Queues

First, some comments

- Pair Programming issues
  - It's not pair programming if you don't meet with your partner
  - It's not fair to either of you if only one of you does the work
  - I'm thinking of a way to adjust scores on the programming assignment based on each partner's participation.
- Friday lab
  - Tried to get one, but...
    - Crown available 4-6, but neither TA can do it then
    - BE 105 (PC lab) available Thursday 4-6, conflicts with class
- Programming Assignment 2 due date?

First, some comments

- This week
  - Finish stacks
  - Queues
  - Maybe start recursion
  - Midterm review – remember, midterm next Tuesday
- Reading
  - Chapter 6: Section "Application: Algebraic Expressions"
  - Chapter 7: All of it
What is a queue?
- Stacks reverse the order of items added to them
  - Last In-First Out
- What if we want to preserve the order in which items are added?
- Solution: queue
  - First In-First Out: items removed in the same order they’re added
  - Similar to line (queue) at bank, supermarket, etc.
- Uses in computer science include
  - Buffering (keyboard, network, etc.)
  - Printer queue, job queue
  - Simulations
  - Lots of other stuff

Operations on queues
- Same kinds of operations as stacks, but slightly different results
- Create
- Enqueue: add to the tail of the queue
- Dequeue: remove from the head of the queue
- Peek: look at the element at the head of the queue
- IsEmpty: tell whether the queue is empty
- DequeueAll: clear all elements from the queue

Queue using circular linked list
- Last element in list points back to the first one
- Enqueue (adding an element)
  - newNode.next = lastNode.next
  - lastNode.next = newNode
- Dequeue (removing an element)
  - firstNode = lastNode.next
  - lastNode.next = firstNode.next
  - return firstNode
- Peek (look at first element)
  - firstNode = lastNode.next

A B C D
head

A B C D
lastNode
Details on queues with linked lists

- Some methods can throw QueueException (like StackException)
  - Dequeue() on an empty queue
  - Peek() on an empty queue
  - Enqueue() on an empty queue is a bit different
    - lastNode = newNode
    - newNode.next = newNode
  - Dequeue() on a queue with exactly one element is different
    - firstNode = lastNode
    - lastNode = null
  - DequeueAll() can be done by lastNode = null

Queues with arrays?

- As with stacks, queues can be implemented with arrays
- Naïve implementation
  - Insert at top of array
  - Remove from bottom of array (element 0) and shift array contents down one place
  - Problem: this can be slow for large arrays!
- Better implementation: circular array
  - Keep track of start and end of queue
  - Queue “wraps around” the end of the array
  - Use modular arithmetic for array indexes
  - Space-efficient and fast

Circular arrays for queues

- Enqueue
  - back = (back+1) % MAX_QUEUE
  - queueArray[back] = newItem
  - count++;
- Dequeue
  - item = queueArray[front]
  - front = (front+1) % MAX_QUEUE
  - count--;
- Wraps around when front or back reaches max_queue
Details on queues with arrays

- Some methods can throw QueueException, as with linked list queues
  - Dequeue() on an empty queue
  - Peek() on an empty queue
- Queue is empty when count==0
  - There can be two situations where front==back
    - If count==max_queue, queue is full
    - If count==0, queue is empty
- More efficient implementation leaves an empty array element when Queue is full.

- Array-based queue can fill up!
  - Enqueue() can throw a QueueException if count==max_queue
  - Make the queue array large enough to avoid this
  - DequeueAll() can be done by setting front=0, back=MAX_QUEUE – 1, count=0
  - Same code as used to initialize an array-based queue.

Implementing queues (and stacks)

- Three choices for implementing queue ADT
  - List ADT
    - Array (circular)
      - List ADT is simpler: less code to write
    - Array
      - Fixed maximum size
      - Low overhead (no link references)
    - Linked list
      - Grows to any size
      - Requires more space for a given number of elements
- In languages other than Java, allocating and deleting elements is an issue
  - This favors arrays, which don’t need to allocate and delete very frequently (array methods don’t call new)

Queue application: simulations

- Computers often used to simulate behavior
  - Customers at a bank
  - Requests serviced by a roomful of Web servers
  - Traffic on roadways
- All of these simulations consist of events
  - An event occurs at a given time, determined by the model used in the simulation
  - Events could include
    - Car X enters Highway 1 at Morrissey
    - Car X switches lanes at mile marker X
    - Car X leaves freeway at 41st Avenue
  - Simulation must keep track of thousands of events
  - Events ordered by the time they occur
  - Must process events in time order
- Use a queue!
Sample simulation: supermarket

- N checkout lines
  - Each is FIFO
  - Each line serves the shopper at the front
  - Time to service is determined by simulation
- Shopper may choose a line
  - Simulation decides how rapidly shoppers arrive
  - Simulation decides which line a shopper picks
  - "Express" line?
  - Test different strategies
- Questions to answer:
  - How many lines should there be?
  - How should a shopper pick the best line?

Simulating a supermarket

- Each line is ordered by time
  - Customer at front of line is next to finish (in that line)
  - Amount of time to finish determined by simulation
- Simulation picks next to finish from front of all queues
  - Advances "time" to t
  - Dequeues the customer who finishes at time t
- This repeats as long as simulation runs
- More advanced simulations may have more complex queueing
  - Time spent in each aisle
  - Time spent looking for items
  - Even more detail...