DON'T FORGET MIDTERM: Each chapter contains Self Check questions (answers are provided at the end of each chapter), I would recommend that you go over each of them.

- Chapter Summary is a good place to check your understanding of the material.
- Algorithms that you were implementing in java.
- Exercises section for each chapter that we study.

FOR TRACING LOOK AT PRELABS!!!!!!

POSTFIX??!?!?!

Prefix incrementing before the loop starts

Postfix incrementing after the first loop

Lecture 1

- **Bags**
- Bags PAGE (61)
  - Stars
    - 5, 8, 9, 10, 26, 36, 37, 40, 41, 47, 51, 52, 56
  - ADT STAND FOR
  - FORLOOPs
    - Iterate through information which goes through certain conditions and performs tasks that are stated inside of it
  - Encapsulation
  - Abstract what is an abstract class???
  -

Lecture 2

- **Bag implementations that use Arrays**
- Bag implementations that use Arrays PAGE (89)
  - Stars

        - 3, 4, 5, 6, 7, 8, 11, 20, 32, 33, 34, 35, 36, 37, 43, 44, 46, 52, 53, 55, 57, 58, 59, 62
    - compareTo vs equals method

Lecture 3

- **Bag implementations that use Linked Lists**
- **The efficiency of Algorithms**
- Bag implementations that use Linked Lists PAGE (133)
- The efficiency of Algorithms PAGE (159)
    - Stars
        - 3,4,5,16,22,31,32,40,43,55

Lecture 4

- **Stack and Stack implementations**
- Stack and Stack implementations PAGE (183)
    - **Stars**
        - **10, 11, 19, 20, 25, 28, 31, 32**

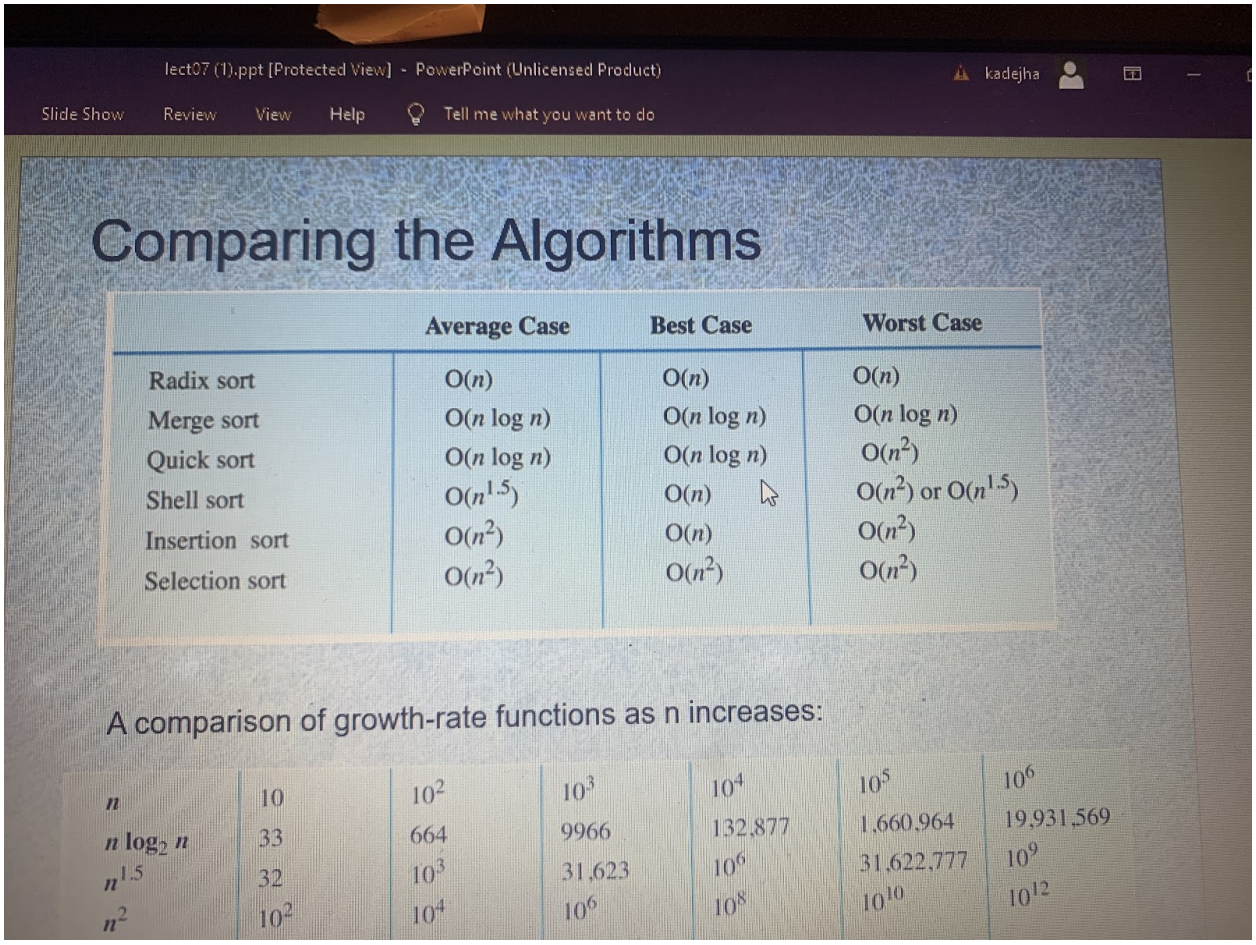
Lecture 5

- **Recursion**
    - Recursion PAGE (277)
    - Stars
        - 5, 6, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 27, 31, 32, 33, 37, 38, 39, 42, 43, 44, 45, 46

Lecture 6

- **Introduction to sorting**
    - Stars**:**
        - **6, 7, 9, 14, 15, 16, 18, 19, 20, 21, 26, 28, 31, 32, 34, 35, 37, 40**
    - Selection Sort PAGE (277)
        -

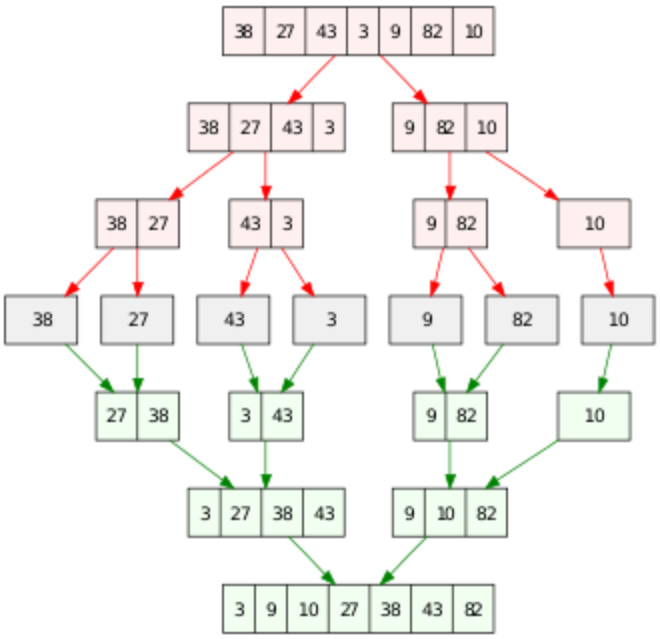
- Insertion Sort PAGE (281)
  -
- Shell Sort (290)
  -
- Bubble Sort
  - Improved Bubble Sort
  - Shaker Improved Bubble Sort



Lecture 7

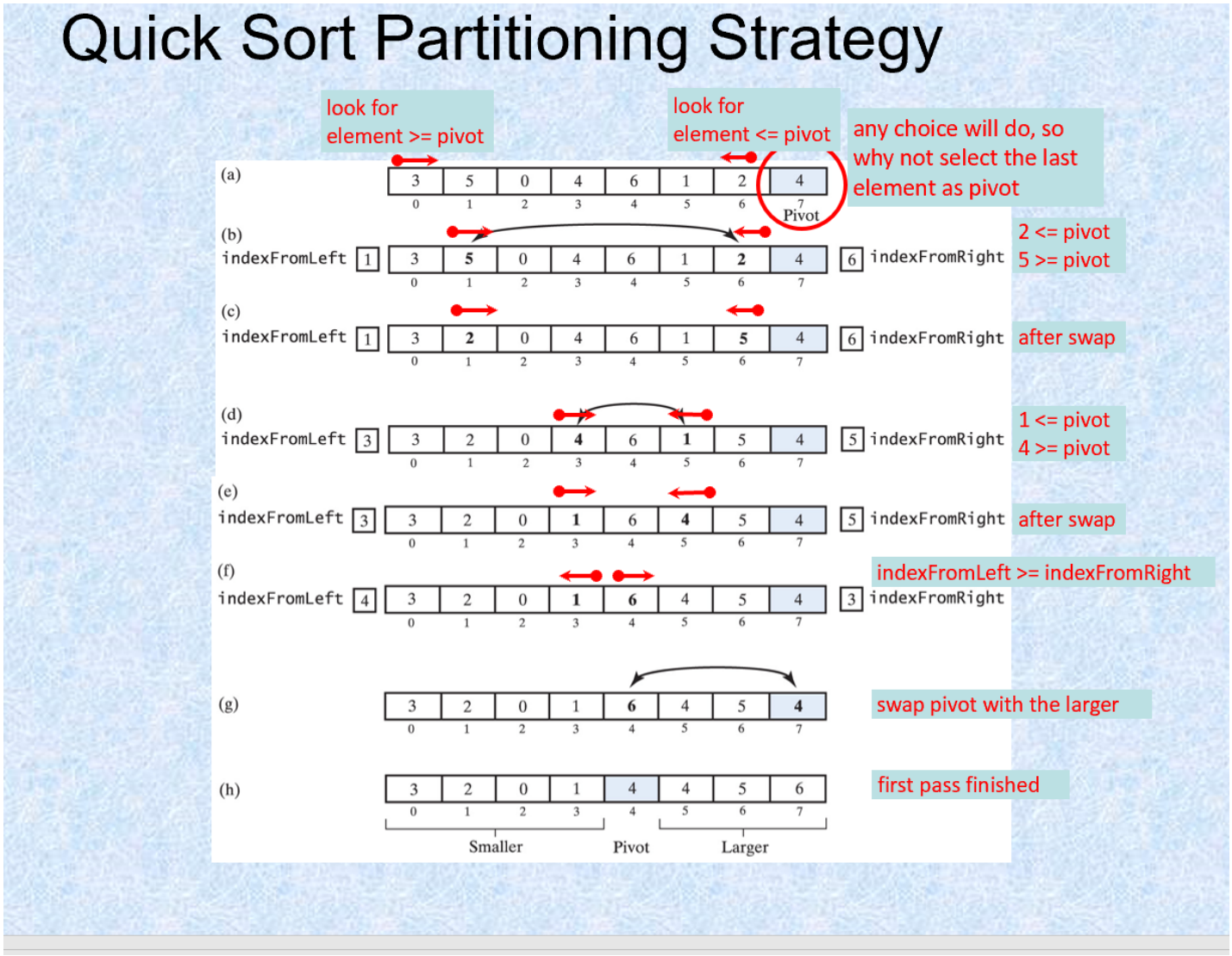
- **Faster Sorting Methods**
  - Stars
    - 8, 12, 14, 15, 16, 19, 20, 22, 23, 25, 29, 31, 33
  - Merge Sort PAGE (301)
    - Recursive merge sort: Ave/Best/Worst case: O(n log n)

- Regular Merge Sort
    - Recursively breaking down into sub-arrays from left to right
    - Once sub-arrays are broken down into sub-arrays of one element merge the two halves together in order.
- **Pseudocode**
    - If the first index is less than the last index…
    - Set the mid = to (first + (last - first)/2)
    - Call this method on the array from the value at first to the value at mid
    - Call this method on the array from the value at mid+1 to the value at last
    - Merge the two halves



- Radix Sort PAGE (314)
    -
- Quick Sort PAGE (308)
    -
        - PsudeoCode

- Divides arrays into three portions (smaller, pivot and larger)
- The pivot selection can be random but it becomes more accurate as more elements are included in its selection (The sort is faster the closer to the center the element is)
- There is an index to the left and index to the right
- Each iteration comparing the indexes to the pivot, and decrementing and incrementing after comparisons
- If index from the left is greater than pivot then swap with an index to the right that is smaller than the pivot.
- If the right index < left index then we found the pivot insertion point
- Swap pivot to the final position of the insertion point
- Then call insertion sort on left and right of the insertion point



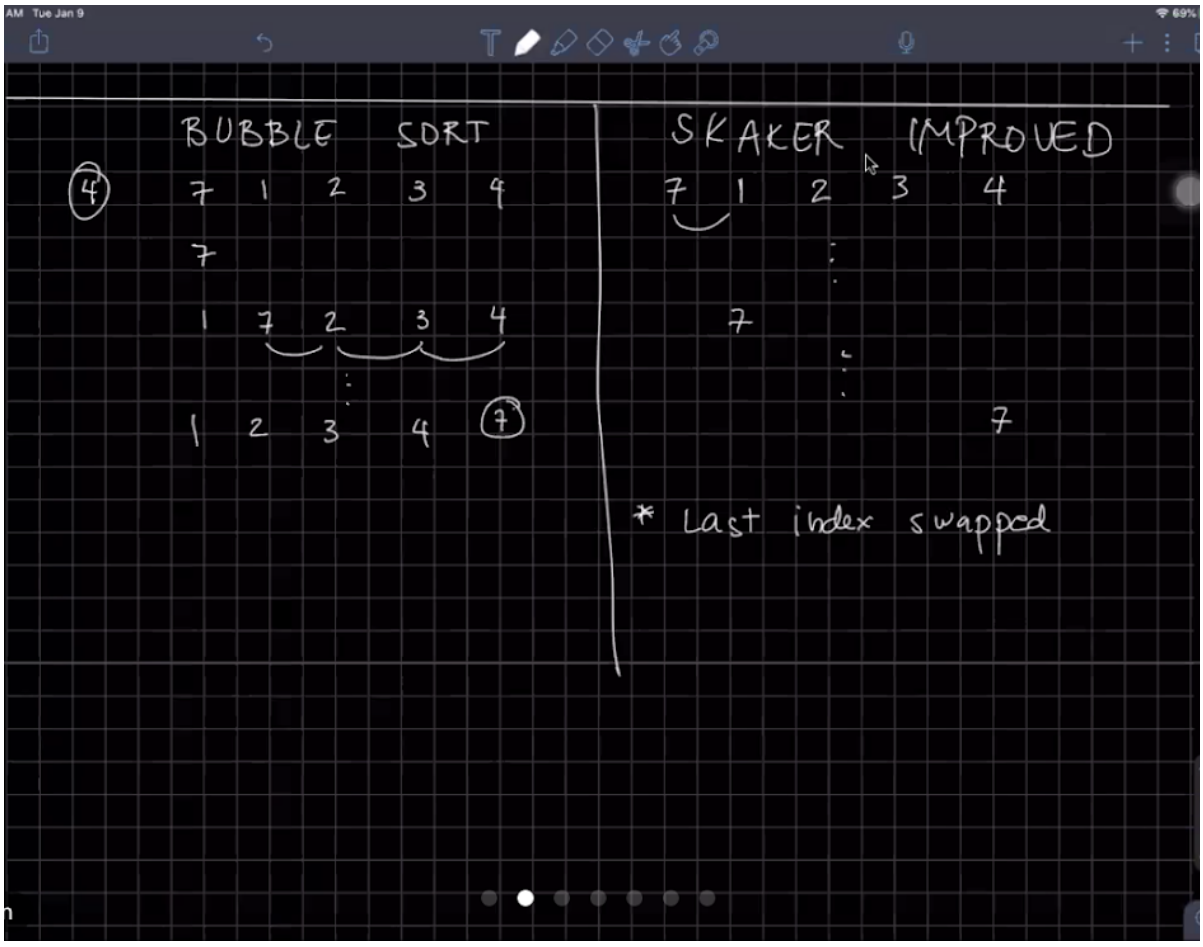
Lecture 8

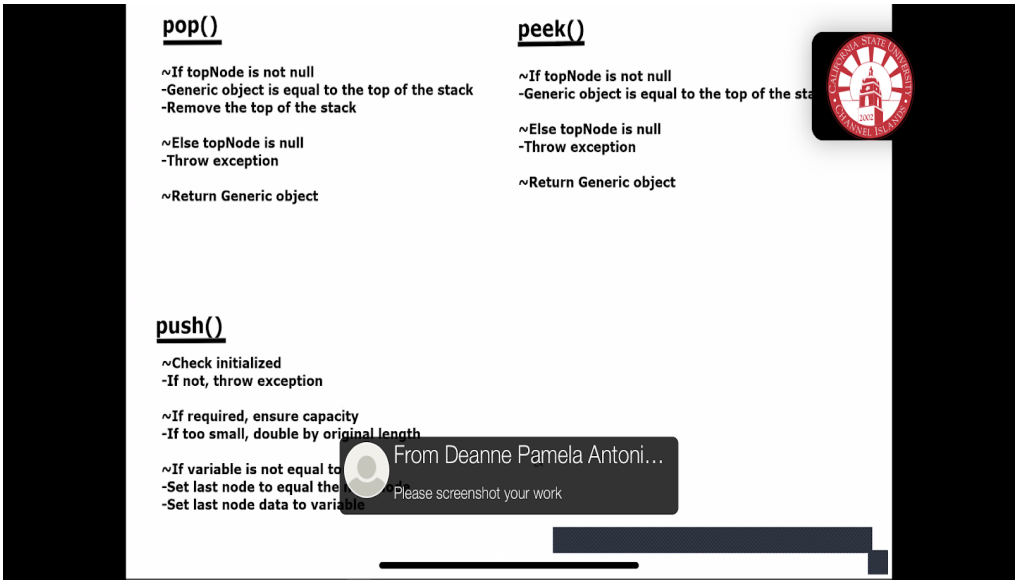
- **Queues, Dequeue, and Priority Queues**
- **Queues, Dequeue, and Priority Queues Implementations**
  - Queues, Dequeue, and Priority Queues PAGE (331)
  - Queues, Dequeue, and Priority Queues PAGE IMP (361)
  - Stars
    - 6, 8, 9, 12, 15, 18, 22, 23, 24, 27, 28, 29, 36, 49, 54
  -

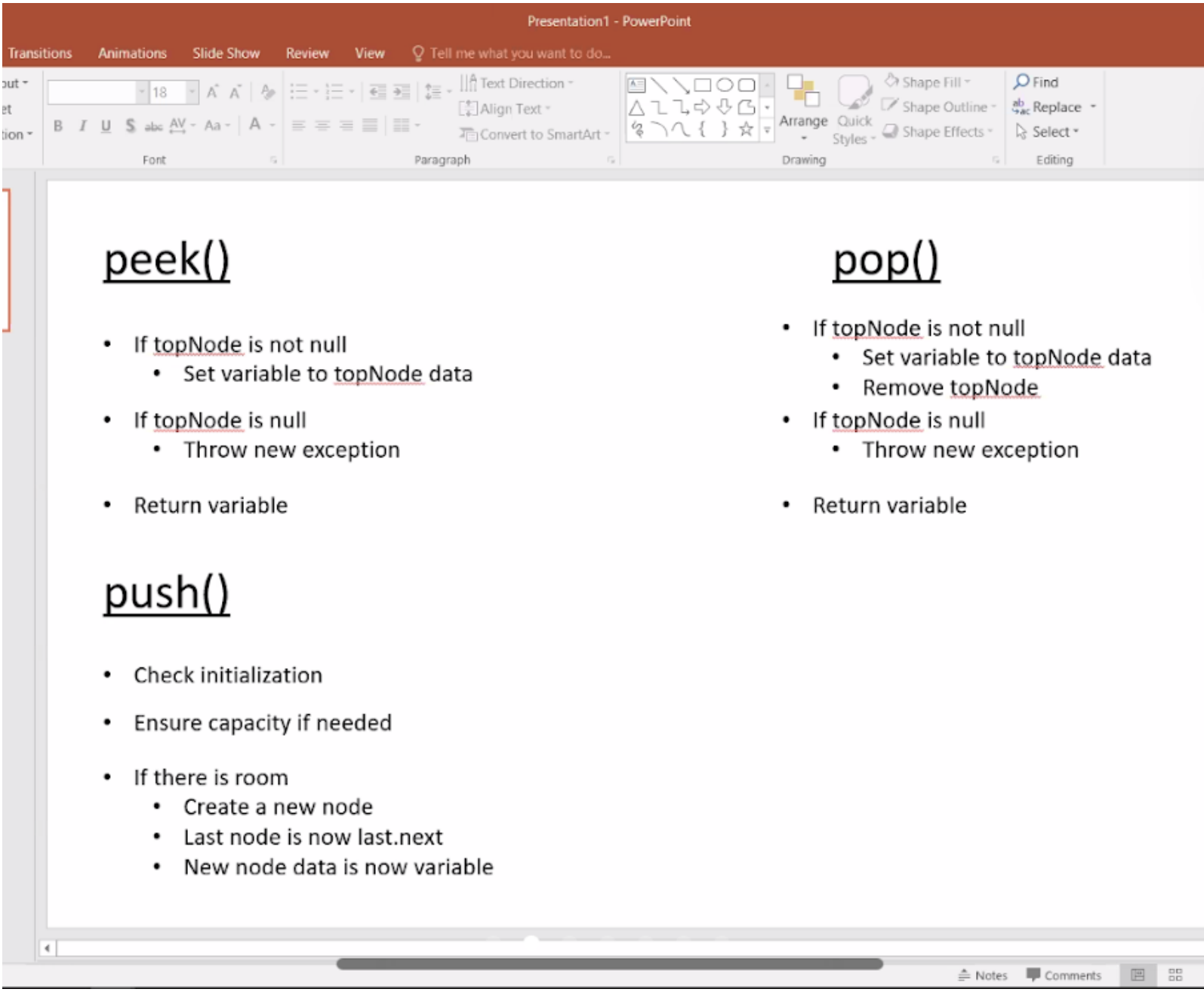
Big (O) Notation

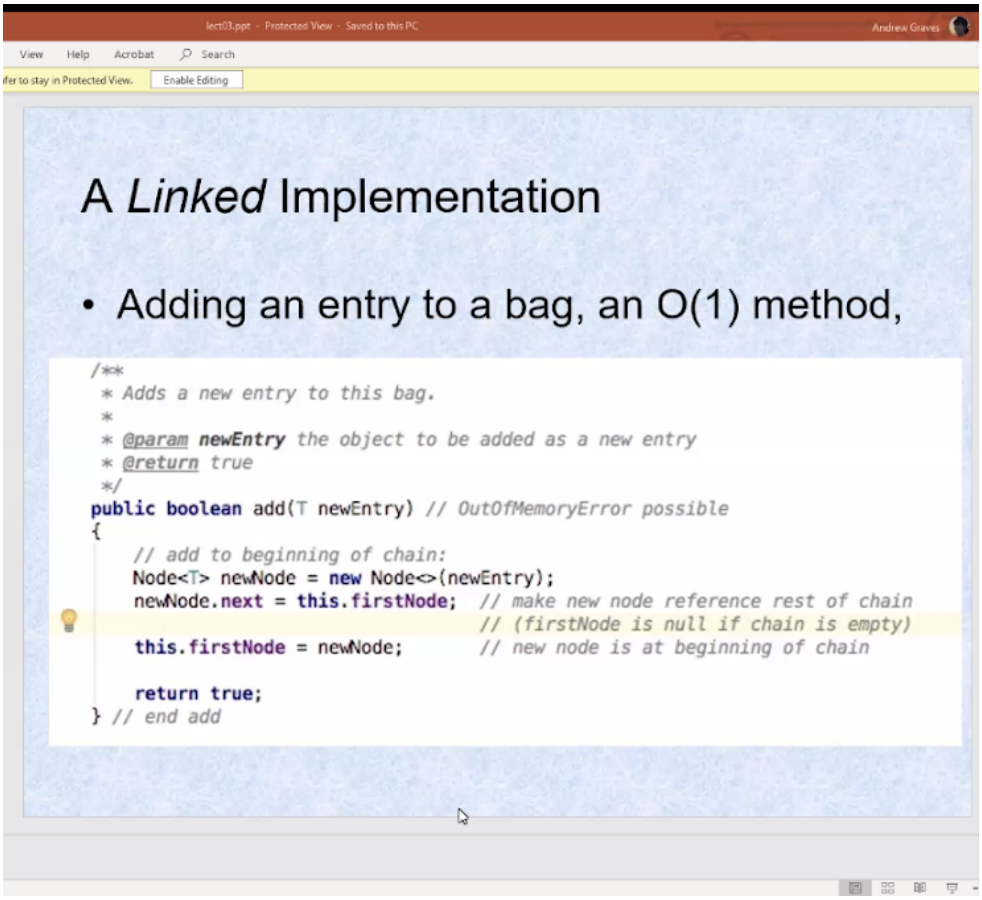
- O(1) - time will always be the same.
    - Also called **bounded time**
    - Ex. assigning a value to the ith element in an array of n elements
    - Ex. adding an entry to a bag
- O(n) - time increases linearly based on the size of data set
    - Also called **linear time**
    - Ex. printing all the elements in a list of n elements is O(n)
    - Ex. searching for an entry in a bag is O(1) best case, O(n) worst/average case so, therefore, it is an O(n) method overall
- O(n^2) - time increases in polynomial time (e.g. nested loops)
    - Also called **quadratic time**
    - Algorithms of this type typically involve applying a linear algorithm n times
        - Most simple sorting algorithms are O(n^2) algorithms i.e **quicksort (only when WORST CASE - if partitions are not roughly equal), bubble sort, insertion sort**
- O(2^n) - time doubles with each data item added
    - Also called **exponential time**
        - **Very costly**
- O(n!) - time grows exponentially due to the number of permutations
    - Also called **factorial time**
        - **Very costly**
- O(log n) - time grows very slowly despite new data being added
    - Also called **logarithmic time**
    - Algorithms that successively cut the amount of data to be processed in half at each step typically fall into this category
    - Ex. finding a value in a list of sorted elements using the binary search algorithm is O(log n)
- O(n log n) - algorithms of this type typically involve applying a logarithmic algorithm n times

- The better sorting algorithms such as quicksort(when BEST CASE), heapsort, mergesort have O(n log n) complexity




pop()
~If topNode is not null
-Generic object is equal to the top of the stack
-Remove the top of the stack
~Else topNode is null
-Throw exception
~Return Generic object
peek()
~If topNode is not null
-Generic object is equal to the top of the sta
~Else topNode is null
-Throw exception
~Return Generic object
push()
~Check initialized
-If not, throw exception
~If required, ensure capacity
-If too small, double by original length
~If variable is not equal to
-Set last node to equal the
-Set last node data to variable
From Deanne Pamela Antoni...
Please screenshot your work


Presentation1 - PowerPoint
Transitions
Animations
Slide Show
Review
View
Tell me what you want to do...
Text Direction
Align Text
Convert to SmartArt
Font
Paragraph
Arrange
Quick Styles
Shape Fill
Shape Outline
Shape Effects
Drawing
Find
Replace
Select
Editing
peek()
If topNode is not null
Set variable to topNode data
If topNode is null
Throw new exception
Return variable
pop()
If topNode is not null
Set variable to topNode data
Remove topNode
If topNode is null
Throw new exception
Return variable
push()
Check initialization
Ensure capacity if needed
If there is room
Create a new node
Last node is now last.next
New node data is now variable
Notes
Comments


lect03.ppt - Protected View - Saved to this PC
Andrew Graves
View
Help
Acrobat
Search
fer to stay in Protected View.
Enable Editing
A Linked Implementation
• Adding an entry to a bag, an O(1) method,
/**
* Adds a new entry to this bag.
*
* @param newEntry the object to be added as a new entry
* @return true
*/
public boolean add(T newEntry) // OutOfMemoryError possible
{
// add to beginning of chain:
Node<T> newNode = new Node<>(newEntry);
newNode.next = this.firstNode; // make new node reference rest of chain
// (firstNode is null if chain is empty)
this.firstNode = newNode; // new node is at beginning of chain
return true;
} // end add