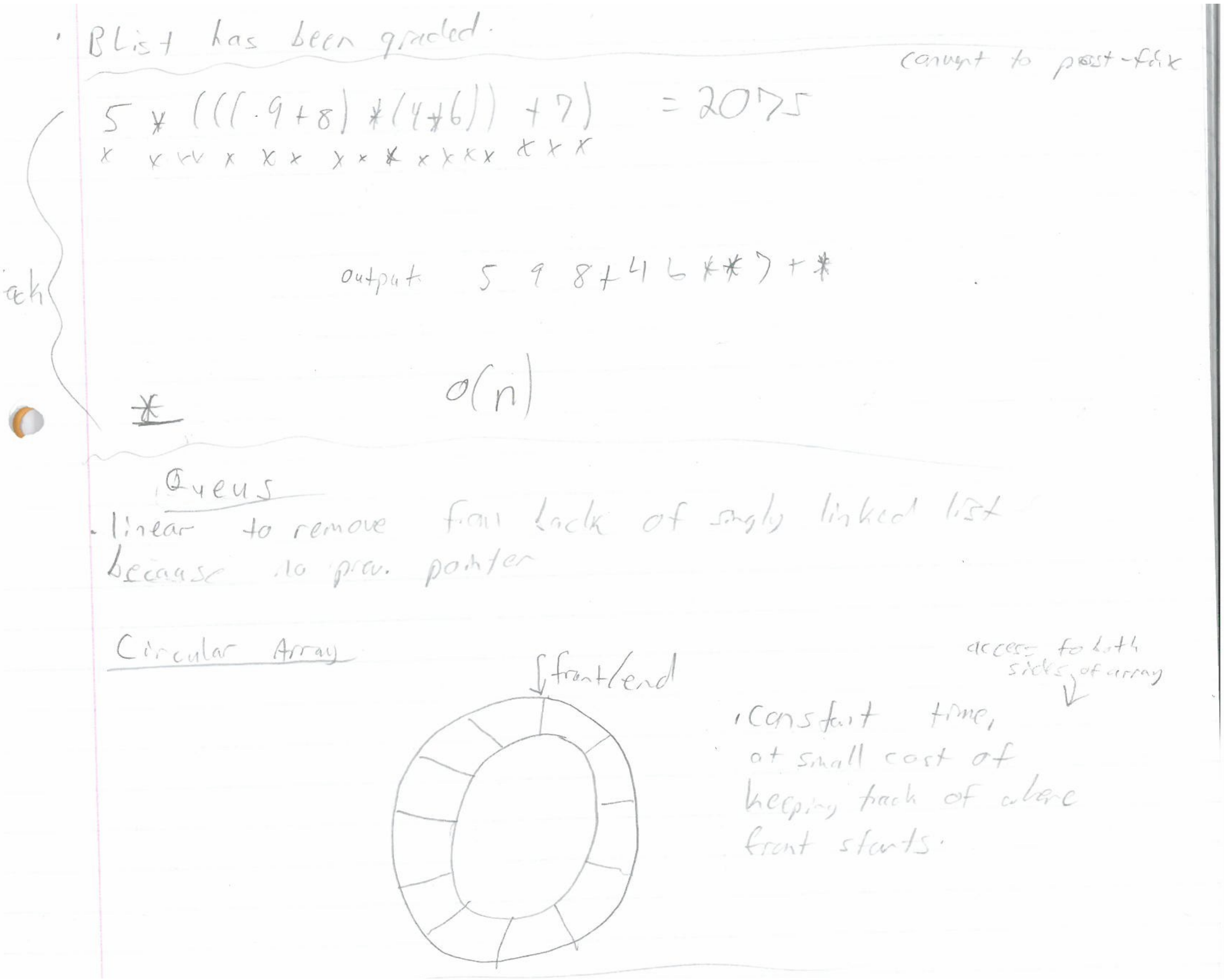
CS280 – Trees, Stacks, Queus
Febuary 9, 2016

http://azrael.digipen.edu/~mmead/www/Courses/CS280/Trees.html

Recursion Assignment

- How do we use the callback function?
  - It's like a regular function, just call it with all the arguments like you normally would
- Blist assignment has been graded



· Blist has been graded.

convert to post-fix

5 * (((9+8) * (4*6)) + 7) = 2075

ack

output 5 9 8 + 4 6 * * 7 + *

O(n)

*

Queus

· linear to remove from back of singly linked list because no prev. pointer

Circular Array

front/end

access to both sides of array

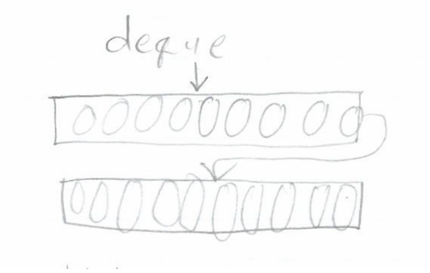
· Constant time, at small cost of keeping track of where front starts.

-Using a stack to convert a math expression to post-fix notation.

-Circular array

Ques

- Linear to remove from back of singly linked list because there is no previous pointer
- Tail moves when we add, head moves when we remove



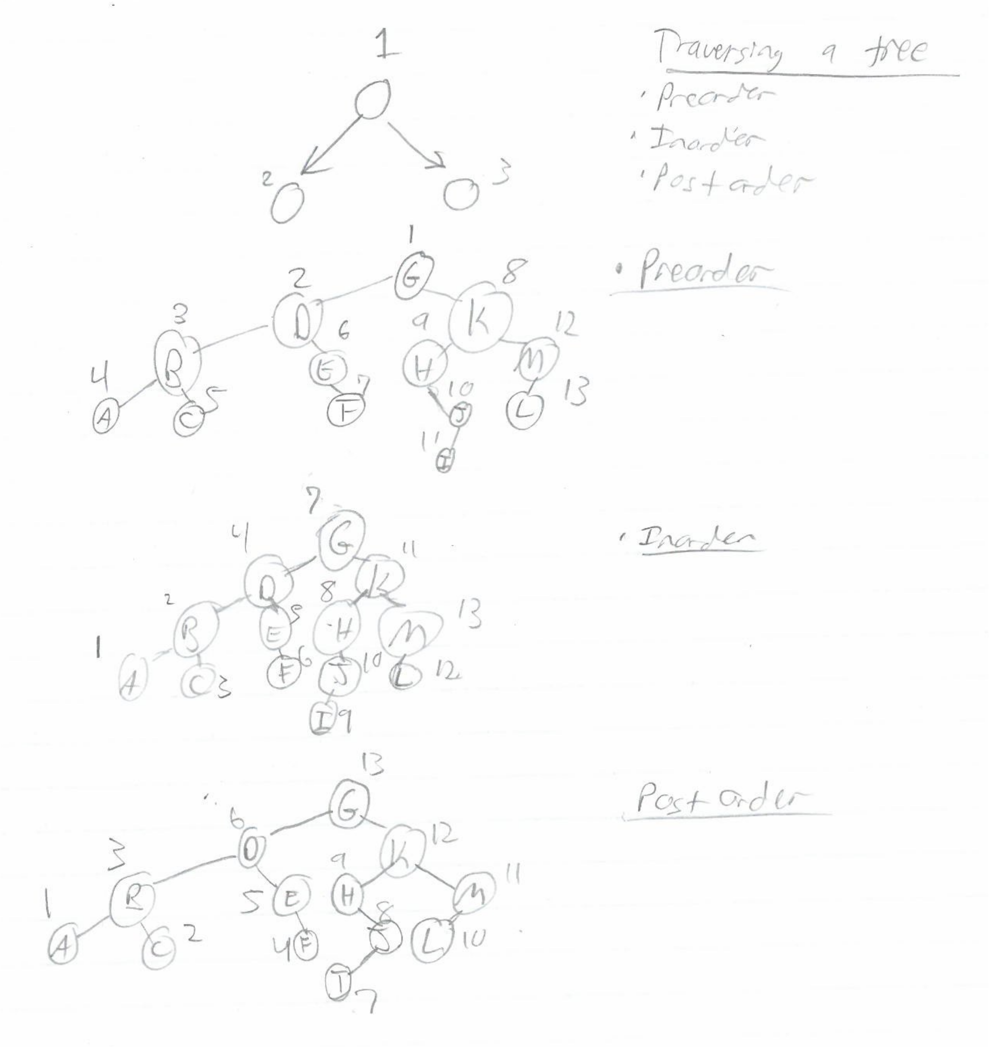
Double ended que

- You wouldn't sort a que or deque
- If you keep things sorted, it is linear to add, constant to access/remove
  - By contrast, if you don't keep it sorted, it is constant to add, linear to access / remove
- Four typical options with data structures
  - Array or linked list
  - Sorted or unsorted
- In a que/deque, when removing, why shift any elements?
  - Just take element from back and fill in the hole left by removing

- Do most modern data structures come down to either arrays or linked lists?
    - Yes, almost all data structures are either 'array-based' or 'node-based'

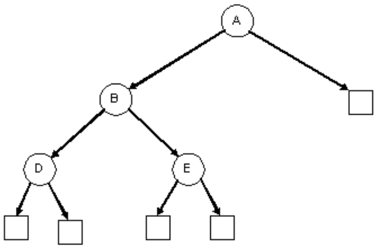
<u>Trees</u>

- Balance tree
    - Lets you binary search a linked list
- Recursion is generally easier when working with trees than using iteration
- Trees are node-based
- Keeping trees 'balanced' is like keeping a list/array sorted
    - More effort on insert, faster on access/modification
- Root of the tree ~= head of linked list
    - Root has no parent
- There is exactly one path from the root to any child
- No shortcut to find the height of a tree
- Degenerate tree
    - Basically just a linked list



Traversing a tree

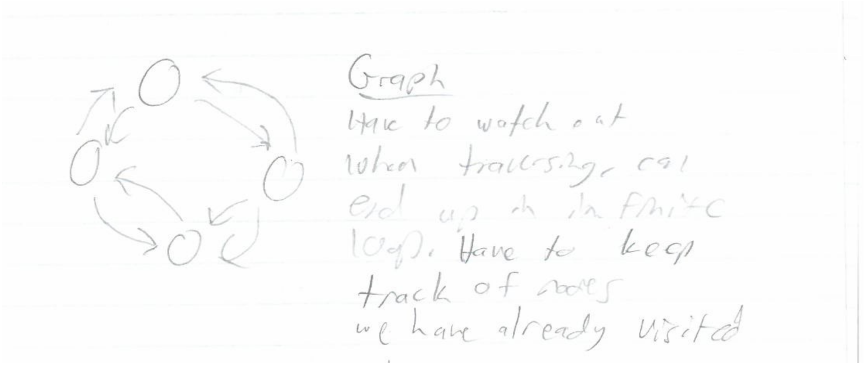
- When answering if a tree is balanced, we have to say which is the “offending node”



Here, 'A' is the “offending node”

- TreeNode struct is the same as double linked list
    - Difference is how we link them up

Watch out for situations like this when we start talking about graphs. Easy to get yourself into an infinite loop when trying to traverse



Graph

Have to watch out when traversing, can end up in infinite loop. Have to keep track of nodes we have already visited