish linked lists