CS 310: Stacks/Queues by Arrays/Links

Chris Kauffman

Week 3-2

Effective Procrastination

- Adam Grant: Can Slowing Down Help You Be More Creative?
  - Start something early (Milestone Deadline)
  - Then take a break
  - Then finish strong (Final Deadline)
- Tim Urban: What Happens In The Brain Of An Extreme Procrastinator?

Early

Later....

Logistics

At Home

- Weiss Ch 15 on ArrayLists
- Weiss Ch 16 Stacks/Queues
- Weiss Ch 17 Linked Lists (next time)
- HW 1 Milestone: Due Sat 6/17
- HW 1 Final: Due 6/24
- Questions on HW 1?

Goals Today

- Finish up ArrayList
- Implementation of Stacks and Queues

HW 1 Worst to Best

- Noncompiling code
- Code that compiles
- Code that compiles and passes most/all tests
- All of the above PLUS is clean and understandable
- All of the above PLUS code clearly meets complexity bounds, perhaps justified in comments

Last Time: `ArrayList` complexities

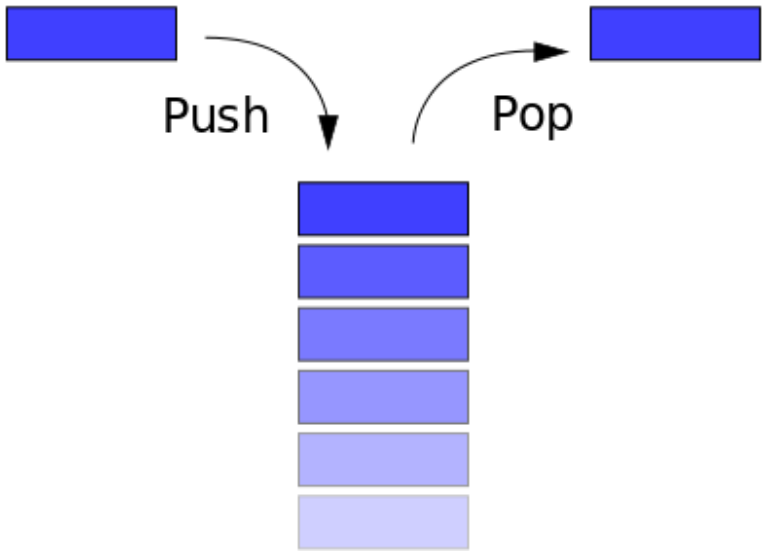
- `ArrayList` of size N
- Total space complexity: $O(N)$

Operation	Method	Worst Time	Average Time	Worst Space	Average Space
Size()	`al.size()`	O(1)	O(1)	O(1)	O(1)
Get(i)	`al.get(i)`	O(1)	O(1)	O(1)	O(1)
Set(i,x)	`al.set(i,x)`	O(1)	O(1)	O(1)	O(1)
Add(x)	`al.add(x)`	O(N)	O(1)	O(N)	O(1)
Insert(i,x)	`al.add(i,x)`	O(N)	O(N)	O(N)	O(1)
Remove(i)	`al.remove(i)`	O(N)	O(N)	O(1)	O(1)

Stacks

Simple structure, supports few operations:

- `T s.getTop()`: return whatever is on top
- `s.push(T x)`: put `x` on top
- `void s.pop()`: remove whatever is on top
- `boolean s.isEmpty()`: `true` when nothing is in it, `false` o/w



Stacks are a LIFO:
Last In First Out

Questions

- Examples of stacks?
- How would you implement a stack using arrays?

Exercise: Array Based Implementation

Just use `ArrayList` to make an `AStack`

```
class AStack<T>{
  private ArrayList<T> stuff;
  public AStack();                  // Constructor
  public void push(T x);            // Like add(x)
  public void pop();                // Like remove(size()-1)
  public T getTop();                // Like get(size()-1)
  public boolean isEmpty();         // Like size()==0
}
```

See: `weiss/nonstandard/ArrayStack.java`

Work It

- Stacks: more or less functionality than `ArrayList`?
- Worst and Amortized Complexity of stack operations?
- Can we do better?

Nodes

To get worst-case $O(1)$ `push`, need to change the underlying representation of the stack implementation.

- Simplest unit to support linked data structure
- `ListNode` in text
- *Cons box* in Lisp
- Tracks a piece of `data` and the `next` node in a sequence
- String them together by setting `next`

`Node` Class

```
class Node<T>{
  public T data;
  public Node<T> next;

  public Node(T d,
              Node<T> n ){
    this.data = d;
    this.next = n;
  }
}
```

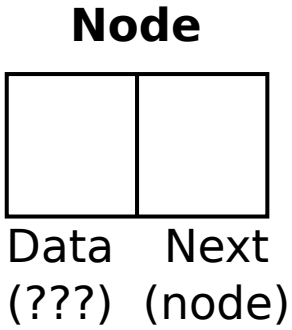
Linked Nodes

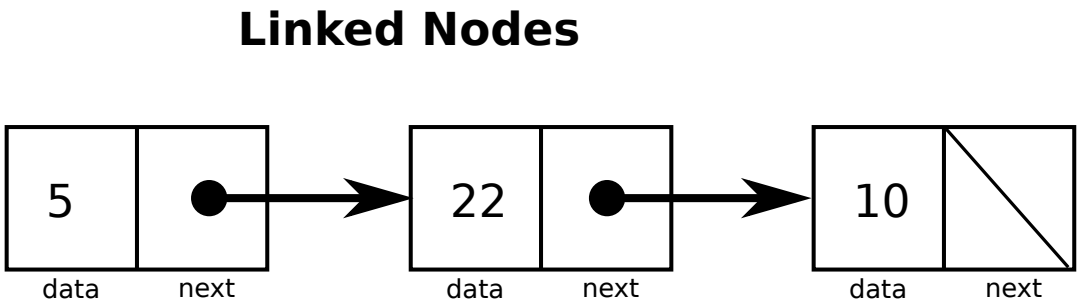
Can string Nodes together by manipulating the next field

```
class Node<T>{
  public T data;
  public Node<T> next;

  public Node(T d, Node<T> n ){
    this.data = d;
    this.next = n;
  }
}
```

```
Node<Integer> n3 =
  new Node<Integer>(10,null);
Node<Integer> n2 =
  new Node<Integer>(22, n3);
Node<Integer> n1 =
  new Node<Integer>( 5, n2);
Node<Integer> head = n1;
```





Implement a Stack with linked Nodes

```
class LinkedStack<T>{
  Node<T> topNode;                // Pointer to top node
  public LinkedStack();           // Constructor
  public void push(T x);          // Push an element
  public void pop();              // Pop an element
  public T getTop();              // Return top element
  public boolean isEmpty();       // True only when empty
}
```

Assume

Node<T> class is available

```
Node<T> n = new Node<T>(data,null);
T stuff = n.data;
n.next = anotherNode;
```

Consider

- Which end of the stack needs to be tracked?
- How would code change if size were needed?

Implementations of Stacks

Weiss Textbook Source

- `package weiss.nonstandard.*`
  - `Stack` interface, `ArrayStack` and `ListStack` implementations
  - `ListNode` and `LinkedList` classes
- `package weiss.util.*`
  - Reimplements `java.util.*` collections
  - `Stack.java` is based on arrays
- Included in today's code pack

Java

- `Deque` interface - slight generalization of stack/queue
- `ArrayDeque` implements with arrays
- `LinkedList` implements with linked nodes

Stack Implementation with Nodes

Generic

Stacks can hold any type of thing

```
class Stack<T>{...}
class UseStack{
  main(){
    Stack<String> strs =
      new Stack<String>();
    strs.push("Hi");
    String s = strs.getTop();
    strs.pop();

    Stack<Integer> ints =
      new Stack<Integer>();
    ints.push(1);
    int one = ints.getTop();
    ints.pop();
  }
}
```

Inside Classes

`Node` class *could* live inside `LinkedStack`

```
public class LinkedStack<T>{
  Node top;
  public void push(T t){...}
  public void pop(){...}
  public T top(){...}

  static class Node<X>{
    X data; Node<X> next;
    public Node(X data, Node<X> next){
      this.data = data;
      this.next = next;
    }
  }
}
```

Note: Contiguous vs. Non-contiguous memory

```
s = new Stack();
s.push(4);
s.push(10);
s.push(5);
s.pop();
s.push(11);
```

Array-based Stack

s	2048	1024
	...	
size	3	2048
data	4048	2052
	...	
length	5	2044
[0]	4	4048
[1]	10	4052
[2]	11	4056
[3]	0	4060
[4]	0	4064

Node-based Stack

s	2048	1024
	...	
topNode	4048	2048
	...	
data	11	4048
next	5050	4052
	...	
data	4	4600
next	null	4604
	...	
data	10	5050
next	4600	5054

Get in Line

Queues are pervasive in computing and life

- Examples?
- Semantics?

Source: kittylittered

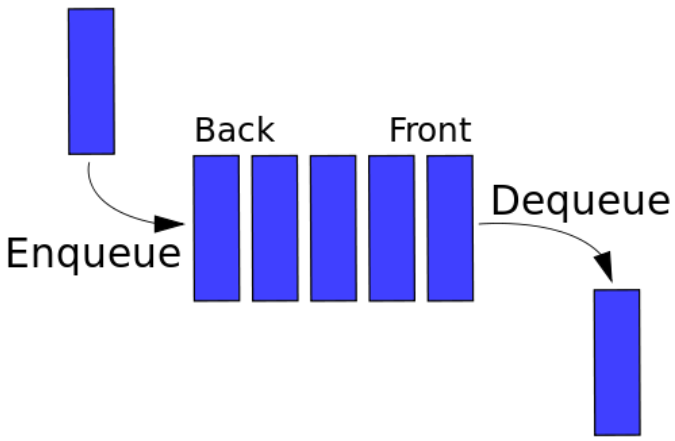
Queues

Support 4 operations

- `enqueue(x)`: `x` enters at the back
- `dequeue()`: front leaves
- `getFront()`: return who's in front
- `isEmpty()`: `true` when nothing is in it, `false` o/w

Goal:

- Worst case $O(1)$ for all ops
- $O(n)$ space



Source: Wikipedia

LinkedQueue: Ideas

- Draw pictures showing data changes for the following code
- Draw the Nodes and connections at each step
- Decide what parts of the queue need to be tracked with fields

```
LinkedQueue<String> bsg = new LinkedQueue<String>();
bsg.enqueue("Adama");         // Add to back
bsg.enqueue("Tye");
bsg.enqueue("Starbuck");
bsg.dequeue();                // Remove from front
String col = bsg.getFront(); // Who's in front
bsg.dequeue();
bsg.enqueue("Apollo");
bsg.enqueue("Baltar");
bsg.dequeue();
bsg.enqueue("Number 6");
bsg.dequeue();
```

Queue Picture Demo

- In `weiss/nonstandard/ListQueue.java`
- Also in code pack from last week
- Uses `ListNode.java`, more verbose `Node`
- JGrasp can draw these reasonably well

Exercise: Create a LinkedQueue with Nodes

```
class LinkedQueue<T>{
  Node<T> front, back;
  public LinkedQueue();
  public void enqueue(T x);      // x enters a back
  public void dequeue()=;        // front leaves
  public T getFront();           // return who's in front
  public boolean isEmpty();      // true when empty
}
```

Consider

- Worst case $O(1)$ for all ops
- How to remove the front?
- How to add to the back?

ArrayQueue: Ideas

Use an array / `ArrayList` to implement a queue.

- Easy... or is it? Beware of memory use: O(N)
- Draw pictures, figure out fields

```
ArrayQueue<String> bsg = new ArrayQueue<String>();
bsg.enqueue("Adama");        // Add to back
bsg.enqueue("Tye");
bsg.enqueue("Starbuck");
bsg.dequeue();               // Remove from front
String col = bsg.getFront(); // Who's in front
bsg.dequeue();
bsg.enqueue("Apollo");
bsg.enqueue("Baltar");
bsg.dequeue();
bsg.enqueue("Number 6");
bsg.dequeue();
String doc = bsg.getFront(); // Who's in front
bsg.enqueue("Boomer");
bsg.enqueue("Helo");
```

ArrayQueue: Code

Use an array / `ArrayList` to implement a queue.

- Easy... or is it?

Define

- `ArrayQueue` data members
- `enqueue(x)`: `x` enters at the back
- `dequeue()`: front leaves
- `getFront()`: return who's in front
- `isEmpty()`: `true` when nothing is in it, `false` o/w

Goals

- Amortized $O(1)$ for all ops
- $O(N)$ space
- Worst-case $O(N)$ `enqueue` is fine
- Most `enqueue()` ops should be $O(1)$
- Control memory use
- Hint: track index of front and rear, *wrap* arrays around

ArrayQueue Demo

- In `weiss/nonstandard/ArrayQueue.java`
- Slightly modified demo version in today's code pack
- Uses plain java arrays, not `ArrayList`
- Array doubling in size done manually

Note interesting functions in Weiss's version

```
private int increment(int x){...}
private void doubleQueue() { ... }
```

jGrasp Drawing

Drawing isn't uber smart

- May have to manually turn on display of some fields
  - `back` in `ListQueue`
  - Wrench Button -> Fields Display
- Doesn't get nested arrays or `ArrayLists`