Announcements

- Next Monday: review for midterm
- Wednesday Feb 21 – Midterm
  - Open book
  - Will have a review session before hand

- Reading:
  - Chapter 3.7, 4.1, 4.2
Outline

- Queues
  - Code
- Trees
  - Code
- Sorts
Running eclipse

I’m installing it out of the class account

~cs3134/public/eclipse

Should work if they give me a little more space today
So far

- **List DS**
  - Group of items
  - Insert, delete, find
  - Trade offs on implementations

- **Stack DS**
  - LIFO
  - Organization of items in specific ways
  - Postfix math
  - Recursion programming
Queue DS

- List with specific property:
  - FIFO

- Enqueue
  - Enter item in the queue

- Dequeue
  - Remove item from queue

- Example:
  - Shared resource, like printer, submit documents to print
Simple

- Array
  - use a simple array
  - Lets talk about how it would work

- What are the runtimes ??
Circular List

- An alternative is to use an array but with two extra indices showing where the head/tail are located

- What will be the run time?
- How to determine size?

- Limitations?
Priority Queue

- Another type of DS in which not exactly FIFO

- This queue also keeps track of how important something is...higher importance should go first

- Will talk about this after we finish with trees, but from high level how long would things take now?
Reverse

- We spoke about reversing the stack DS

- Can you think of a fast way to reverse a queue??
Examples

- Phone service
  - Call up for service
  - Wait on queue
  - Hear some type of music
  - More music
  - More
  - Either hang up in disgust
  - Or get operator...
Simulations

- Many different branches of science use computers besides jokeoftheday.com or paperweight 😊

- Useful programs to run calculations
  - Usually super expensive so everyone leases times on these machines
  - Guess what DS is involved?
Elevator simulation

- Assuming everyone has used elevators

- Anyone know what software are run on elevators??

- What are the basic tasks that the software would aim to achieve??
Switch

- ok, let's leave basic data structures
- I’m hoping you feel comfortable with the code
- and start on more complicated ones....
List limitations

- So there are some clear limitations with lists in general

- How to improve things?
Improvement one

- Lets not line up things

- Then...

- Lets do some work earlier to make things faster
Definition

-[Tree]

- a collection of nodes consisting of a root node and zero or more non-empty subtrees each of whose roots are connected by a directed edge from the primary root
Definition

- Parent node
  - node at the beginning of a directed edge

- Child node
  - node at the end of a directed edge

- Internal node
  - non root with children
- basic tree
  - just root with no children

- leaf
  - tree with no children
Definition

- A Path from node\(i\) to node\(k\) consists of a sequence of nodes \(n_i, n_{i+1}, \cdots, n_k\) such that \(n_i\) is the parent of \(n_{i+1}\) and \(i \leq k\)
Question

- So what can trees represent??
- family relationships
- file systems
- organizations
- game strategy
- dictionary
If the tree has up to $M$ number of children per node then it's called a $M$-ary tree

- For example: coding dictionary, an ideas on how to do it

Binary Tree

- either single node or node with up to 2 children which are binary trees
- left, right

Because trees are recursively defined, the algorithms to manipulate them are usually coded recursively
public class BintreeNode <anytype> {
    anytype item;
    BintreeNode<anytype> left;
    BintreeNode<anytype> right;
}

public class BinaryTree <sometype> {
    BintreeNode<sometype> root;
}
definition

- Binary Search Tree
  - tree in which every element to the left is less than current node, and every element to the right is greater than current node
- lets do some code
- can you fill this in ??

```java
class BST <ANYTHING>{
}
```
Methods to do

- constructors
- insert
- find
- delete
- print
Empty tree

```java
BinaryTree<sometype>() { 
    root = null;
}
```
Question

- So what does an empty tree look like??
isEmpty

public boolean isEmpty(){
    return root == null;
}