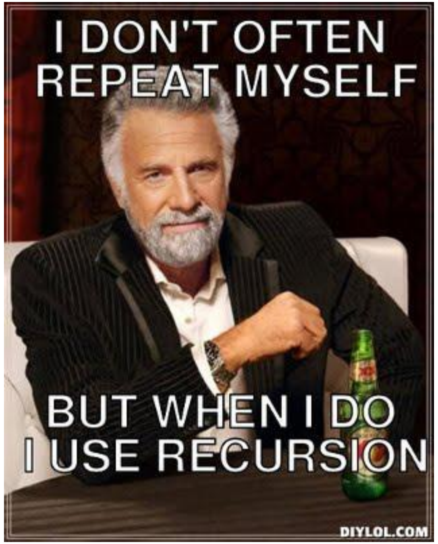
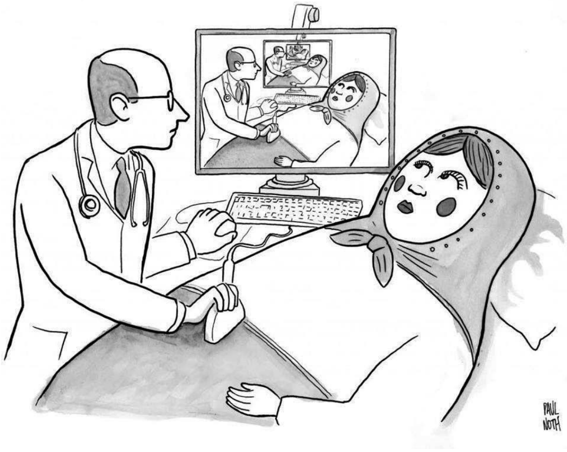
Sorting Videos!

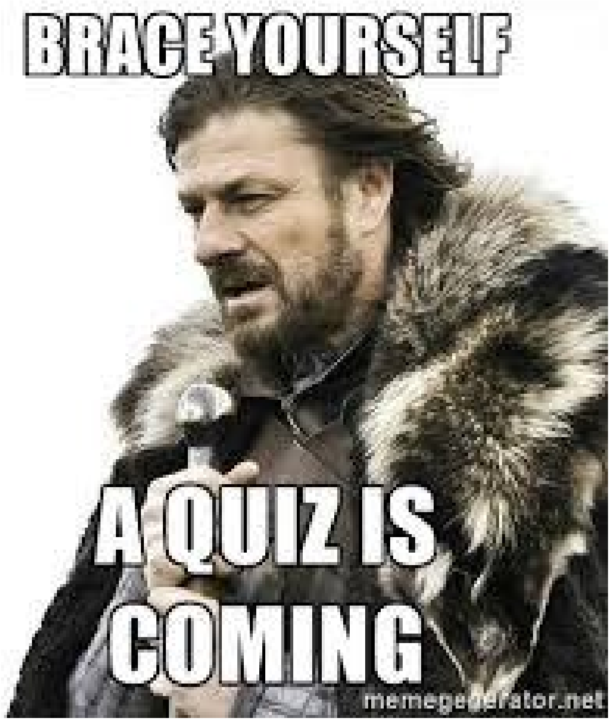




Warm-Up!

Recursion Worksheet #4

Quiz is Coming! Recursion on Thursday (end of class)



This week is our final topic!
Searching and Sorting Lists of Data

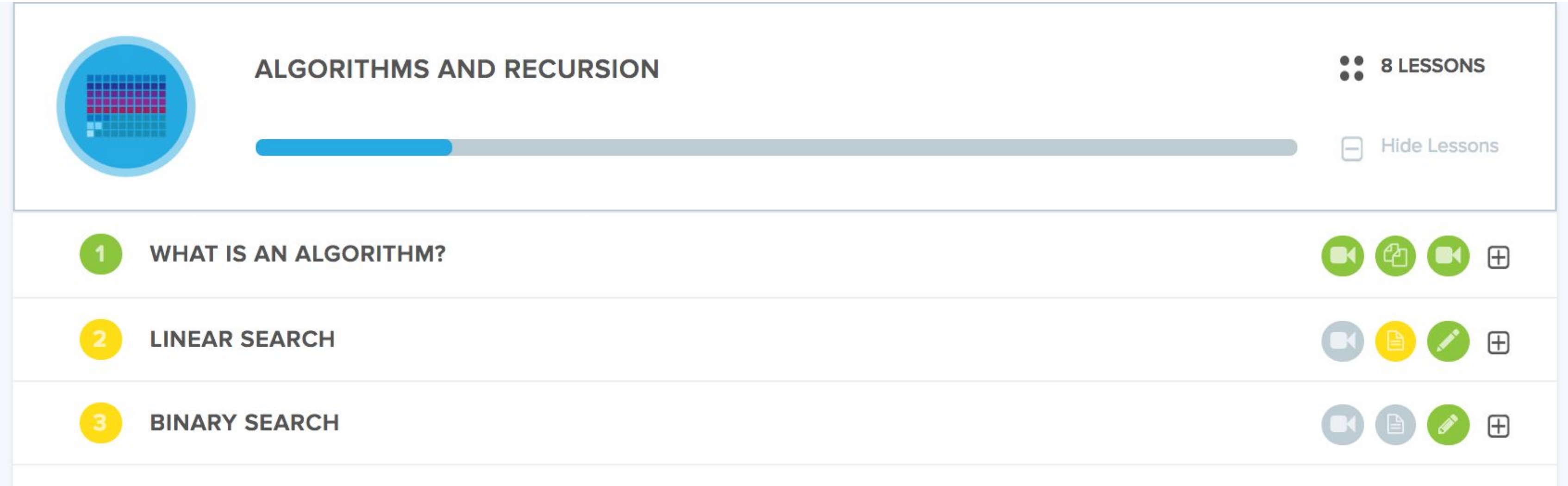
Sort/Search Run-Time

The amount of memory it will take to sort or search an array of n elements. It is denoted as "O(n)", the average sort time.

OK: $O(n^2)$

BEST: $O(n \log n)$

Homework


ALGORITHMS AND RECURSION
8 LESSONS
Hide Lessons
1
WHAT IS AN ALGORITHM?
2
LINEAR SEARCH
3
BINARY SEARCH

The less efficient but easy to understand sorts

$O(n^2)$:

- Bubble
- Insertion
- Selection

These make n passes through an array of length n to compare two elements at a time. They have long names which makes it easier to remember they are long sorts.

Warm-Up! Coding Bats

Recursion #1

- **CountHi**
- **ChangeXY**

Linear Search vs Binary Search

Linear Search vs Binary Search

	Linear	**Binary**
Rules	Can be used on any list	Can only be used on sorted lists
Big-O	n^2	log n

Bubble Sort: run-time $O(n^2)$

1) Start at the beginning (or end)
2) The first element looks at the second element. If they are out of order, they switch.
3) The second element looks at the third element. If they are out of order, switch. ETC
4) The biggest element will “float” to the top (or bottom) and be done
5) Start the process over again at the second element.
6) Keep going

Bubble Sort - Reverse order (Descending)

Selection Sort: run-time $O(n^2)$

1) Start at the very beginning of n elements
2) Find the smallest element between 0 and n-1. Put it at index 0.
3) Find the smallest element between 1 and n-1. Put it at index 1.
4) Find the smallest element between 2 and n-1. Put it at index 2.
5) Keep going through n passes

Selection Sort

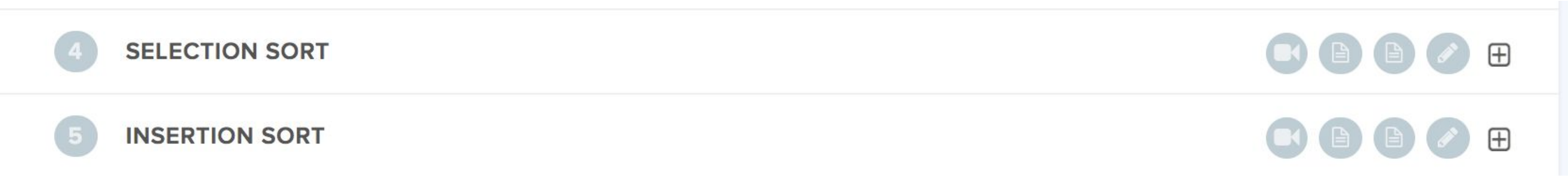
Insertion Sort: run-time $O(n^2)$

1) Put elements 0 and 1 in the right order
2) Take element 2. Put it in the right order with 0 and 1.
3) Take element 3. Put it in the right order with 0,1, & 2.
4) Keep going through element n-1

Insertion Sort

Homework

Code HS, Algorithms



Sort/Search Packet Worksheets #1,2

Warm-Up Part 2

Recursion #1 Coding Bat

- ChangePi
- NoX
- Array6

Warm-Up: AP Sort Question

Given this sorted list; 1, 3, 6, 7, 10, 12, 19

a) **How many comparisons will it take to find the value 10 in a *linear* search?**
b) **How many comparisons will it take to find the value 10 in a *binary* search?**

Warm-Up: AP Sort Question

Given this sorted list; 1, 3, 6, 7, 10, 12, 19

a) **How many comparisons will it take to find the value 10 in a *linear* search? 5**
b) **How many comparisons will it take to find the value 10 in a *binary* search? 3**

Warm-Up: AP Sort Question

Given this unsorted list; 1, 3, 6, 7, 10, 6, 19

What number will never be found?

Warm-Up: AP Sort Question

Given this unsorted list; 1, 3, 6, 7, 10, 6, 19

What number will never be found? 10

More efficient but harder to understand sorts

O(n log n): Merge Sort, Quick Sort

These are “DIVIDE AND CONQUER” sorts that RECURSIVELY divide the array into smaller and smaller arrays to be sorted.

Merge Sort

Quick Sort: run-time O(n log n)

1) Very similar to merge sort, except the array size is not necessarily *half.* (There are different ways to do this)
2) Instead, choose a PIVOT point that divides the array
3) Recursively sort the array to the left of the pivot point
4) Recursively sort the array to the right of the pivot point
5) Keep going until you hit the base cases (you get to a sorted array)
6) Merge the sorted arrays

Quick Sort

SEARCH!!!!

Sequential: Start at 0 and check each element until you find the element you need.

Pros: Works on ANY list, easy to code

Con: O(n)

Binary: Take a *sorted* list, look in the middle. If the value you want is less, look halfway down. If it's more, look halfway up. Keep halving until you find your value. 'DIVIDE AND CONQUER"

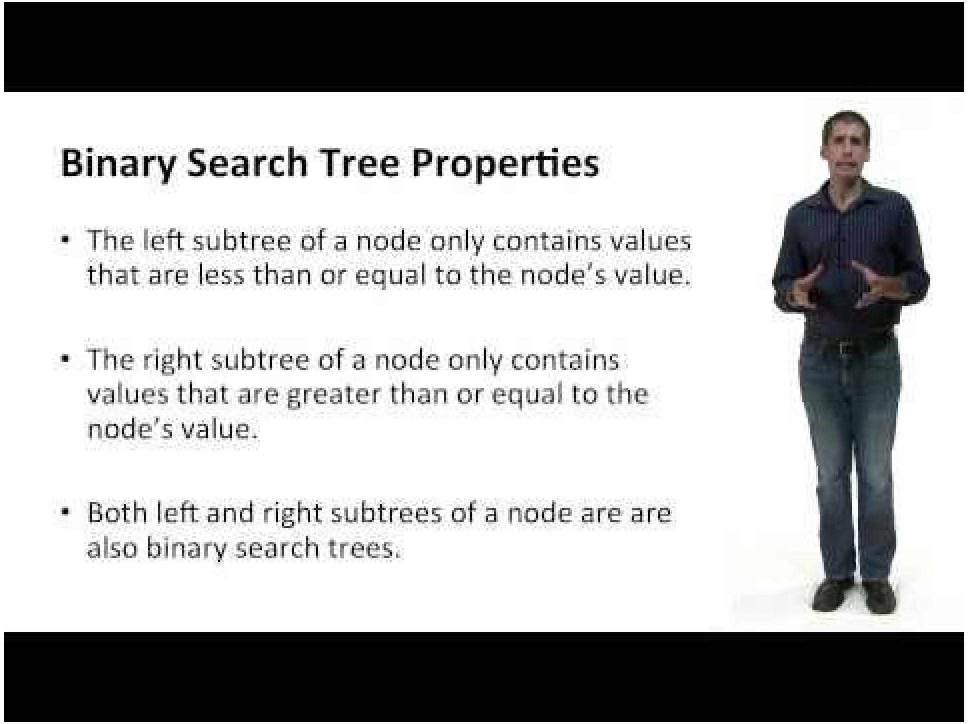
Pro: Fast! O(log n)

Con: harder to code

Cool Comparison Website

https://www.toptal.com/developers/sorting-algorithms

Sequential vs. Binary Search



Homework

Sort/Search Worksheets #3,4

TRY the quiz - not graded

6 ADVANCED: RECURSION

7 MERGESORT

8 UNIT QUIZ: ALGORITHMS