ANNOUNCEMENTS

Today is the last week of class
Today is Donut Monday (grab a donut!)
Wednesday is Fun Final Review (kahoot? who can say...)

WARMUP

what is object-oriented programming?
are you an object-oriented programmer?
can you jam with the console cowboys in cyberspace?

TODAY

OOP, encapsulation & inheritance

---

do you know anything about hackers?

---

record LEC-02

OOP

---

indiana, let it go

object
(instance of a **class**)

anatomy of a class (1/2)

```
class ClassName {
    VariableOneType variableOne;
    ...

    FunctionOneReturnType functionOneName(...) { ... }
    ...
}
```

- a **class** is (a blueprint for) a lil chunk of data that you can make elsewhere
  - a class may have any number of **variables** (fields)
    - `int foo; // objects of this class have an int called foo`
  - a class may have any number of **functions** (methods)
    - `int bar() { ... } // objects of class have function bar`

anatomy of a class (2/2)

```
class Vector2 {
    // instance variables
    double x;
    double y;

    // constructor
    Vector2(double x, double y) { ... }

    // instance methods
    double length() { ... }
    ...
}
```

an **object** is **an instance of a class**

```
// v is a reference to an instance of the Vector2 class
// v is an instance of the Vector2 class
// v is a Vector2 object
// "v is a Vector2"
Vector2 v = new Vector2();
```

object-oriented programming (OOP)

note: jim maybe has opinions (who's to say)

object-oriented programming (OOP)

- **object-oriented programming** means *thinking* in terms of nouns
  - "how can i break down this problem into classes/objects?"

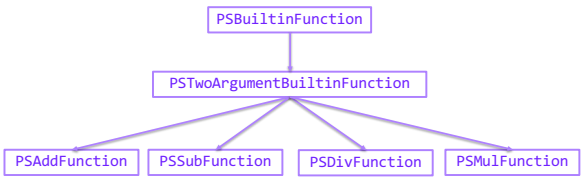
object-oriented programming (OOP)

- **object-oriented programming** is NOT just *having* classes/objects
  - recall, a **class** is just (a blueprint for) a lil chunk of data
  - rather, OOP means my problem-solving is *oriented* around objects
    - ...instead of, for example, data (**data-oriented design**)
    - ...functions (**functional programming**)

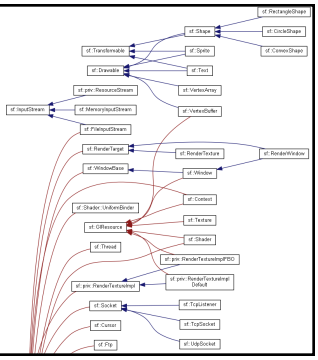
object-oriented programming

- **example:** to implement an Object-Oriented PostScript interpreter...
  - `class PSInterpreter`
  - `class PSProgram`
  - `class PSStack`
  - `class PSMap`
  - `class PSBuiltinFunction`
  - `class PSTwoArgumentBuiltinFunction extends PSBuiltinFunction`
  - `class PSAddFunction extends PSTwoArgumentBuiltinFunction`
  - `class PSSubFunction extends PSTwoArgumentBuiltinFunction`
  - `class PSMulFunction extends PSTwoArgumentBuiltinFunction`
  - `class PSDivFunction extends PSTwoArgumentBuiltinFunction`
  - `...`

unified modeling language (UML) diagram



real-world example:
SMFL



note that we still haven't written any actual code

we've made a *plan* for how to break the problem into objects

note: it can be hard to break problems into objects

Consider a very basic question: should a Message send itself?
'Sending' is a key thing I wish to do with Messages,
so surely Message objects should have a 'send' method, right?
If Messages don't send themselves, then some other object
will have todo the sending, like perhaps some not-yet-created Sender object.
Or wait, every sent Message needs a Recipient, so maybe instead Recipient
objects should have a 'receive' method.
This is the conundrum at the heart of object decomposition.
Every behavior can be re-contextualized by swapping around
the subject, verb, and objects.
Senders can send messages to Recipients;
Messages can send themselves to Recipients;
and Recipients can receive messages.
--Brian Will

so

it is very hard to break a problem into objects

but

it is also very *popular* to break a problem into objects

so let's learn some OOP 🙂 👍

inheritance

one class can **inherit** from another

- a **child class** (**derived class**, **subclass**) inherits from its **parent class** (**base class**, **superclass**)
  - a child **inherits** (gets, has) its parents' variables and functions

```
// Inheritance: HW13 is an App
class HW13 extends App {
    // HW13 overrides loop()
    void loop() {
        if (keyPressed('a')) {
            ...
        }
    }
}
```

```
class App {
    Vector2 mousePosition;
    boolean keyPressed(...);
    void setup() { ... }
    void loop() { ... }
    ...
}
```

inheritance is convenient, but NOT fundamental

(except sometimes in Java)

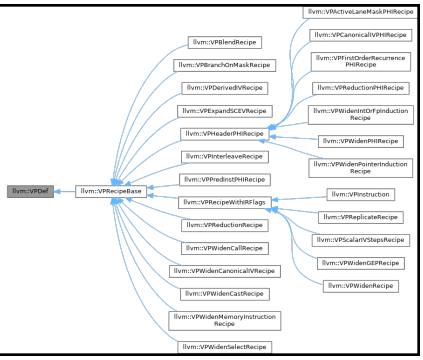
- instead of extending a class, we can store a reference to an instance of it
  - this is called "**composition**"
  - we will have to use the dot operator (a lot) more, but c'est la vie

```
// Inheritance: HW13 is an App
class HW13 extends App {
    // HW13 overrides loop()
    void loop() {
        if (keyPressed('a')) {
            ...
        }
    }
}
```

```
// Composition: HW13 has an App
class HW13 {
    App app;
    void loop() {
        if (app.keyPressed('a')) {
            ...
        }
    }
}
```

the fundamental point is that maybe it's nice to reuse code

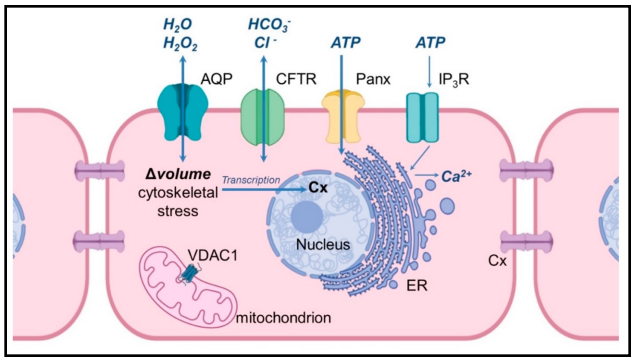
maybe



final note: inheritance doesn't simplify a problem so much as i t s p r e a d s i t o u t

(and spreads it *around*?)

encapsulation



encapsulation

- **encapsulation** is the idea that a class should be like a "capsule"
  - the variables inside the capsule should be `private`
    - users of the class CANNOT touch them
  - for its users, the class should expose safe, `public` functions

```
void put(KeyType key, ValueType value) { ... }
ValueType get(KeyType key) { ... }
int size() { return this._size; }
```

```
ArrayList<KeyValuePair>[] buckets;
int _size;
```

encapsulated hashmap

note: this probably makes sense

the typical user of a hashmap shouldn't be messing with the `private` array

(and perhaps the exceptional user should write their own hashmap)

big idea: encapsulation can hide away messy, dangerous details

note: this is just a metaphor;
do NOT play with fire

however

encapsulation can maybe be taken too far

```
bullet.age++;
```

```
bullet.setAge(bullet.getAge() + 1);
```

```
bullet.ageUp(); // ?
```

OOP considered maybe mildly frictious to prototyping

final note: encapsulation doesn't *add* functionality
encapsulation *removes* functionality

A N N O U N C E M E N T S
check HW13 page for more resources (in ~4 hrs)
today is Fun Kahoot Final Review
friday is the last day of class

W A R M U P
what is a computer program?
what is programming?
what is a data structure?

T O D A Y
review

ANNOUNCEMENTS
check HW13 page for more resources (in ~4 hrs)
today is Fun Kahoot Final Review
friday is the last day of class

WARMUP
what is a computer program?
what is programming?
what is a data structure?

TODAY
review

what has this course been about?

data structures & big O runtime

as i review...
use the big O runtimes to remind yourself how the data structures work
😊👍

data structures

what is the most important data structure?

quadtrees

just kidding

it's arraaaaaaays.

(also trees, but mostly arrays.)

an **array** is a sequence of contiguous elements

- the length of an array is fixed
- ⏱ creating an array takes O(n) time, where n is the length of the array
- because the elements are all the same size and all right next to each other, **accessing** an array is FAST
  - ⏱ getting the value of the i-th element of an array takes O(1) time
  - ⏱ setting the value of the i-th element of an array takes O(1) time

a **string** is internally an array of characters

- ⏱ adding (concatenating) two strings of length n is an O(n) operation
  - we have to make a new string of length 2n, which is O(n)

we used an array to implement an **array list**, which automatically resizes itself

- ⏱ when the internal private array is not full, adding is O(1)
- ⏱ when the internal private array is full, adding is O(n)
  - have to make a new array (of, for example, twice the length) and copy

we used an array list to implement a **stack** and a **queue**

- stacks and queues restrict the way we access a sequence of elements
  - a **stack** can **push** an element to the top & **pop** the top element
    - 🤷
  - a **queue** can **add** an element to the back & **remove** the front element
    - 🚶🚶🚶🚶🚶🚶🚶

we can use an array of array lists to implement a **hash map**

- ⏱ putting (adding) a key-value pair into a hash map is O(1)
  - it's a "slow O(1)," which involves **hashing** the key
    - in **separate chaining** (like we did in class), this takes us to a **bucket**
      - if the key was already in the bucket, we overwrite its value
      - otherwise we add the key-value pair to the bucket
- ⏱ get key's value from a hash map is O(1)

arrays.

(also **trees**, but mostly arrays.)

a **node** is a little teeny class

- a node can store **data** (like a **value**) as well as **references** to other nodes

a chain of nodes is a **linked list**

- a linked list has the same **interface** as an array list, but acts very differently
- ⏱ adding to the end of a singly linked list is O(n)
- ⏱ getting an element by index is O(n) 😟

a branching chain of nodes (with no cycles) is a **tree**

- each node has references to its **children**
- a node's children & its children's children & its children's children's children, ... are called its **descendants**
- the node with no parents is called the **root**

a tree where each node can only have up to two children is a **binary tree**

- a node in a binary tree can have a **left child** and a **right child**
- the height of a **balanced** binary tree is O(log n)

binary search trees and heaps are special kinds of binary trees

- in a **binary search tree**, each node's value is greater than the values of all of its left descendants and less than the values of all of its right descendants
  - ⏱ **searching** a **balanced** binary search tree is O(log n)

- in a **max binary heap**, each node's value is greater than the values of both of its children
  - ⏱ **adding** an element to a max binary heap is O(log n)
    - 🍁 add to the first empty slot and **swim up**
  - ⏱ **removing** the max element (root) from a max binary heap is O(log n)
    - ⚓ swap the **last element** with the root and **sink down**

what does it all mean?

here's my attempt to summarize the course in a few sentences

136

- **data** means numbers (letters are numbers, colors are numbers, ...)
- there's no one perfect way to organize your data
  - it depends on the problem
- to find a good data structure for a particular problem, we can...
  - ...just try an array first; if that works, great, we're done 😀 👍
  - ...compare different data structures using math (**big O** notation)
    - ...in terms of speed
    - ...in terms of space
  - ...compare different data structure's **usage code**
    - ...an array might be fast, but is it *pleasant*?

136 was us moving from finding *a* solution...

...to finding a good solution

and starting to build the tools and confidence to solve bigger problems

like the final project 😀 👍

what does it all mean?

what does it all meme?

Tutorial 00

Jim edited this page on Oct 19 · 148 revisions

drjava is the best IDE ever. It comes with a working `Debug Mode`, easy to click `Compile` and `Run` buttons, an `Interactions` pane for easy experimentation, and the simplest automatic indentation in the business (just highlight your code and press `Tab`). Meanwhile, VS Code is clocking in with...Dark Mode and the ability to read your mind? Pffff... 😊 Don't believe the hype. DrJava is where it's at. 🩺☕🔥

increment operator

- to "**increment**" means to increase the value of a number by one
  - `i = i + 1;`
  - `i += 1;`

  - the **pre-increment** `++i` increments `i` and returns the new value of `i`
  - `j = ++i; // i = i + 1;`
  - `// j = i;`

  - the **post-increment** `i++` increments `i` and returns the old value of `i`
  - `j = i++; // j = i;`
  - `// i = i + 1;`

decrement operator

- to "**decrement**" means to decrease the value of a number by one
  - `i = i - 1;`
  - `i -= 1;`

  - the **pre-decrement** `--i` decrements `i` and returns the new value of `i`
  - `j = --i; // i = i - 1;`
  - `// j = i;`

  - the **post-decrement** `i--` decrements `i` and returns the old value of `i`
  - `j = i--; // j = i;`
  - `// i = i - 1;`

story time ☕

aws

Get Started for Free | Contact Us

What are the key principles in SSL/TLS certificate technology?

SSL/TLS stands for secure sockets layer and transport layer security. It is a protocol or communication rule that allows computer systems to talk to each other on the internet safely. SSL/TLS certificates allow web browsers to identify and establish encrypted network connections to web sites using the SSL/TLS protocol.

Encryption

Encryption means scrambling the original message so that it can only be decrypted by the intended recipient. For example, you change the word *cat* to *ecv* by moving every letter forward in the alphabet by two places. The recipient knows the rule (or key) and reverses each letter by two places to read the actual word. SSL/TLS encryption builds on this concept by using public key cryptography, with two different keys to encrypt and decrypt a message. PKI

About the security content of iOS 7.0.6

This document describes the security content of iOS 7.0.6.

For the protection of our customers, Apple does not disclose, discuss, or confirm security issues until a full investigation has occurred and any necessary patches or releases are available. To learn more about Apple Product Security, see the Apple Product Security website.

For information about the Apple Product Security PGP Key, see "How to use the Apple Product Security PGP Key."

Where possible, CVE IDs are used to reference the vulnerabilities for further information.

To learn about other Security Updates, see "Apple Security Updates".

iOS 7.0.6

- **Data Security**

  Available for: iPhone 4 and later, iPod touch (5th generation), iPad 2 and later

  Impact: An attacker with a privileged network position may capture or modify data in sessions protected by SSL/TLS

  Description: Secure Transport failed to validate the authenticity of the connection. This issue was addressed by restoring missing validation steps.

  CVE-ID

  CVE-2014-1266

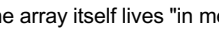
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
        OSStatus        err;
        ...

        if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                goto fail;
        if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                goto fail;
                goto fail;
        if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                goto fail;
        ...

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
        OSStatus        err;
        ...

        if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                goto fail;
        if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0) {
                goto fail; }
        goto fail;
        if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                goto fail;
        ...

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

Caaaaaaaaaaarl

- **e.g.,** Imagine a classroom with n students. I want to figure out if any students are named Carl.
  - I need an ✨Algorithm✨ -- `boolean isAnyoneNamedCarl(Student[] students);`
  - What is the big O of the following algorithms?
    - **Algorithm 1:** Ask each student, one at a time, "Are you named Carl?"
    - **Algorithm 2:** Pass a paper around the room, and have each student write their name on it. Then take the paper, and read through it.
    - **Algorithm 3:** The students draw straws. The student who draws the short straw must leave. On their way out of the room, ask them whether their name is Carl. Repeat this procedure until the room is empty.
    - **Algorithm 4:** Play Kahoot. The winner legally changes their name to Carl.

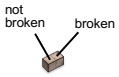
mental model of an array

- elements live in equally-sized boxes all right next to each other
  - 
  - the array itself lives "in memory"

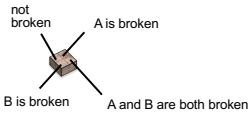
💀 **example:** bad very bad broken array copy

💀 **example:** bad very bad broken array copy

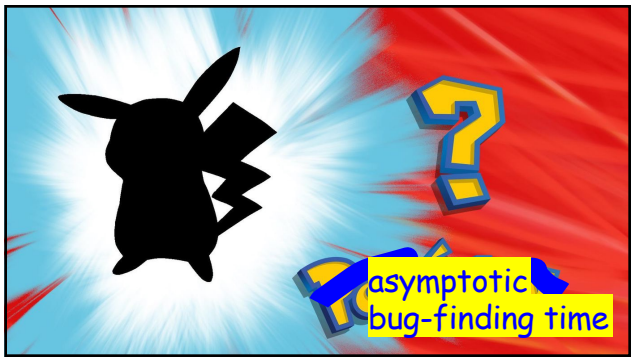
```
import java.util.*;

class Main {
    public static void main(String[] arguments) {
        int[] source = { 3, 4, 5 };

        int[] destination = source;

        source[0] = 7;
        System.out.println(Arrays.toString(source));
        System.out.println(Arrays.toString(destination));
    }
}
```

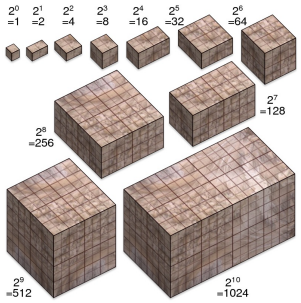


developing one thing at a time



developing two things at a time



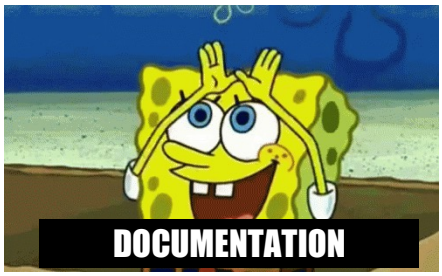


$\mathcal{O}(2^n)$

gl;hf 😊 👍

code up the entire program without compiling it

compile early; compile often



Tungsten break 😀

did you know the density of Tungsten is 19.25 g/cm^3?
Gold is 19.32 g/cm^3

did you know Tungsten is really hard?
Tungsten alloys can be as hard as Sapphire! that's really hard!

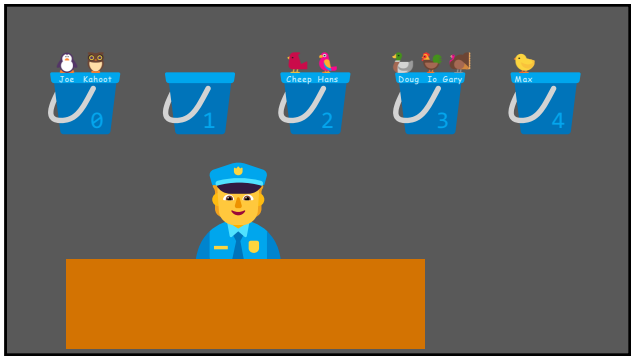
Tungsten is expensive (that cube is $60) but not like...that expensive

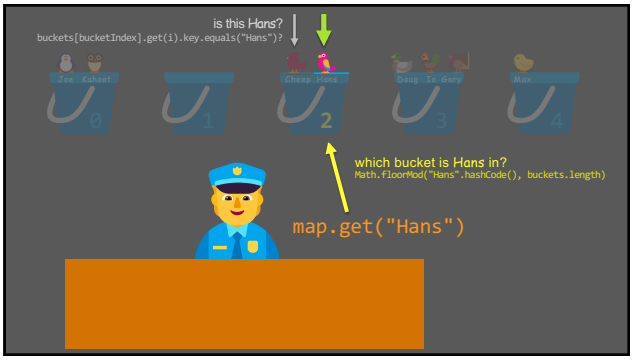
what could YOU do with Tungsten?

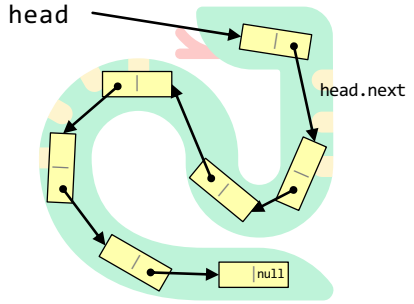
```
if (token.type == 2) {
    ...
}
```

```
if (token.type == Token.TYPE_STRING) {
    ...
}
```

previously, on *Hans the Parrot*







"the stack"

local variable primitives & **references** to Objects

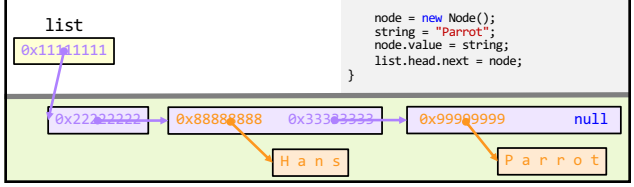
undefined *by default (will NOT compile if used)*

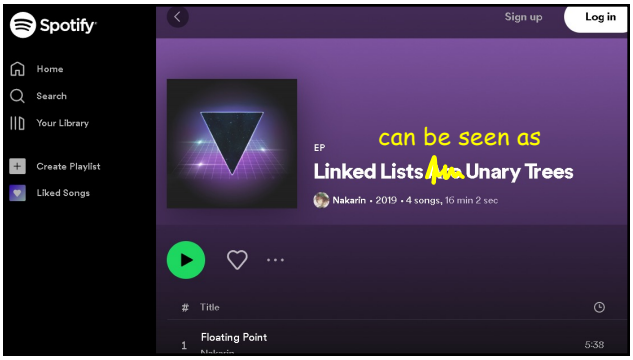
"the heap"

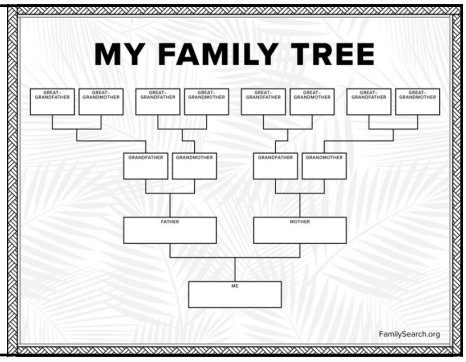
the actual **Objects** (arrays and String's count as Objects)

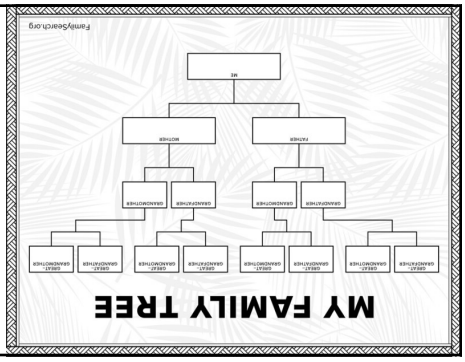
cleared to 0 or null by default

```
LinkedList list;
{
    Node node;
    String string;

    list = new LinkedList();

    node = new Node();
    string = "Hans";
    node.value = string;
    list.head = node;

    node = new Node();
    string = "Parrot";
    node.value = string;
    list.head.next = node;
}
```



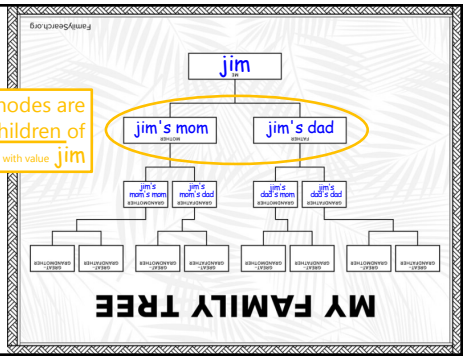




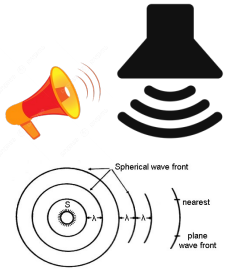


careful: these nodes are the children of jim

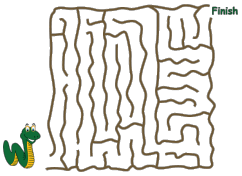
the node with value jim



breadth-first vs. (pre-order) **depth-first**

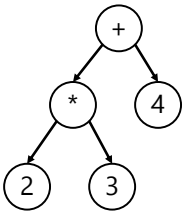


"a persistent, optimistic, very thorough worm" –Mark Hopkins



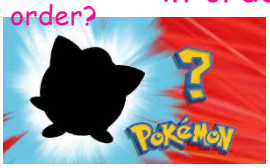
quadtree rabbit hole

2 * 3 + 4

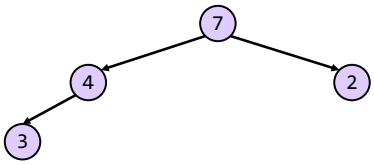


who's that traversal order?

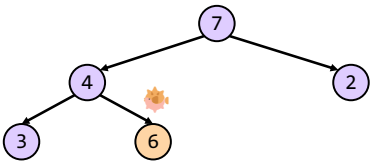
in-order!



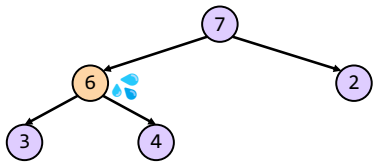
[7 4 2 3 6 1 8 5 9]



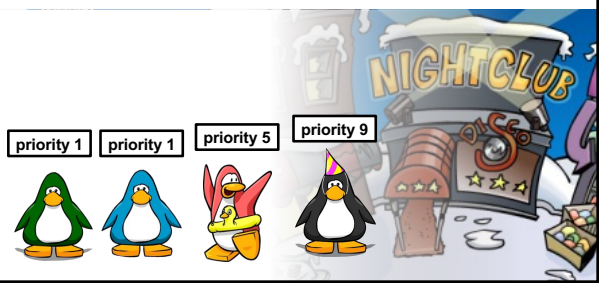
[7 4 2 3 6 1 8 5 9]



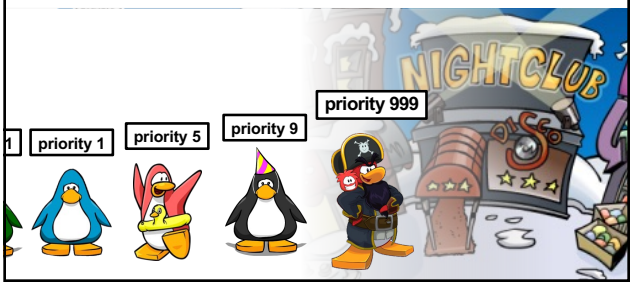
[7 6 2 3 4 1 8 5 9]



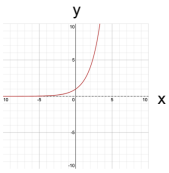
extension: priority queue



extension: priority queue

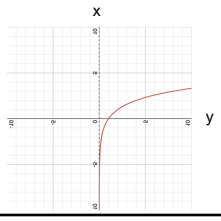


$y = 2^x$



$x = \log_2 y$

$y = 2^x$



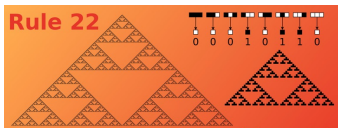
$x = \log_2 y$

😊 👍

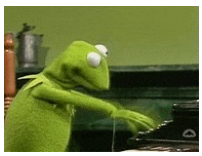
😳 👍

homework

✨ **how do the post-midterm homeworks *work?***
can you "run them by hand?"

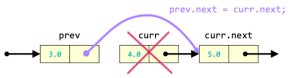


how do you interpret PostScript?

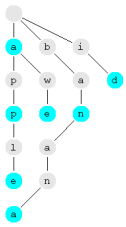


how do you generate text?

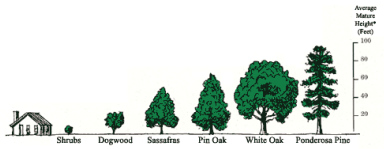
remove (**delete**) a node

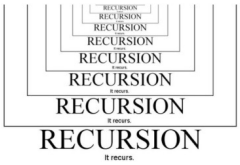


how do you create & change a linked list?



how do you store a vocabulary in a tree (trie)?





what is the big O runtime of a piece of code?

kahoot

questions