Data Structures & Algorithms: Sorting

Outline

What's interesting about DS&Algs? (3 min)

Our Running Example: Sorting (3 min)

Activities for younger students, and takeaways (10 min)

Run-through acticity for high school (and you!) (25 min)

Questions

What are Data Structures and Algorithms?

Data Structures and Algorithms

Data Structure: A way of organizing data so that it is easier to **use** and **maintain**.

Algorithm: A step-by-step process for completing a task.

Data structures can lead to simpler or faster algorithms.

Theme for today

Sorting

Sorting your data makes it easier to work with.

A fun playground for thinking about “how can I accomplish this task?” Lots of different algorithms for the same goal.

Sample Activities

Activity 1: Binary Search (Elem. and older)

Is 7 in the list or in your pocket?

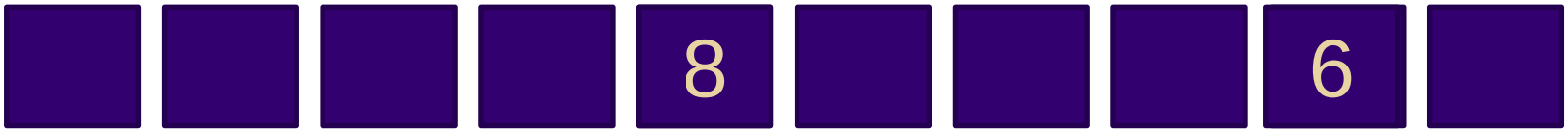
Place most of the cards face-down, rest in your back-pocket. How many do you flip to answer the question?

Now promise face-down cards are sorted. How does answer change?

Takeaway: If things are in order, check the middle.

Practice with “which is bigger?”; Useful for your students in real life.

Where is 7?



Activity 1: Binary Search (Elem. and older)

Is 7 in the list or in your pocket?

Place most of the cards face-down, rest in your back-pocket. How many do you flip to answer the question?

Now promise face-down cards are sorted. How does answer change?

Takeaway: If things are in order, check the middle.

Practice with “which is bigger?”; Useful for your students in real life.

Activity 2: Sort the numbers (middle school and older)

Execute insertion sort, selection sort, merge sort, quicksort.

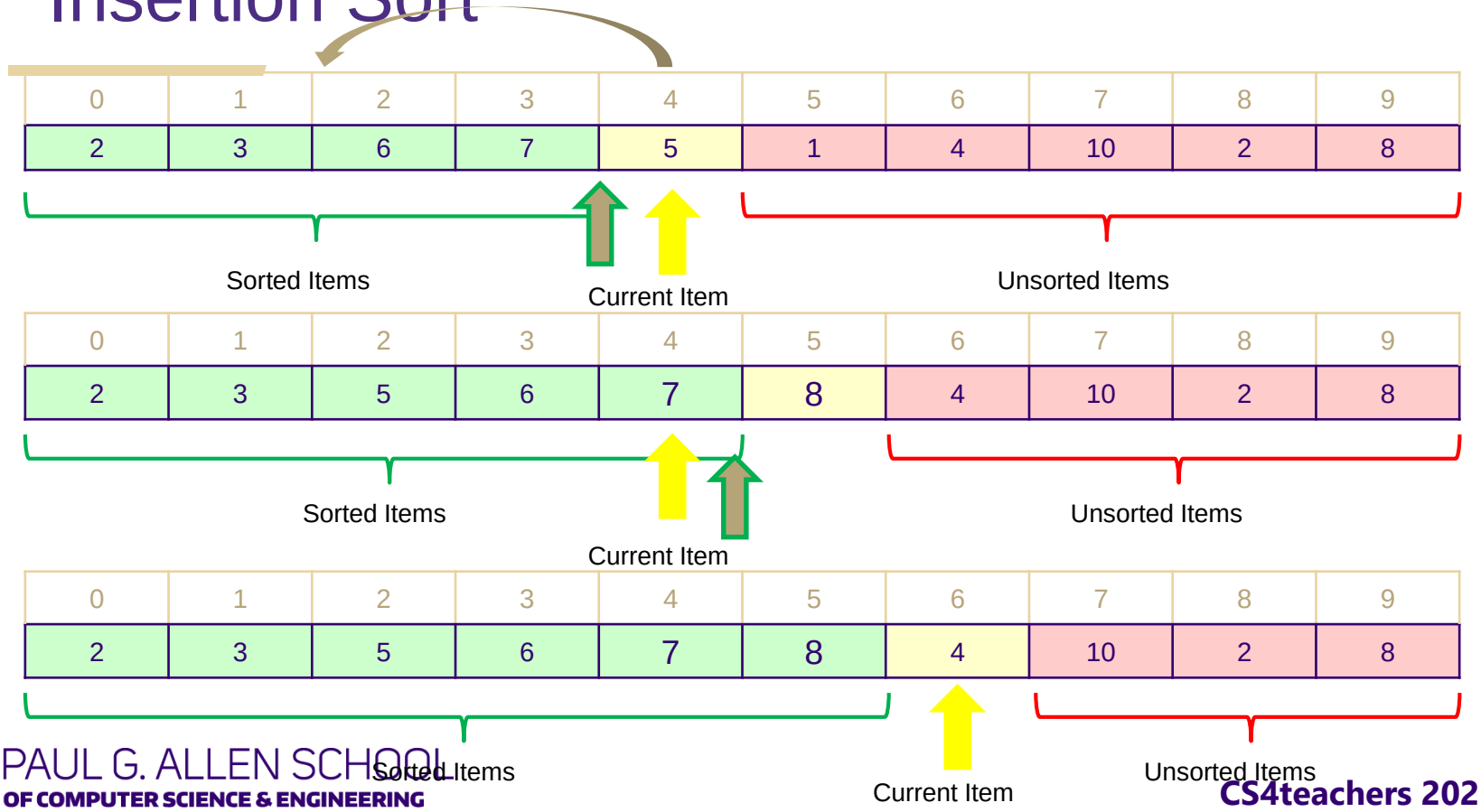
Move things like a computer: one special temporary location, one card in your hand. Everything else in the array.

Takeaway: There is more than one way to accomplish the same task! Also try "peasant multiplication"

Comparison of speed depending on data.

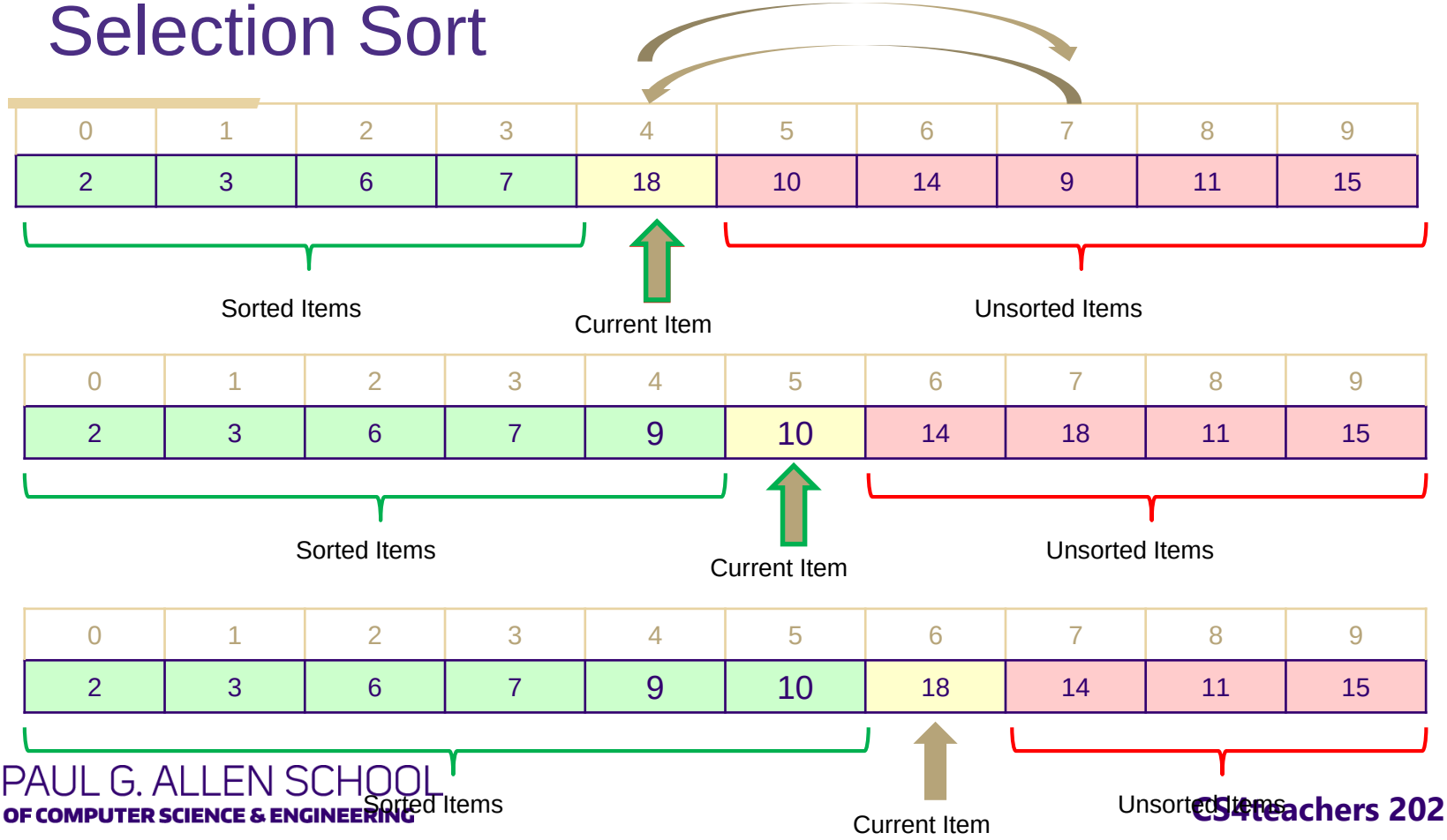
https://www.youtube.com/watch?v=ROalU379l3U

Insertion Sort



https://www.youtube.com/watch?v=Ns4TPTC8whw

Selection Sort



Activity 2: Sort the numbers (middle school and older)

Execute insertion sort, selection sort, merge sort, quicksort.

Move things like a computer: two special temporary locations, one card in your hand. Everything else in the list.

Takeaway: There is more than one way to accomplish the same task! Also try "peasant multiplication"

Comparison of speed depending on data.

Activity 3: Heaps

Given a scenario, what kind of data structure should we use?

What do we need to be able to do? And how fast can we do those things?

Takeaways: Motivation for/review of algebra II concepts; compare between two very different implementation ideas – which data structure works better?

Activity 3: Heaps

Scenario: Write the program to manage a TikTok feed.

Specifically which video is next?

Assumption: each video comes with a numerical score of how much the user will like it.

When the current video ends, select the best not-yet-shown one and remove. (`removeMax`)

When a new TikTok is posted, add it to the list. (`insert`)

Binary Heaps

A Binary Heap is

1. A Binary Tree
2. Every node is greater than or equal to all its children
   - In particular, the largest element must be the root!
3. The tree is complete
   - Every level of the tree is completely filled, except possibly the last row, which is filled from left to right.

What's a binary tree?

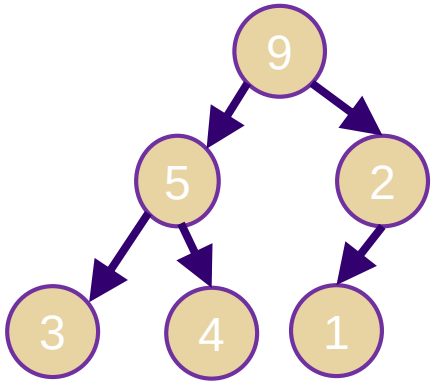
Every data point is a **node** (circle), and can have a "left child" and/or a "right child"

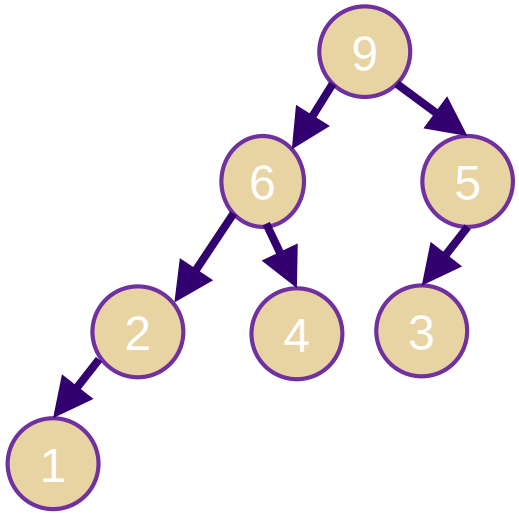
Binary Heaps

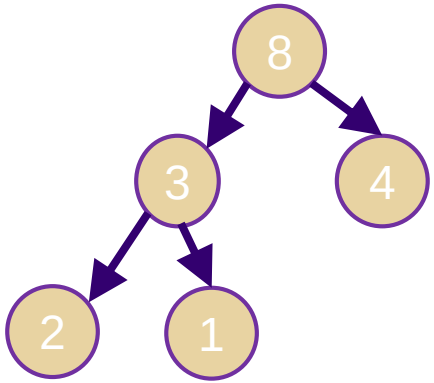
A Binary Heap is

1. A Binary Tree
2. Every node is greater than or equal to all its children
   - In particular, the largest element must be the root!
3. The tree is complete
   - Every level of the tree is completely filled, except possibly the last row, which is filled from left to right.

Heap? Yes or no.

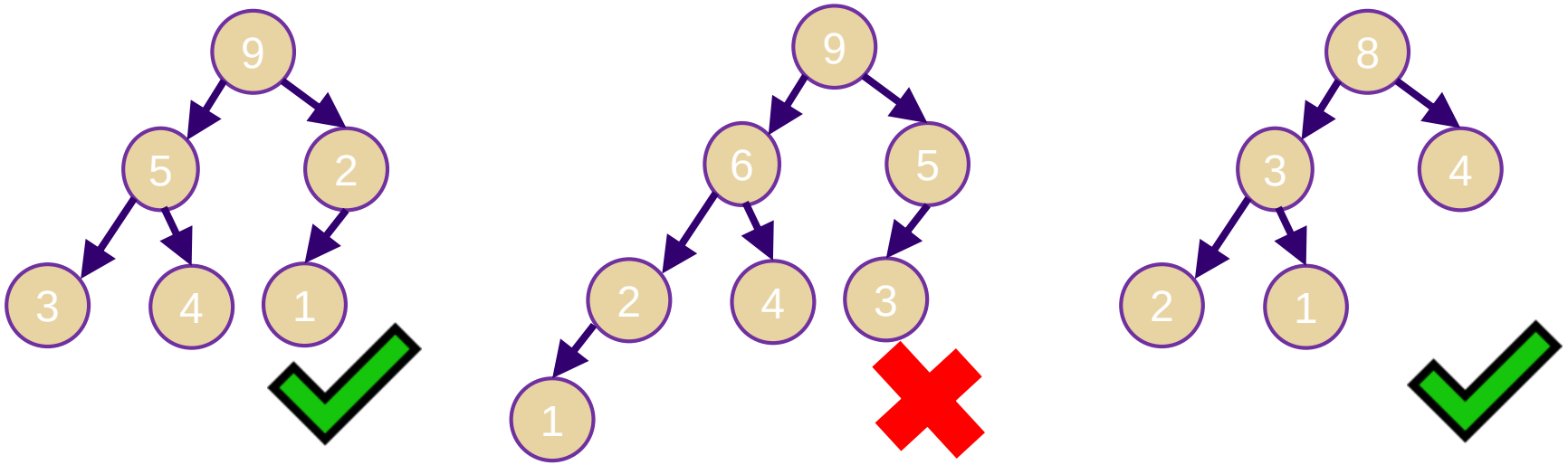






Takeaway: Practice applying definitions. Key ability in CS.

Heap? Yes or no.



Takeaway: Practice applying definitions. Key ability in CS.

Activity 3: Heaps

Scenario: Write the program to manage your TikTok feed.

When the current video ends, select the best not-yet-shown one and remove. (`removeMax`)

When a new TikTok is posted, add it to the list. (`insert`)

Idea: Binary Heap

Implementing Heaps

Let's start with `removeMax`.

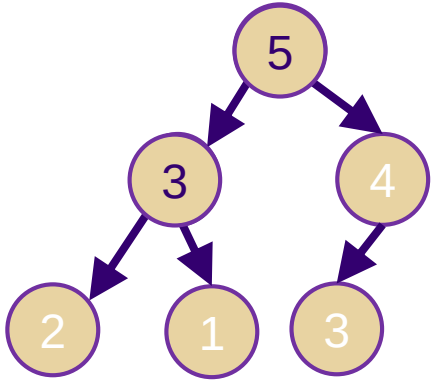
Where's the max?

Take it out…but now we don't have a heap!

Idea: take the bottom right-most node and use it to plug the hole

Shape is correct now

But that value might be to big. We need to "percolate it down"



`percolateDown` details
Compare to your two children,
If not largest, swap with your larger child //it's the biggest of you three.

Iterate until at the bottom or larger than children

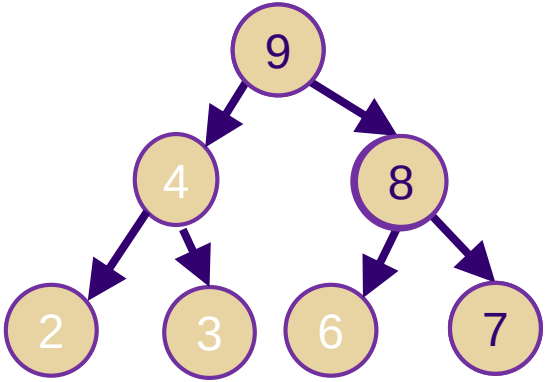
Implementing Heaps

How do you want to `insert`?

Ask the same questions: where does the new data point go to have the right shape?

How do we restore "big things are at the top"?

Insert



`percolateUp` details
Compare to your parent,
If larger, swap with your parent

Iterate until you're the root, or your parent is larger than you.

What is the shape going to be after the insertion?

Again, plug the hole first.

Might violate the heap property. Percolate it up

Hang on…

Can computers actually make swaps like this? Quickly?

Yes! Ask Robbie at Office Hours this afternoon if you've never seen it.

In fact, to estimate the running time it's enough to count the number of swaps we'd need.

Let's see if our algorithms are efficient.

How long does it take?

`RemoveMax`?

Need to: Remove top element, swap bottom element to top, push top element down where it belongs.

`Insert`?

Need to: Place new element at bottom, push element up where it belongs.

Need to know: **How tall will the heap be?**

How tall will the heap be?

Suppose we have n videos in our data structure. How tall is the tree?

Ask the opposite question, if our tree has h full levels, how many nodes does it have?

$$\sum_{i=1}^{h} 2^{i-1} = 2^h - 1$$

So the number of swaps is…

About $\log_2(\mathrm{n})$

Summations and $\log$ s are both common tools in analysis of data structures – motivation when you introduce them in Algebra 2.

$\log_a b$ answers the question a to what power gives b? So if 2^h gives (about) n, then $\log_2(n)$ gives the exponent, i.e. the height.

Punchline

Compare this implementation to the first one your students would think of.

Probably a fully-sorted array. (another option: unsorted array)

`insert` will be $O(n)$ (shifting elements to make room)

`removeMax` will be $O(1)$

Which is better? Probably a heap, but one could argue otherwise.

Comparison

	`removeMax`	`insert`
Binary Heap	$O(\log n)$	$O(\log n)$
Sorted Array	$O(1)$	$O(n)$
Unsorted Array	$O(n)$	$O(1)$

Extensions

`DecreaseKey` given an element, change its priority to a smaller number. How long does that take with a heap? With a

Motivation: don't want to show old videos unless our users will really like them! Decrease the values over time.

Extensions

`Build` given a list of n videos, turn them into the data structure

Motivation: When someone first makes an account, you have to start somewhere!

Potential Learning Objectives

- Understand that sorting makes finding things easier.
- Use binary search in real life.
- There's a difference between a **problem** and an **algorithm** to solve it. There might be many algorithms for the same problem!
- Design a new data structure together.
- **Compare** pros and cons of a few different possible implementations.