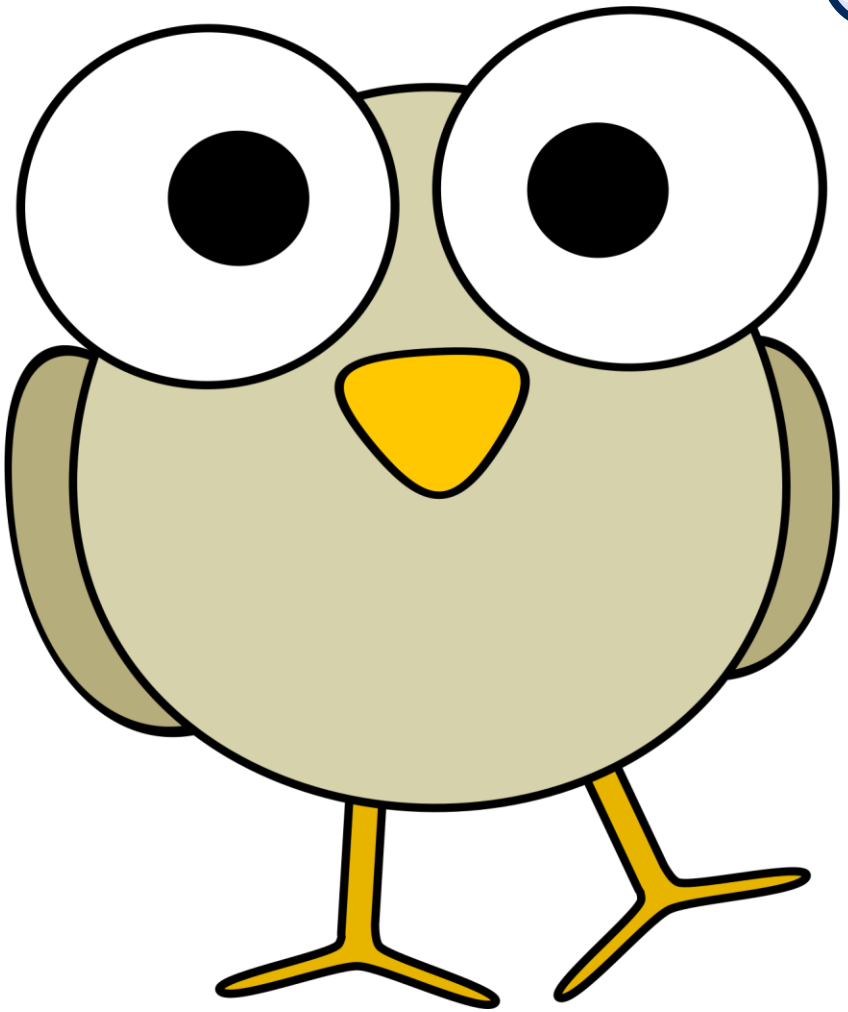
CSCI 200: Foundational Programming Concepts & Design Lecture 41

Trees & Graphs

Previously in CSCI 200

- BFS and DFS -- $O(n^2)$
  - Same pseudocode except for Queue or Stack implementation
    - Explore neighbors recursively via iterative implementation
  - Reach same conclusion, potentially at varying speeds
  - "Graph" algorithms to discover paths between connected nodes

Questions?



On Tap For Today

- Trees
- Graphs
- Practice

On Tap For Today

- Trees
- Graphs
- Practice

Data Structure Operations

Operation		Array	Doubly-Linked List
Element Access		$O(1)$	$O(n)$
Traversal	Forwards	$O(n)$	$O(n)$
	Backwards		
Add	Front	$O(n)$	$O(1)$
	Middle		$O(n)$
	Back		$O(1)$
Delete	Front	$O(n)$	$O(1)$
	Middle		$O(n)$
	Back		$O(1)$
Search		$O(n)$	$O(n)$
Min / Max		$O(n)$	$O(n)$
Memory		n*sizeof(T) contiguous	n*(sizeof(T)+16) fragmented

Algorithm Complexities

Algorithm	Worst Case	Best Case	Average Case
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Bubble Sort	$O(n^2)$	$O(n)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Algorithm	**Worst Case**	**Best Case**	**Average Case**
Linear Search	$O(n)$	$O(1)$	$O(n)$
Binary Search	$O(\log n)$	$O(1)$	$O(\log n)$

Data Structure Operations

Operation		Array	Doubly-Linked List
Element Access		$O(1)$	$O(n)$
Traversal	Forwards	$O(n)$	$O(n)$
	Backwards		
Add	Front	$O(n)$	$O(1)$
	Middle		$O(n)$
	Back		$O(1)$
Delete	Front	$O(n)$	$O(1)$
	Middle		$O(n)$
	Back		$O(1)$
Sort		$O(n \log n)$	$O(n \log n)$
Search	Linear		
	Binary		
Min / Max	Unsorted	$O(n)$	$O(n)$
	Sorted	$O(1)$	$O(1)$
Memory		n*sizeof(T) contiguous	n*(sizeof(T)+16) fragmented

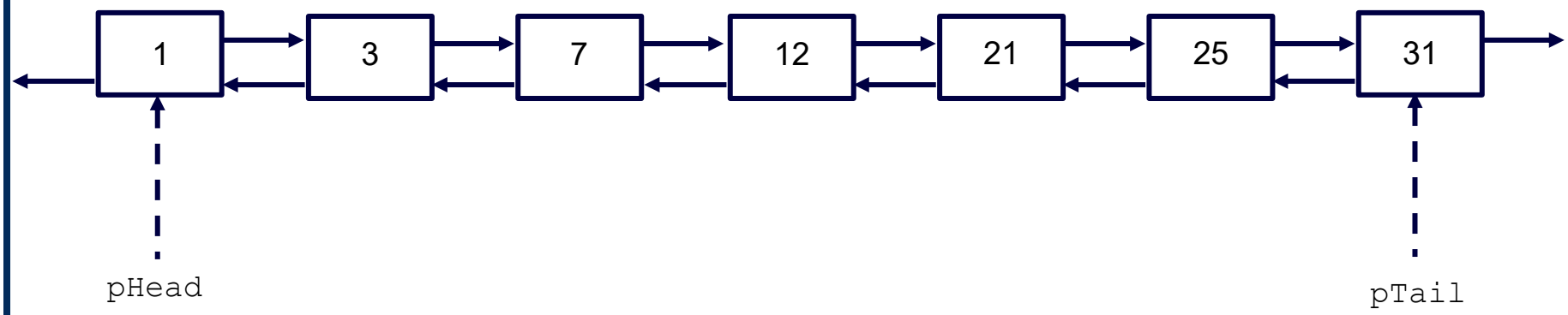
Data Structure Operations

Operation		Array	Doubly-Linked List
Element Access		O(1)	O(*n*)
Traversal	Forwards	O(*n*)	O(*n*)
	Backwards		
Add	Front	O(*n*)	O(1)
	Middle		O(*n*)
	Back		O(1)
Delete	Front	O(*n*)	O(1)
	Middle		O(*n*)
	Back		O(1)
Sort		O(*n log n*)	O(*n log n*)
Search	Linear	O(*n*)	O(*n*)
	Binary		
Min / Max	Unsorted	O(*n*)	O(*n*)
	Sorted	O(1)	O(1)
Memory		n*sizeof(T) contiguous	n*(sizeof(T)+16) fragmented

Data Structure Operations

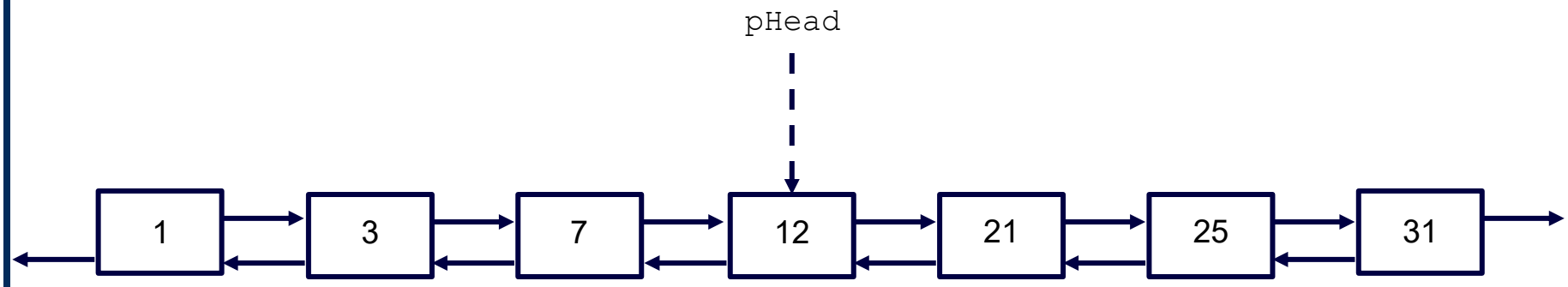
Operation		Array	Doubly-Linked List
Element Access		O(1)	O(*n*)
Traversal	Forwards	O(*n*)	O(*n*)
	Backwards		
Add	Front	O(*n*)	O(1)
	Middle		O(*n*)
	Back		O(1)
Delete	Front	O(*n*)	O(1)
	Middle		O(*n*)
	Back		O(1)
Sort		O(*n log n*)	O(*n log n*)
Search	Linear	O(*n*)	O(*n*)
	Binary	O(*log n*)	O(*n*)
Min / Max	Unsorted	O(*n*)	O(*n*)
	Sorted	O(1)	O(1)
Memory		n*sizeof(T) contiguous	n*(sizeof(T)+16) fragmented

Doubly Linked List



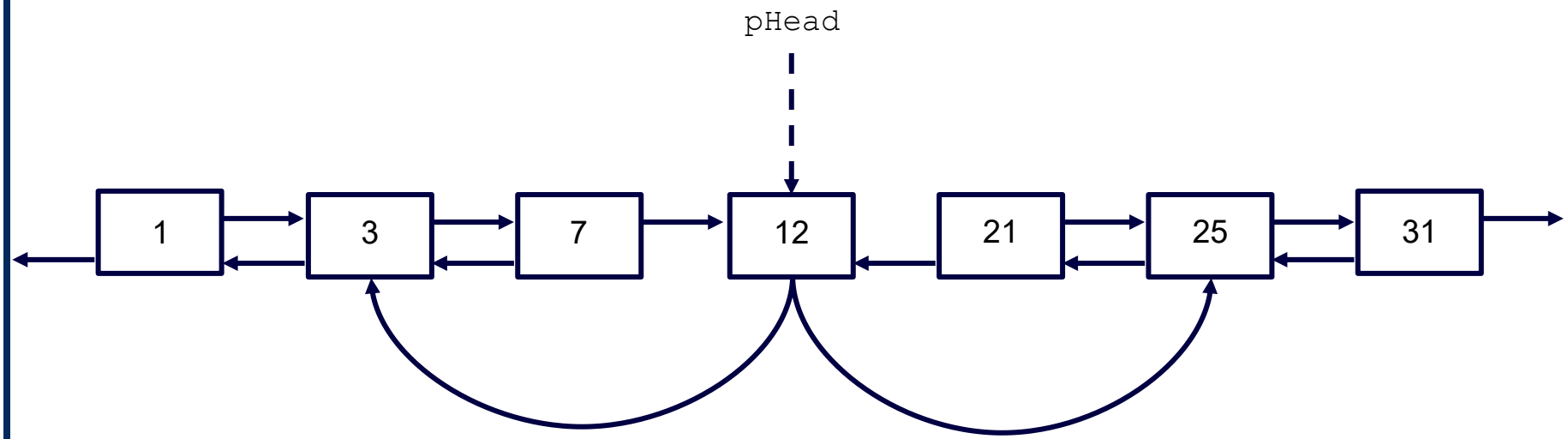
Doubly Linked List

- Point to midpoint



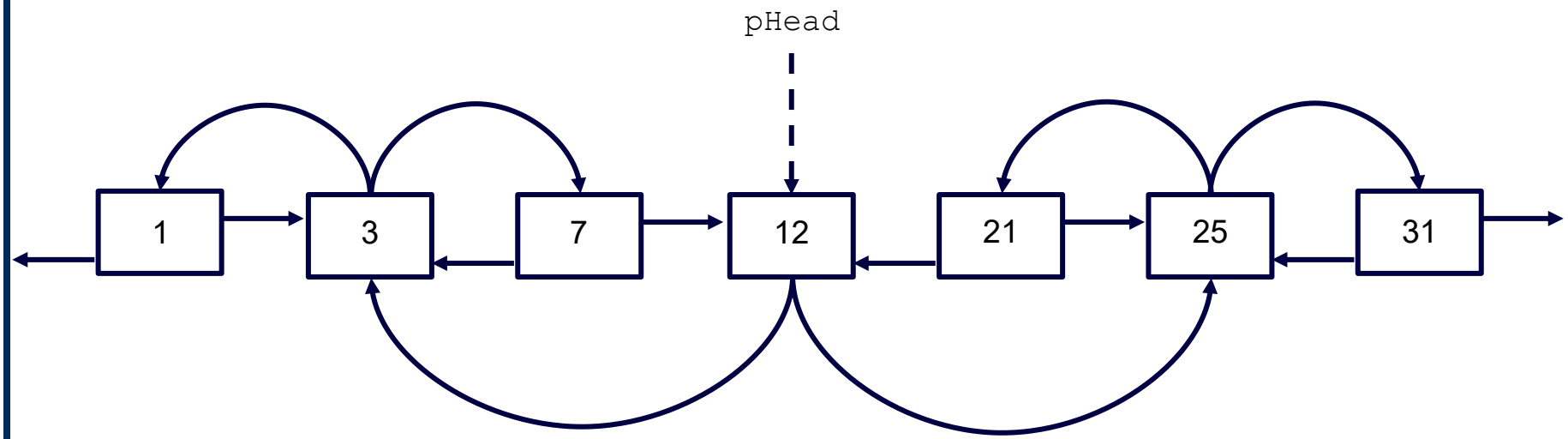
Doubly Linked List

- Point to midpoint
  - Have that point to midpoints



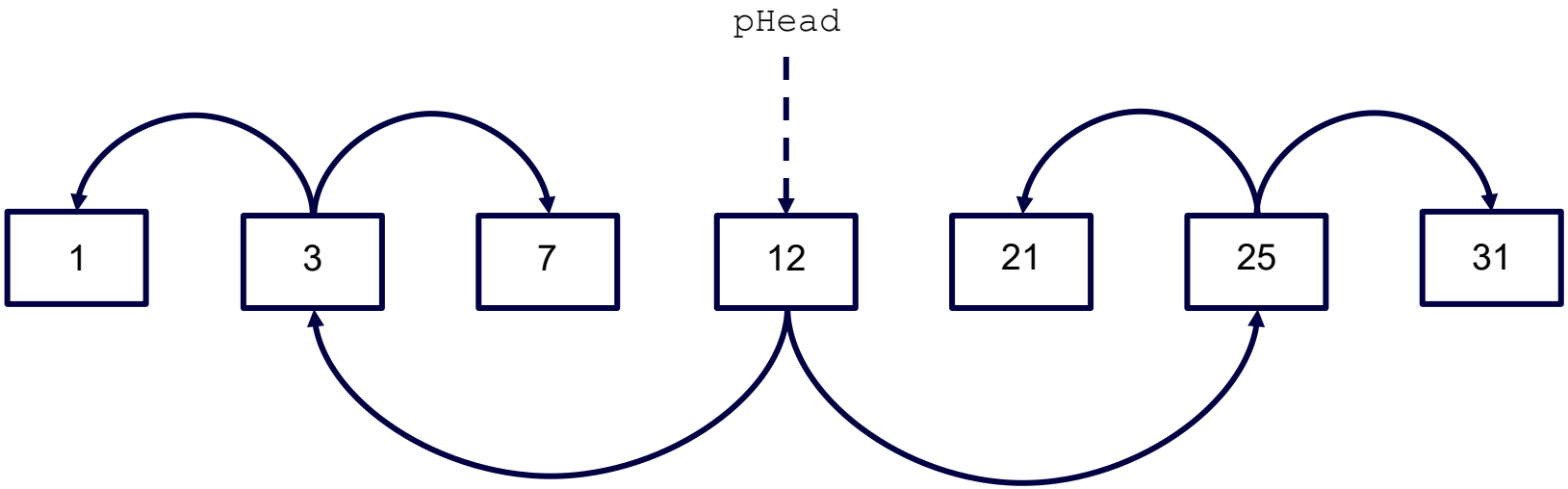
Doubly Linked List

- Point to midpoint
  - Have that point to midpoints
    - And so on

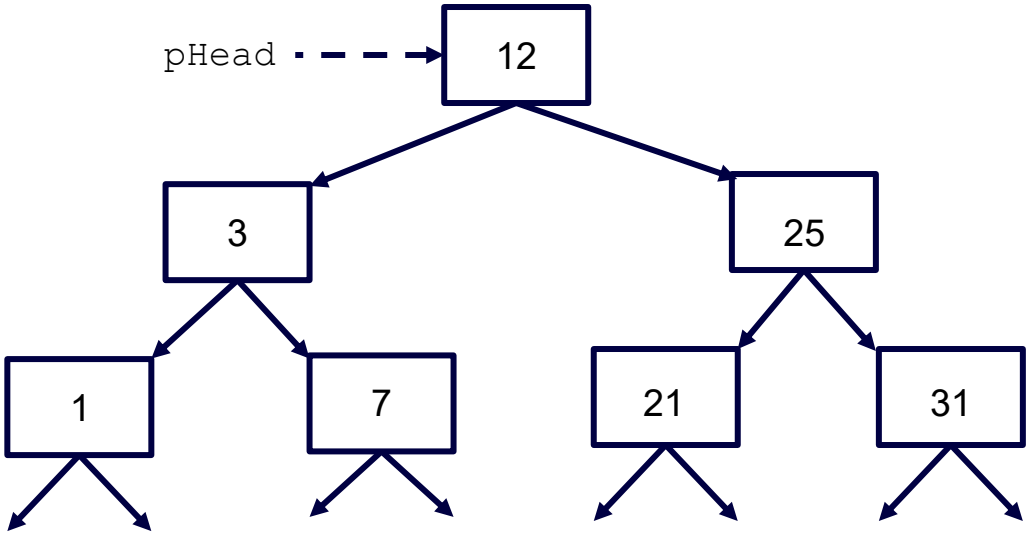


Doubly Linked List

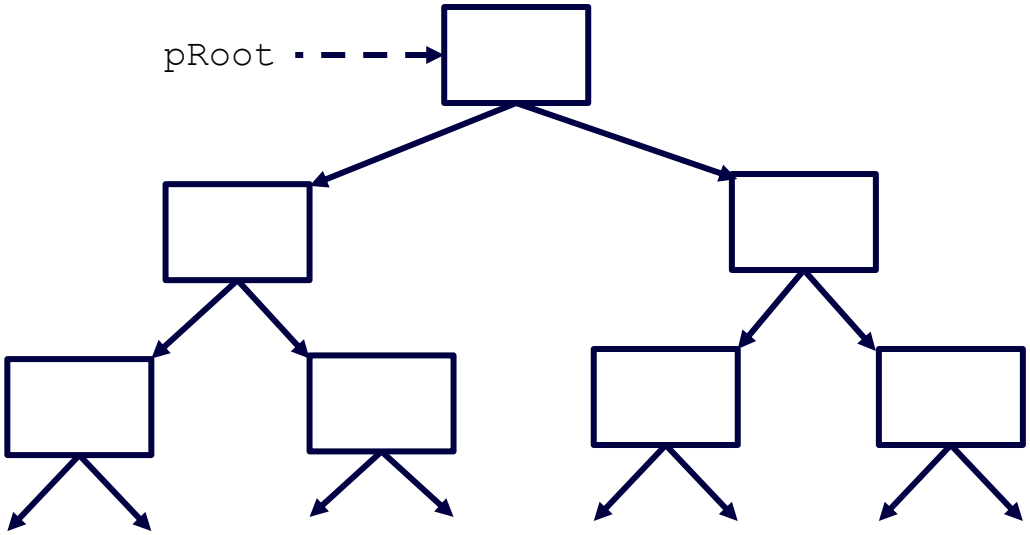
- Cost of “Binary Search”?

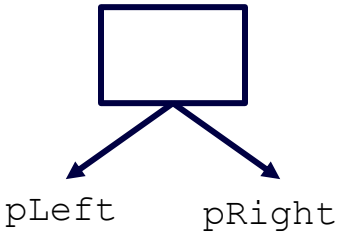


Doubly Linked List



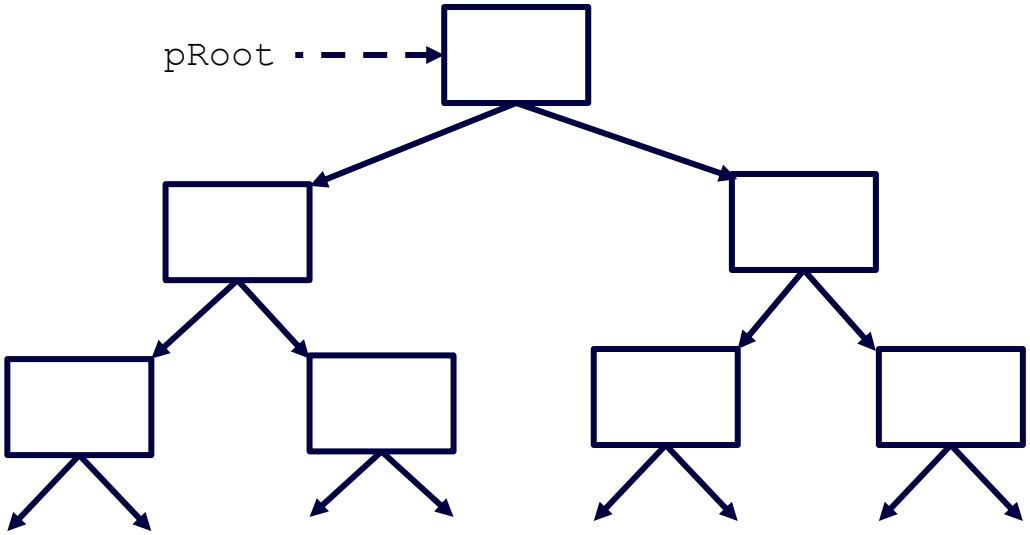
Binary Tree

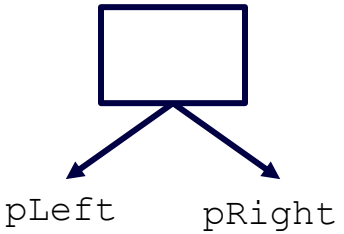




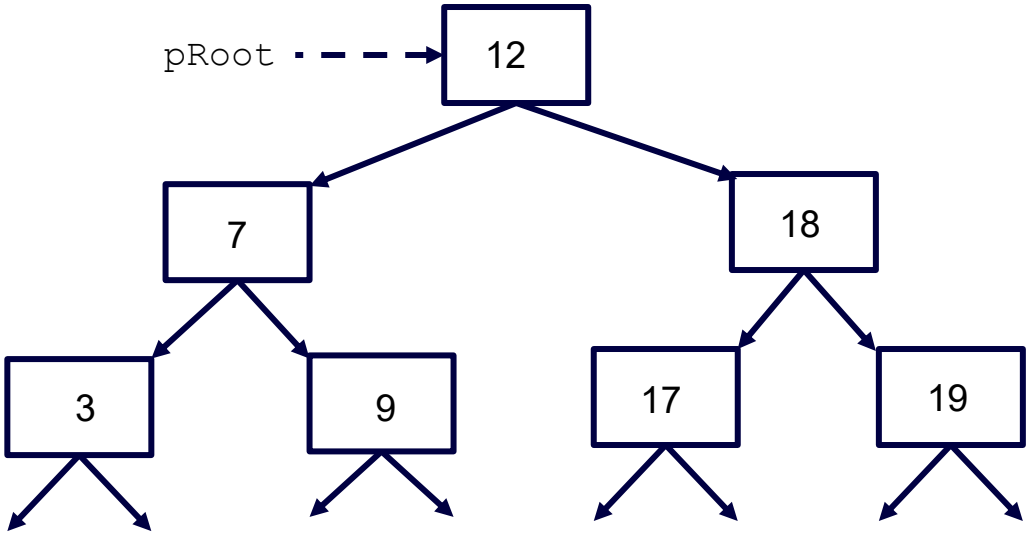
Binary Tree

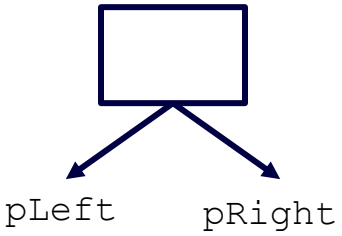
- When adding – apply insertion sort



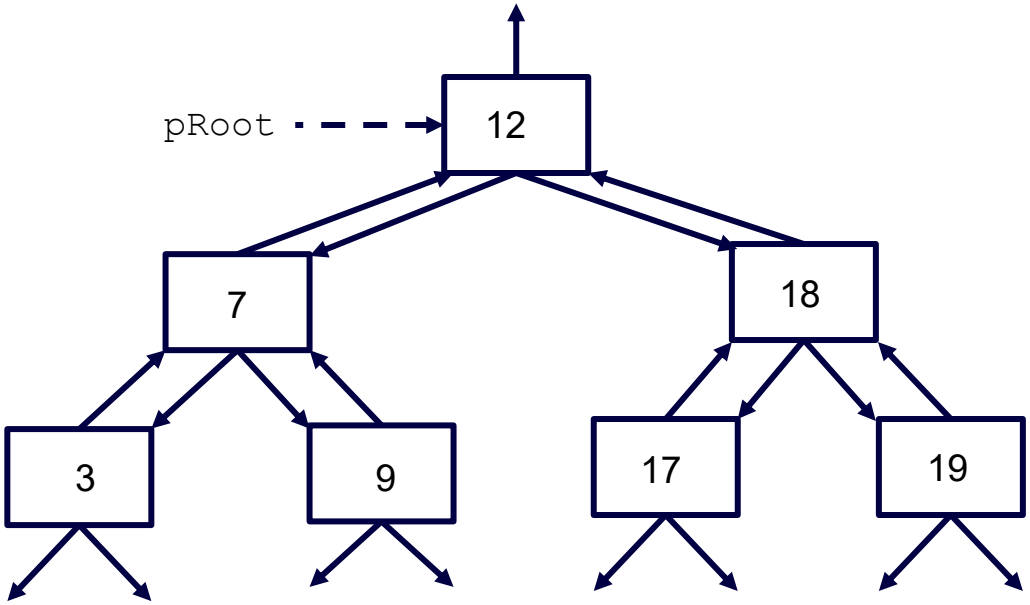


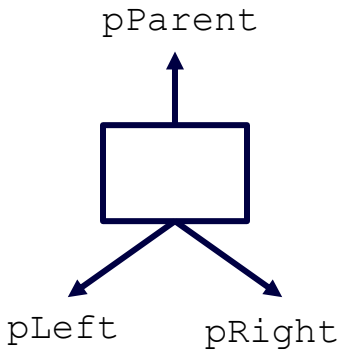
Binary Search Tree





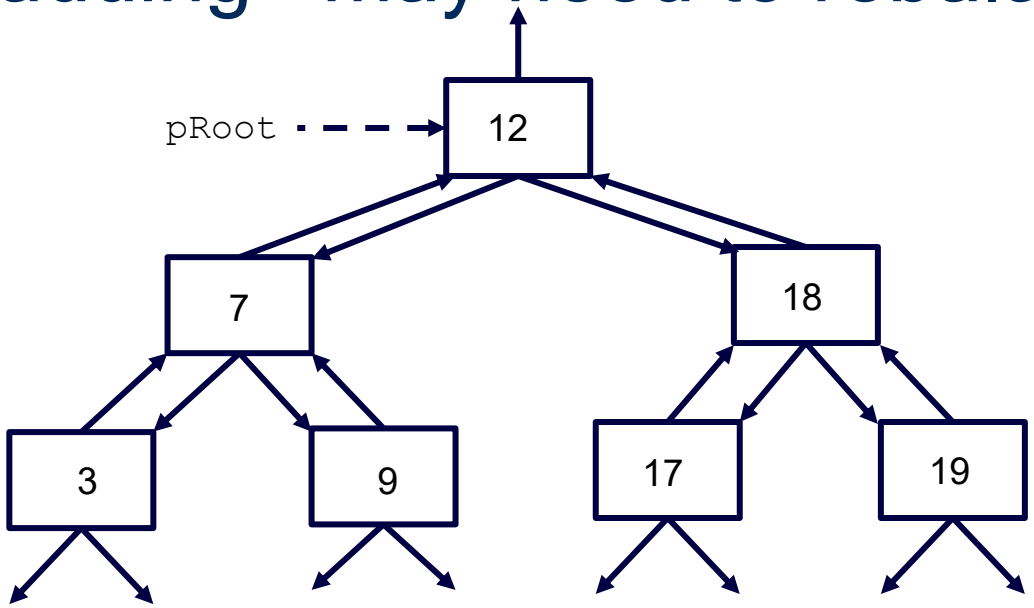
Binary Search Tree



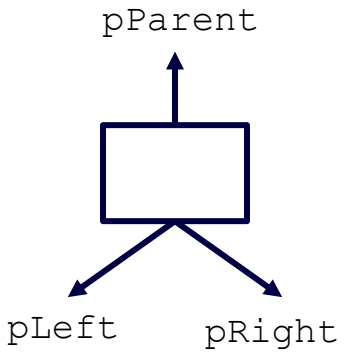


Binary Search Tree

- When adding – may need to rebalance



- **Note: For AXC - not doing a self-balancing tree**

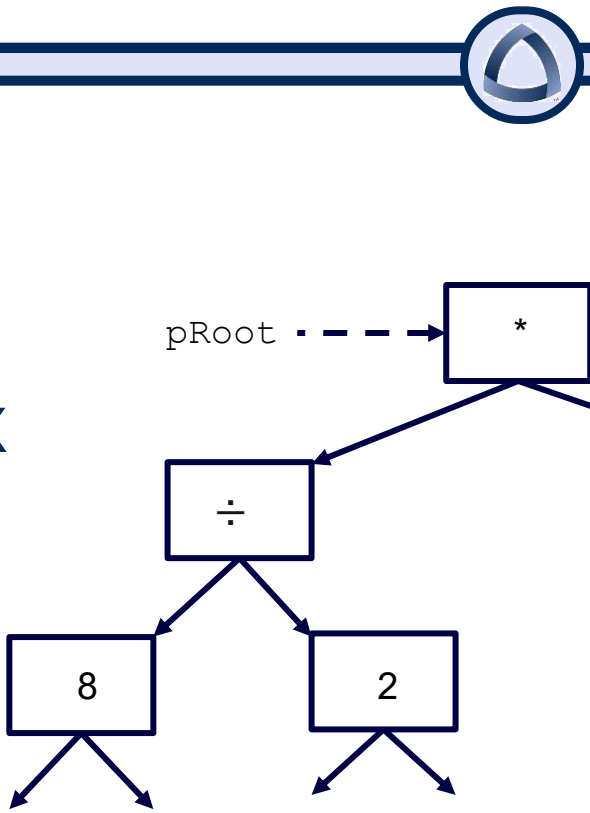


Tree Traversal

-

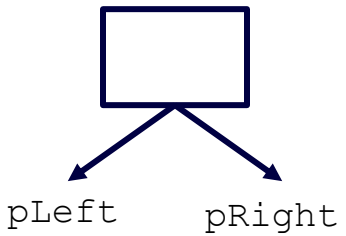
- Solve: $8 \div 2(2+2)$
  - BODMAS / PEMDAS

Tree Traversal



- Infix
- Prefix
- Postfix

- Solve: $8 \div 2(2+2)$
  - BODMAS / PEMDAS



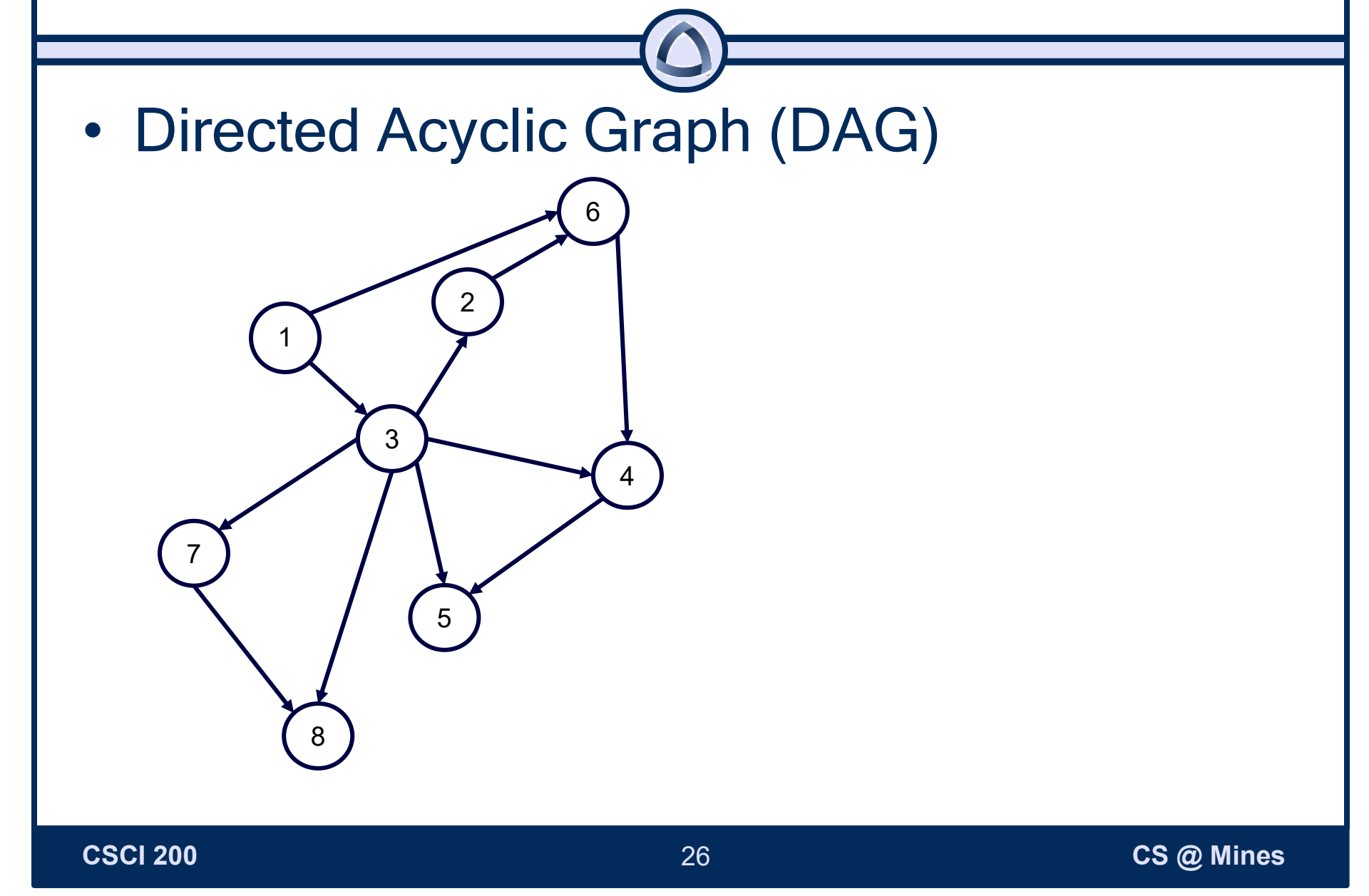
Data Structure Operations

Operation		Array	Doubly-Linked List	Balanced BST
Element Access		O(1)	O(*n*)	O(*log n*)
Traversal	Forwards	O(*n*)	O(*n*)	O(*n*)
	Backwards			
Add	Front	O(*n*)	O(1)	O(*log n*)
	Middle		O(*n*)	
	Back		O(1)	
Delete	Front	O(*n*)	O(1)	O(*log n*)
	Middle		O(*n*)	
	Back		O(1)	
Sort		O(*n log n*)	O(*n log n*)	*N/A*
Search	Linear	O(*n*)	O(*n*)	*N/A*
	Binary	O(*log n*)	O(*n*)	O(*log n*)
Min / Max	Unsorted	O(*n*)	O(*n*)	*N/A*
	Sorted	O(1)	O(1)	O(*log n*)
Memory		n*sizeof(T)	n*(sizeof(T)+16)	n*(sizeof(T)+16)

On Tap For Today

- Trees
- Graphs
- Practice

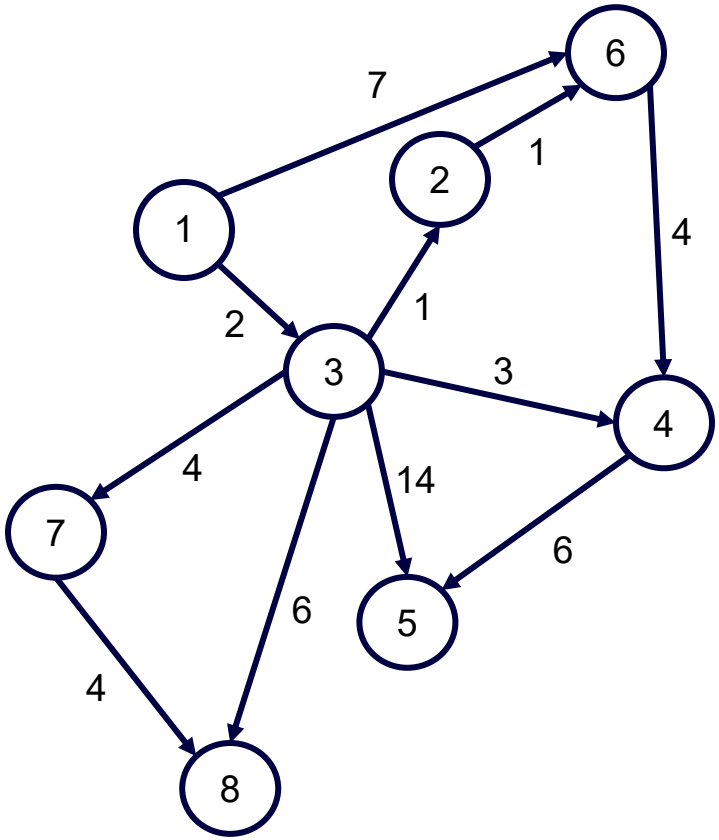
Nodes with *n* Pointers



- Directed Acyclic Graph (DAG)

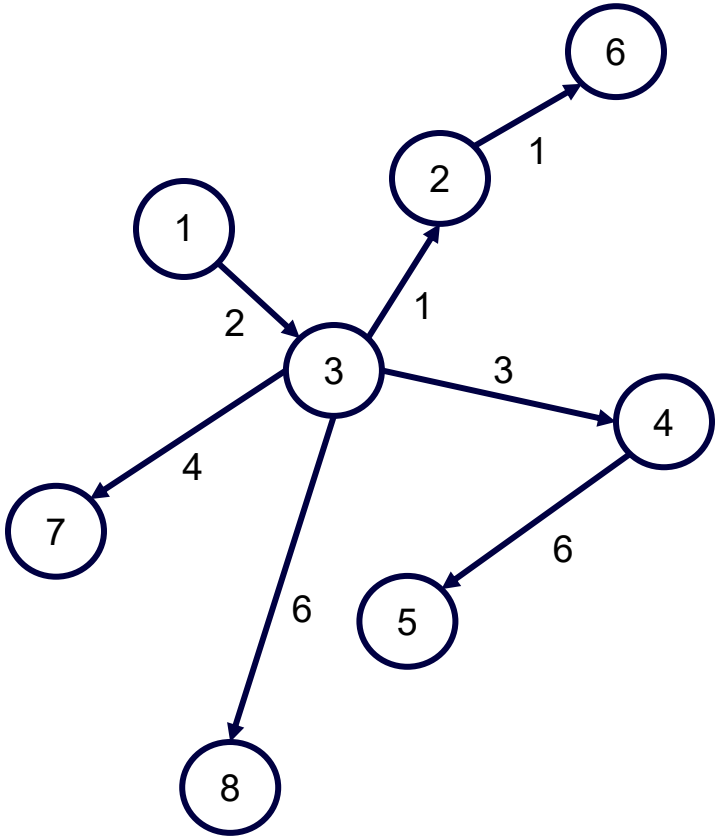
Nodes with *n* Pointers

- Weighted Directed Acyclic Graph



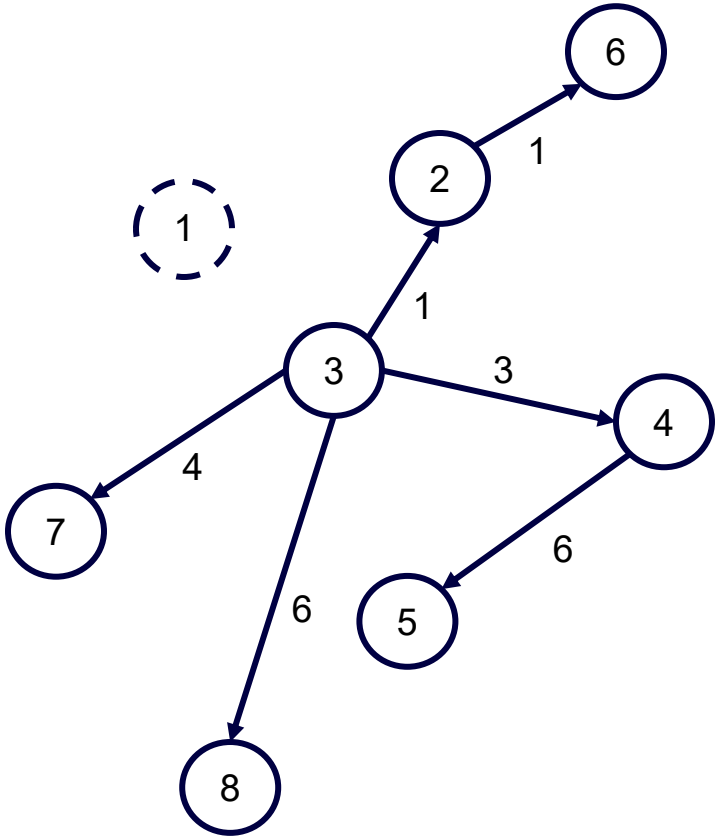
Nodes with *n* Pointers

- Shortest Path Tree from 1 to ? (Djikstra's)

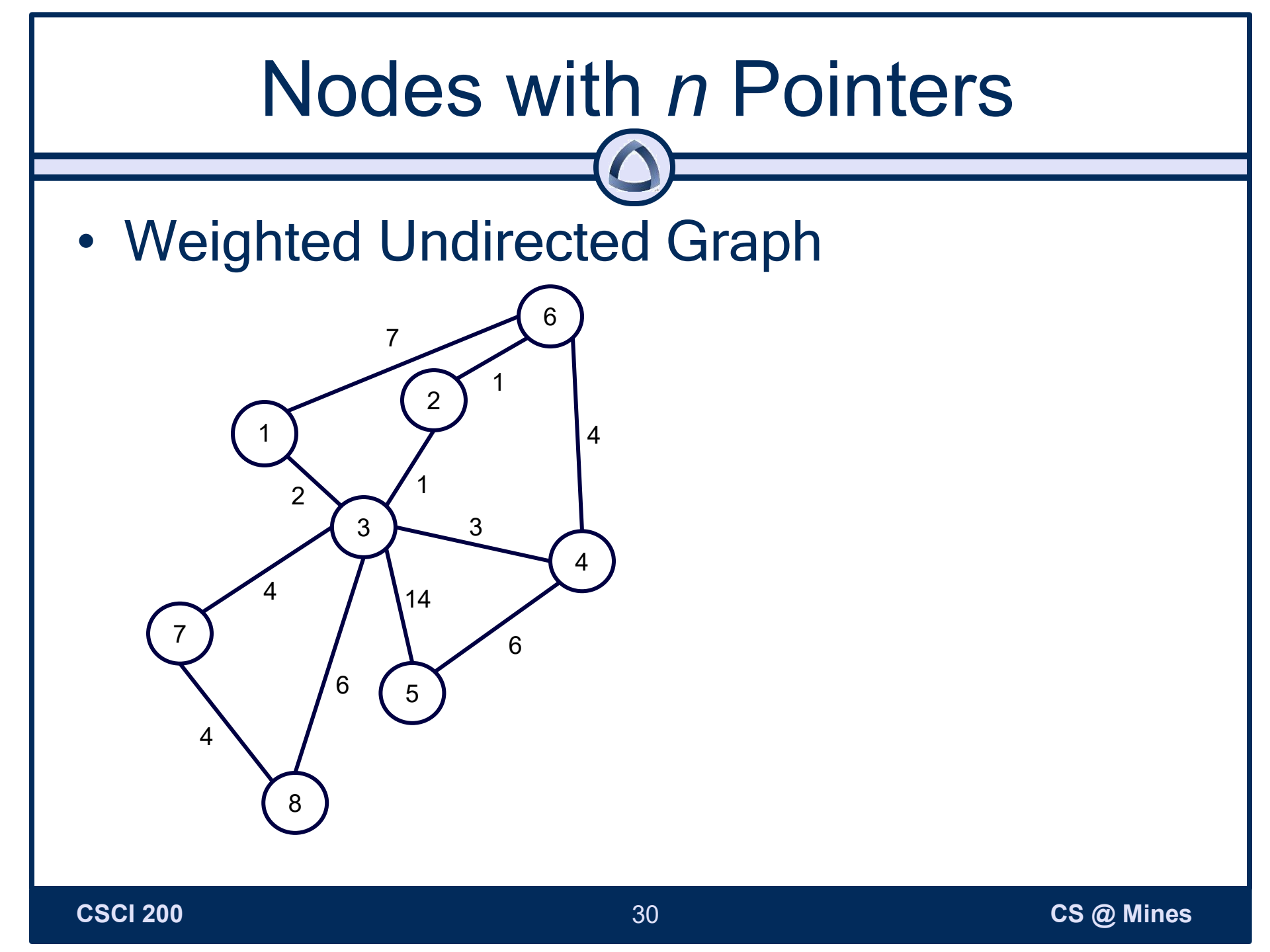


Nodes with *n* Pointers

- Shortest Path Tree from 3 to ? (Djikstra's)



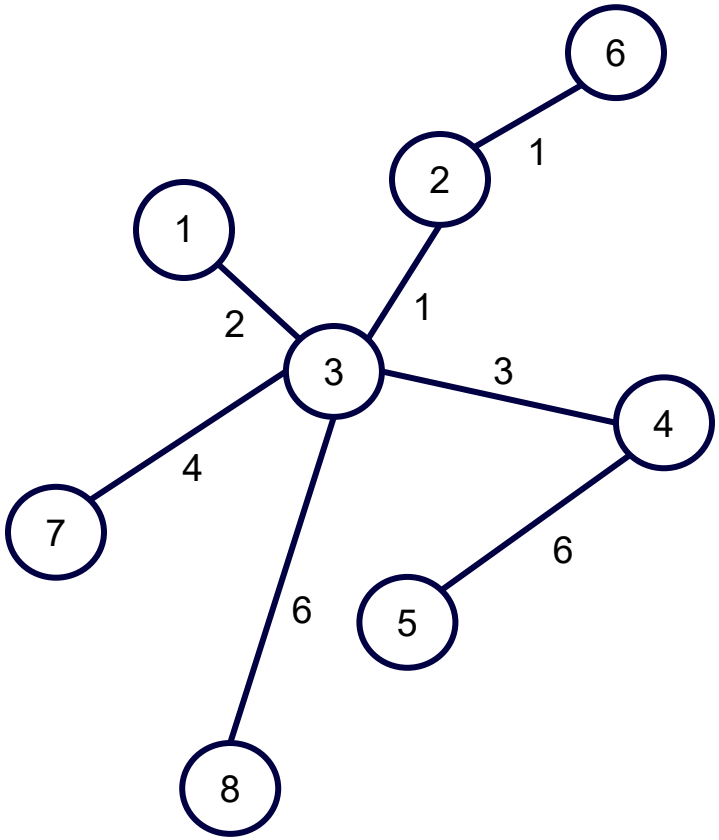
Nodes with *n* Pointers



- Weighted Undirected Graph

Nodes with *n* Pointers

- Minimum Spanning Tree



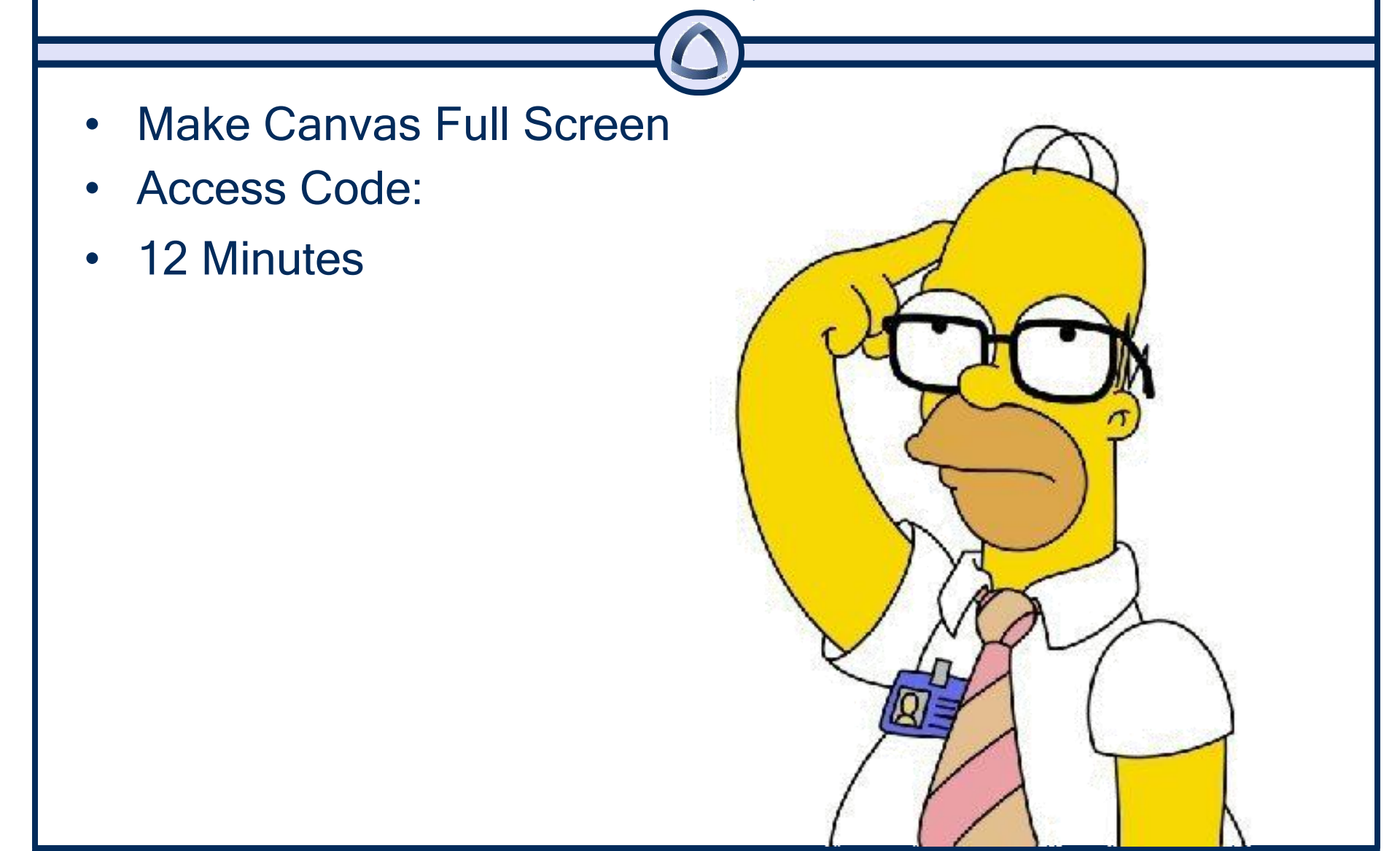
On Tap For Today

- Trees
- Graphs
- Practice

To Do For Next Time

- Rest of semester
  - W 12/10: Exam Review, L6C due, Exam XC due
  - R 12/11: A6, AXC, Final Project due
  - M 12/15 8am – 10am: Final Exam

List Quiz



- Make Canvas Full Screen
- Access Code:
- 12 Minutes