CS 171: Introduction to Computer Science II

Queues

Li Xiong
Announcements/Reminders

• Hw1 mean: 91, median: 99
• Hw2 due Monday, February 18
Roadmap

• Stacks
• Queues
• Deque
• Bag
Stack Review

• **Stack**: a useful abstract data type
• Key operations: **push, pop**
• Property: Last in first out (LIFO)
• **Implementations**: using (resizing) arrays and generics
• Applications: reversing, matching, expression evaluation, N-queens/backtracking
Stacks and queues

Fundamental data types.

- Value: collection of objects.
- Operations: insert, remove, iterate, test if empty.
- Intent is clear when we insert.
- Which item do we remove?

**Stack.** Examine the item most recently added. LIFO = "last in first out"

**Queue.** Examine the item least recently added. FIFO = "first in first out"
Queues

- The word *Queue* is British for *Line*.
  - The first person that enters the queue gets served
Queues

• Key operations
  – enqueue (insert)
  – dequeue (remove)
• Property: First-in-first-out (FIFO)
• First element referred to as front (or head)
• Last element referred to as rear (or tail)
Queue Applications

• Queues are very useful in a computer
  – Printer queue
  – Keyboard buffer
  – Network buffer
  – ...

public class QueueOfStrings

QueueOfStrings()  
create an empty queue

void enqueue(String s)  
insert a new item onto queue

String dequeue()  
remove and return the item least recently added

boolean isEmpty()  
is the queue empty?

int size()  
number of items on the queue
Queue: Implementation

• Use an array similar to stack
• Where to enqueue? where to dequeue?
• How many elements are in the queue?
• What happens if head is at the end of the array but there is still space at the front of the array?
• When is the queue empty, full?
Array implementation of a queue.

- Use array `q[]` to store items in queue.
- `enqueue()`: add new item at `q[tail]`.
- `dequeue()`: remove item from `q[head]`.
- Update `head` and `tail` modulo the `capacity`.
- Add resizing array.

<table>
<thead>
<tr>
<th>q[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>the</td>
</tr>
<tr>
<td>best</td>
</tr>
<tr>
<td>of</td>
</tr>
<tr>
<td>times</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>null</td>
</tr>
<tr>
<td>null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>head</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`capacity = 10`
Queue: Implementations

• Generic Queue using resizing array
  ResizingArrayQueue.java
• Generic queue using linked list (later)
  Queue.java
Queue applications: Josephus problem

N people agree to the following strategy to reduce the population. They arrange themselves in a circle (at positions numbered from 0 to N-1) and proceed around the circle, eliminating every Mth person until only one person is left.

Print out the order in which people are eliminated.
public class Josephus {
    public static void main(String[] args) {
        int M = Integer.parseInt(args[0]);
        int N = Integer.parseInt(args[1]);

        // initialize the queue
        Queue<Integer> q = new Queue<Integer>();
        for (int i = 0; i < N; i++)
            q.enqueue(i);

        // eliminating every Mth element
        while (!q.isEmpty()) {
            for (int i = 0; i < M-1; i++)
                q.enqueue(q.dequeue());
            StdOut.print(q.dequeue() + " ");
        }
        StdOut.println();
    }
}
Bonus Question

Suppose an intermixed sequence of (queue) *enqueue* and *dequeue* operations are performed. The enqueue operations put the integers 0 through 9 in order onto the queue; the dequeue operations print out the return value. Which of the following sequence(s) could not occur?

(a) 0 1 2 3 4 5 6 7 8 9  
(b) 4 6 8 7 5 3 2 9 0 1  
(c) 2 5 6 7 4 8 9 3 1 0  
(d) 4 3 2 1 0 5 6 7 8 9

Suppose that an intermixed sequence of (stack) *push* and *pop* operations are performed. The pushes push the integers 0 through 9 in order; the pops print out the return value. Which of the following sequence(s) could not occur?

(e) 1 2 3 4 5 6 9 8 7 0  
(f) 0 4 6 5 3 8 1 7 2 9  
(g) 1 4 7 9 8 6 5 3 0 2  
(h) 2 1 4 3 6 5 8 7 9 0
Bonus Question

Suppose that an intermixed sequence of (stack) push and pop operations are performed. The pushes push the integers 0 through 9 in order; the pops print out the return value. Which of the following sequence(s) could not occur?

(e) 1 2 3 4 5 6 9 8 7 0
(f) 0 4 6 5 3 8 1 7 2 9
(g) 1 4 7 9 8 6 5 3 0 2
(h) 2 1 4 3 6 5 8 7 9 0
Roadmap

- Stacks
- Queues
- Deque
- Bag
Deque

• **Double-ended** queue

• Can insert and delete items at either end

• Can be a Stack or a Queue!
  – addFirst, addLast, removeFirst, removeLast

• **Stack**: if only **addLast** and **removeLast**

• **Queue**: if only **addLast** and **removeFirst**
Bag

- A *bag* is a collection where removing items is not supported.
- To support insert and iteration through the collected items.
- **Stats.java** is a bag client that reads a sequence of real numbers from standard input and prints out their mean and standard deviation.
Bag

```java
public class Bag<Item> implements Iterable<Item>
```

- Bag()
- void add(Item item)
- boolean isEmpty()
- int size()

create an empty bag
add an item
is the bag empty?
number of items in the bag

FIFO queue

```java
public class Queue<Item> implements Iterable<Item>
```

- Queue()
- void enqueue(Item item)
- Item dequeue()
- boolean isEmpty()
- int size()

create an empty queue
add an item
remove the least recently added item
is the queue empty?
number of items in the queue

Pushdown (LIFO) stack

```java
public class Stack<Item> implements Iterable<Item>
```

- Stack()
- void push(Item item)
- Item pop()
- boolean isEmpty()
- int size()

create an empty stack
add an item
remove the most recently added item
is the stack empty?
number of items in the stack
Priority Queue

• An abstract data type like a regular queue, but each element is associated with a priority value, and the element with highest priority will be removed first
  – Queue: first in first out
  – Stack: last in first out
  – Priority queue: highest priority first out

• Priority queue applications
  – Task scheduling
  – Search and optimization
Implementing Priority Queue

• Assumes an element with a smaller value has higher priority.

• Implementation using array

• Insert
  – Inserts an element to the correct position, similar to insertion into an ordered array

• Remove
  – Deletes the front element
Using Priority Queues

- Example:

```java
PriorityQueue s = new PriorityQueue(10);
s.insert(25);
s.insert(35);
System.out.println(s.remove());
s.insert(45);
s.insert(15);
System.out.println(s.remove());
System.out.println(s.remove());
```
Roadmap

• Abstract data types
  – Stacks
  – Queues

• Linked list

• Implementing stacks and queues using linked list