CSCI 132:
Basic Data Structures and Algorithms

Final Exam Review

Reese Pearsall

Spring 2023

Announcements

- Program 5 due **Sunday**

- Fill out the course evaluation

→ Current Response Rate: **91%**

→ **Extra credit has been achieved!**

- Final Exam on Monday (5/8) at **2:00 PM – 3:50 PM** in our normal classroom

- Take some time this week to double check your grades

Meatball wishes you good luck on your final exams

Final Exam Logistics

Same format/rules as the midterm exam

- Bring your laptop if you need

Roughly about the same length

3% will be added to your exam score

Probably wont curve it

1. Basic Java Class Structure
2. Stacks
3. Searching
4. Short Answer
5. Sorting
6. Multiple Choice
7. Recursion
8. Queues

Basic Java Class Structure

- Be able to identify/define instance fields and methods
- Write a constructor
- Understand basic Java keywords
- Write a basic method that does a simple operation

Stacks

- Be able to understand basic stack methods (push pop peek)
- Given code that utilizes a stack, be able to visualize and illustrate the contents of a stack
- Know the running time of stack operations
- Write code the uses a stack

Searching

- Understand the differences between linear search and binary search
- Understand the running times of those algorithms
- Be able to look at code for linear search and binary search and understand what is happening

Short Answer

- Basic Java Classes, Class Structure, Methods, Operations, if statements, loops, OOP
- Basic Linked Lists
- Big-O Notation, How to determine running time of an algorithm
- Stacks
- Queues
- Bubble Sort
- Selection Sort
- Merge Sort
- Quick Sort
- Linear Search/Binary Search
- Recursion

Sorting

- Bubble sort, selection sort, merge sort, quick sort
- Be able to describe/illustrate the steps of these sorting algorithms
- Know the running time for each sorting algorithm
- Know which ones are efficient/not efficient

Multiple Choice

- Basic Java Classes, Class Structure, Methods, Operations, if statements, loops, OOP
- Basic Linked Lists
- Big-O Notation, How to determine running time of an algorithm
- Stacks
- Queues
- Bubble Sort
- Selection Sort
- Merge Sort
- Quick Sort
- Linear Search/Binary Search
- Recursion

Recursion

- Given a basic recursion function, derive the output and number of recursive calls made
- Understand how to calculate the running time of a recursive algorithm
- Understand limitations/benefits of recursion

Queues

- Be able to understand basic queue methods (enqueue dequeue peek)
- Given code that utilizes a queue, be able to visualize and illustrate the contents of a stack
- Know the running time of queue operations
- Write code the uses a queue

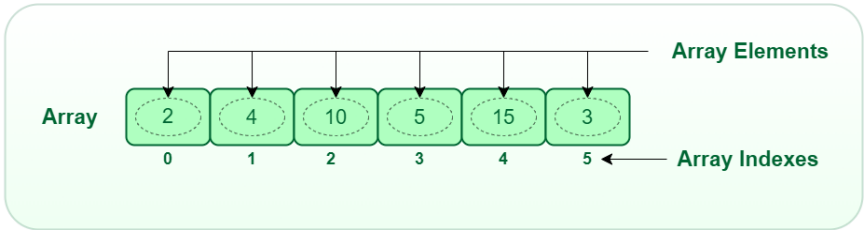
Final Exam Study Guide

Course Goals

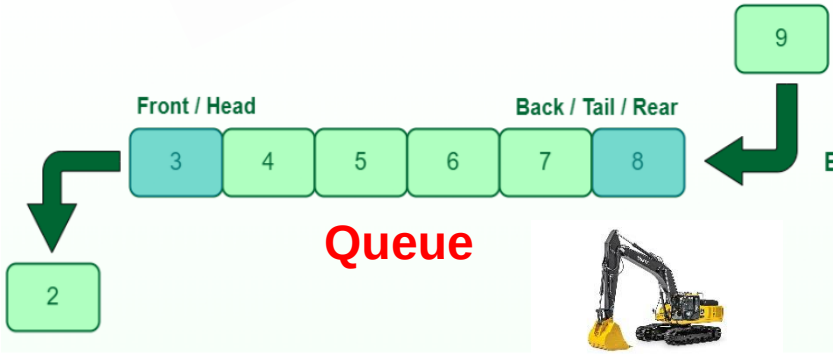
- Design and Implement programs of simple and moderate complexity in Java
- Explain the concept of an ADT *(meh)*
- Understand and implement basic data structures: Linked lists, stacks, and queues
- Given a simple algorithm, determine the time complexity using Big-O notation
- Understand basic searching and sorting algorithms and their runtime
- Understand how recursion works, be able to analyze recursion runtime, and be able to implement recursion in a program
- Be able to debug programs and become an independent problem solver

Takeaways

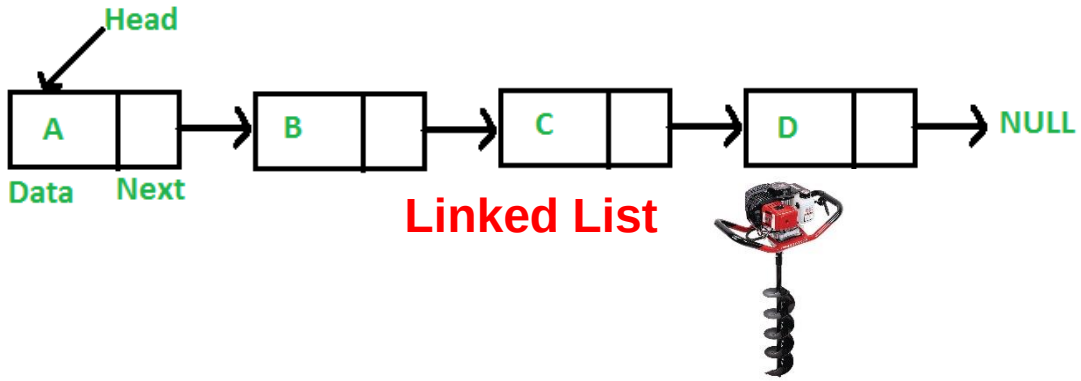
We have different data structures that handle data differently. There are **tradeoffs** between using these data structures



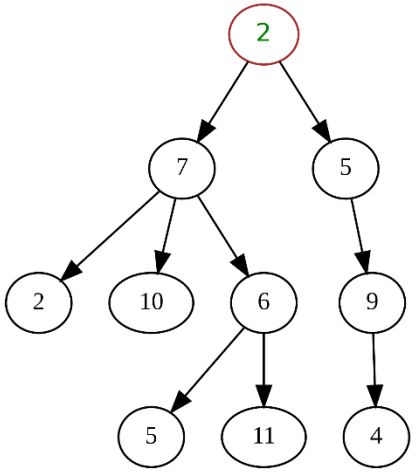
Arrays



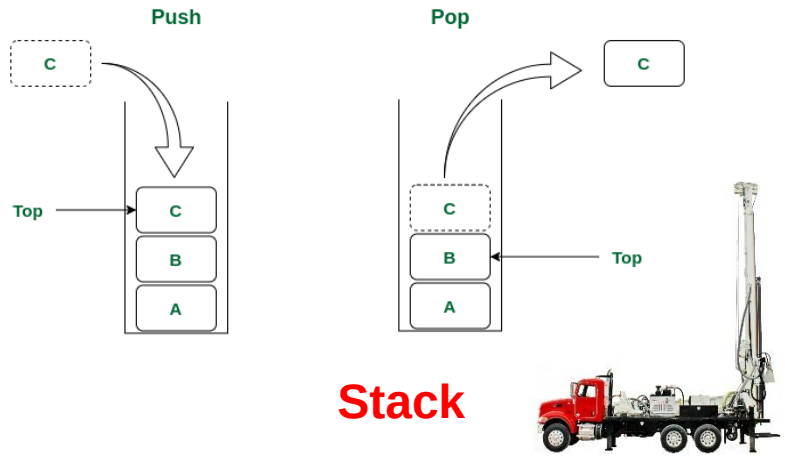
Queue



Linked List



Trees



Stack

Given a problem, you should be able to identify a good candidate for a data structure and provide a justification

Takeaways

- There be many different types of algorithms.
- Some algorithms are much more efficient than others
- The algorithm you select is important. It can be the difference between your program finishing in 6 seconds, or you program *never finishing at all*
- We have methods for measuring the efficiency of some algorithm (big-O notation).
- When you write an algorithm, you should be able to broadly describe the effectiveness and efficiency of it

Takeaways

Bubble Sort	Iterate through array and swap pairs of numbers. Large numbers ("bubbles") will rise to the top naturally	$O(n^2)$
Selection Sort	Iterate through the array and find the minimum value **n** times, and place minimum in correct position	$O(n^2)$
Merge Sort	Use recursion to split array in sub-arrays of size. Sort the sub-arrays while merging until you solve the original problem	$O(n \log n)$
Quick Sort	Partition array around a pivot value. Use recursion and place pivot in correct spot and repeat until array is sorted	$O(n^2)$** **Put usually performs much better (O(n log n())

We cannot sort faster than O(n log n). There are no (known) algorithms that can sort in O(n), O(logn), or O(1) … why?

My Goals for you

Get you comfortable with writing basic Java programs

Give you a good toolset that can help you solve a variety of problems (Data Structures)

Give you techniques and methods for solving a variety of problems (Algorithms)

Give you the skills to analyze the algorithms that you write (Big-O notation)

Thank You!

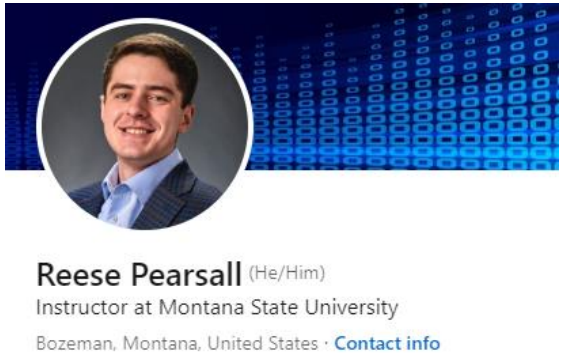
This class has been a blast to teach. Thank you for your patience, flexibility, kindness, and for laughing at my jokes ☺

There were a lot of long nights, and I know things were not perfect, but I am happy with how things went

I hope you enjoyed this class, and I hope the stuff you learned will be helpful in your career/future classes

If I can be of assistance to you for anything in the future (reference, advising, support), please let me know!

I will be teaching CSCI 460, 466, and 132 next semester



Connect with me on LinkedIn!

If you are taking 232 with me this Summer, I'll see you again soon ☺

Congrats to those that are graduating next weekend! I hope you find a job that you love!