

ANNOUNCEMENTS



how is the linked list hw?

today is "We Ran Out of Time For Kahoot Friday" Monday!

WARMUP

- here is a **tree**:



what are some uses of trees?

- Kahooooot!

TODAY for-each loops, directed graphs & trees



record LEC-02

~review

for-each loop



Python's for loop is a **for-each** loop

```
lst = [ True, 5.0, "Hello" ]  
for element in lst: # for each item in the list...  
    print(element)
```

a **for**-each loop can be a convenient alternative to a for loop

```
for (int turretIndex = 0; turretIndex < turrets.length; ++turretIndex) {  
    Turret turret = turrets[turretIndex];  
    ...  
}
```

```
for (Turret turret : turrets) { // for each turret in turrets...  
    ...  
}
```

```
for (Turret turret : turrets) { // for each turret in turrets...
    ...
}
```

for-each loop can be a convenient alternative to a for loop

```
for (int strokeIndex = 0; strokeIndex < currentFrame.size(); ++strokeIndex) {  
    ArrayList<Vector2> stroke = currentFrame.get(strokeIndex);  
    for (int pointIndex = 0; pointIndex < stroke.size(); ++pointIndex) {  
        Vector2 point = stroke.get(pointIndex);  
        ...  
    }  
}
```

```
for (ArrayList<Vector2> stroke : currentFrame) {  
    for (Vector2 point : stroke) {  
        ...  
    }  
}
```

```
for (ArrayList<Vector2> stroke : currentFrame) {
    for (Vector2 point : stroke) {
        ...
    }
}
```

for-each loop

- a **for-each loop** can be a convenient alternative to a for loop
- you never "need" a for-each loop
- can always "fall back to" a for loop with an index
- **for-each loops have the same dangers as "convenient references"**


```
for (int turretIndex = 0; turretIndex < turrets.length; ++turretIndex) {
    Turret turret = turrets[turretIndex]; // convenient reference
    ...
}
```

```
for (Turret turret : turrets) { // convenient reference
    ...
}
```

- ```
for (Turret turret : turrets) { // convenient reference
 ...
}
```

The diagram illustrates a linked list structure. It consists of three nodes and a null pointer. The first node contains the value '0.0, 0.0' and points to the second node. The second node contains the value '0' and points to the third node. The third node contains the value '0.0, 0.0' and points to a null value. The null value is represented by 'null' and 'nu'.

## for-each loop

- In Java, a for-each loop can iterate through `String`'s with `toCharArray()`
  -  `toCharArray()` is  $O(n)$ , but fine for CSCI 136

```
for (char c : string.toCharArray()) {
 ...
}
```

- ```
for (char c : string.toCharArray()) {  
    ...  
}
```

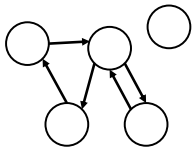
directed graphs & trees

TODO (Jim): Talk about math.

directed graph

a directed graph is a super general linked list

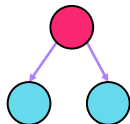
- a node in a **graph** can have references to any number of other nodes
 - nodes are drawn as circles
 - references are drawn as arrows



(directed) tree

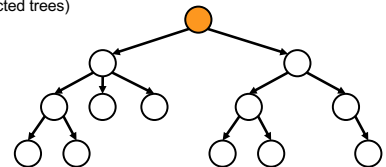
trees use kinship terms

- a node has references to other nodes
- it is their **parent**
- they are its **children**

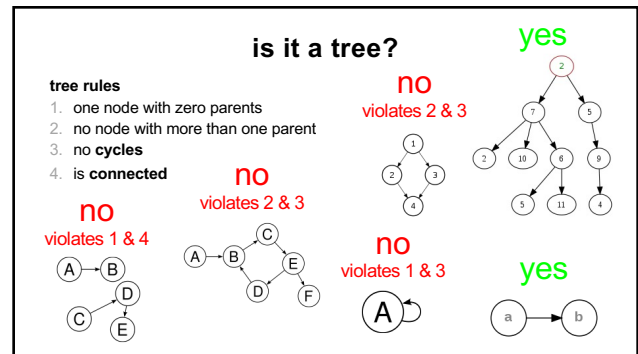


a tree is a graph that...

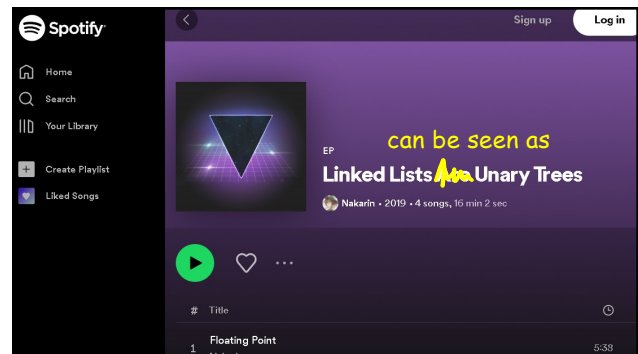
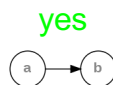
- has exactly one node with zero parents (the **root**)
- has no nodes with more than one parent
- has no **cycles** (loops)
- is **connected** (just one tree, not multiple disconnected trees)



time for everyone's favorite
home game...



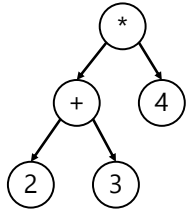
is it a tree?



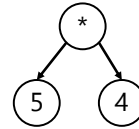
uses of trees

simplifying expressions

$(2 + 3) * 4$



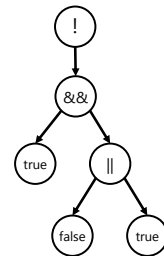
$5 * 4$



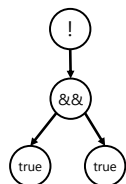
20

20

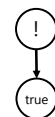
$!(true \ \&\& \ (false \ || \ true))$



$!(true \ \&\& \ true)$



$!true$

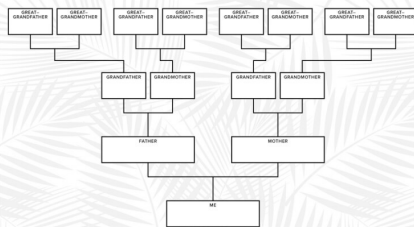


false



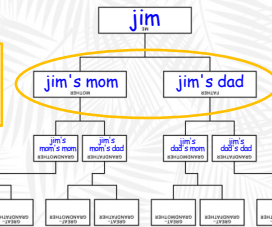
and so much more!

MY FAMILY TREE

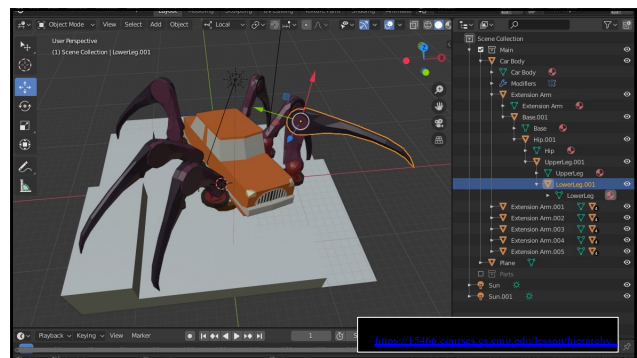
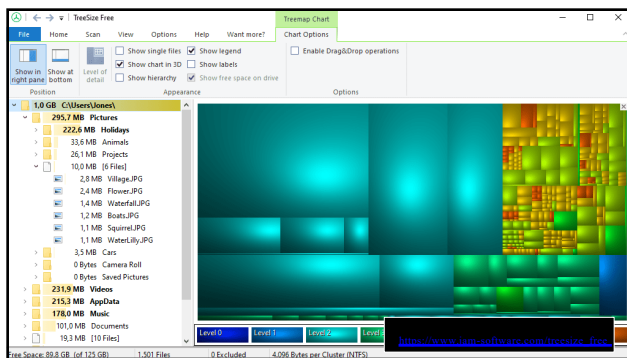


FamilySearch.org

careful: these nodes
are
the children of
the node with value **jim**



MY FAMILY TREE





ANNOUNCEMENTS
today is Laptop Wednesday

WARMUP
evaluate this expression tree

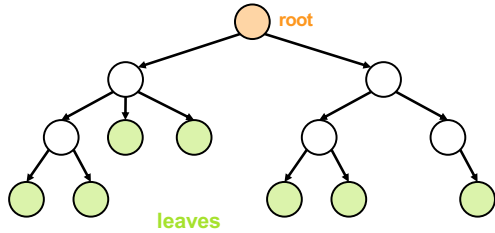
TODAY
tree terminology and a terrific tree traversal tutorial

record LEC-02

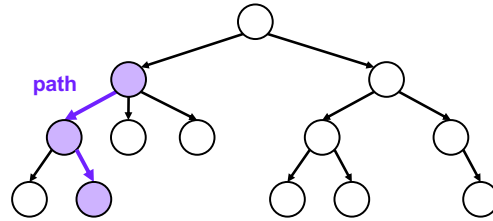
tree terminology



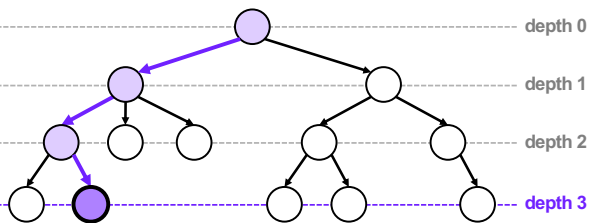
the **root** has no parents
a **leaf** has no children



a **path** is a chain of distinct edges



a node's **depth** is the number of edges
in the path from the root to that node



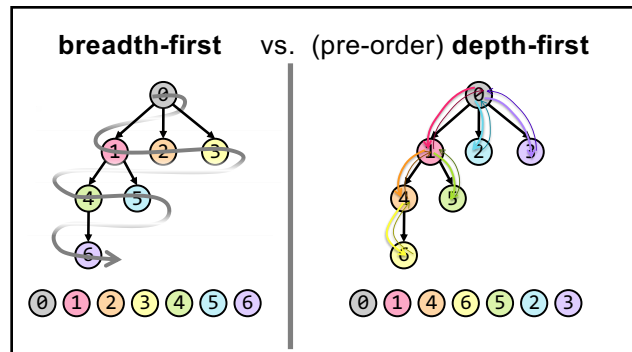
level means either *depth* or $(depth + 1)$
depending on who you ask



tree traversal

let us first experience
tree traversal visually

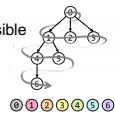
switch slide deck



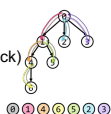
tree traversal (walking the tree)

- **tree traversal** means iterating through all of the tree's nodes

- **breadth-first** explores as broadly (widely) as possible
 - one **level (depth)** at a time
 - uses a **queue**



- **depth-first** explores as deeply as possible
 - one **root-to-leaf path** at a time
 - uses a **stack** (or **recursion**, which uses the callstack)



- ⚠ there are actually 6 subtly different variants of depth-first traversal (I will explain them when we get to binary trees)

why might we want to traverse a tree?

why did we want to traverse a linked list?

do we every "traverse" an array?

gathering all words in a **trie** ([HW09](#))

searching for something ([HW10?](#))

evaluating an **expression tree**

gathering all the nodes
into a list ([Tut09](#))

Terrific Tree Traversal Tutorial

ANNOUNCEMENTS

today is Fun Friday?

WARMUP

vibes seem low, so here is
an extra a-MAZE-ing warmup
draw this maze as a tree

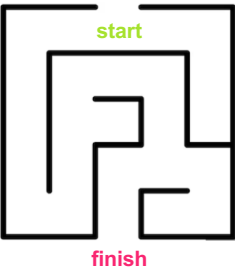
TODAY

tree traversal wrapup; quadtree reading

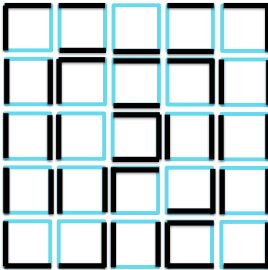
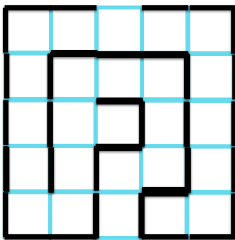
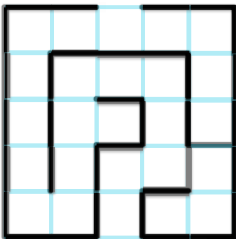


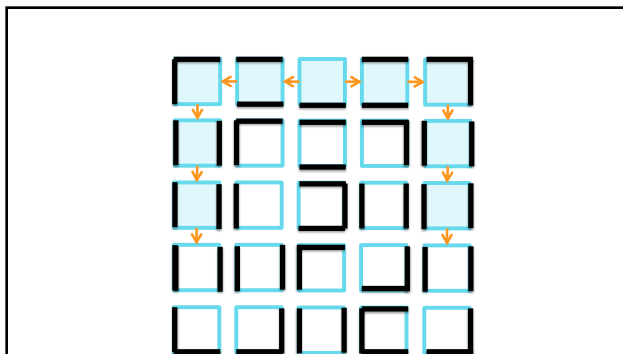
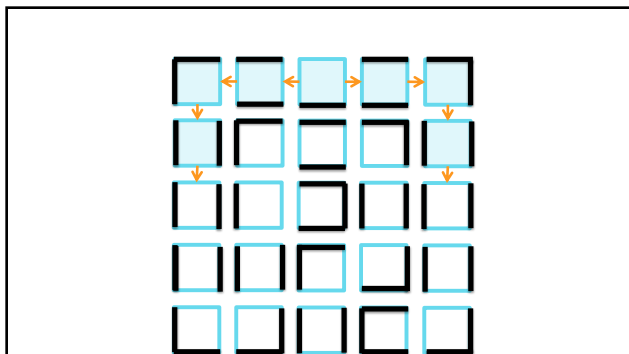
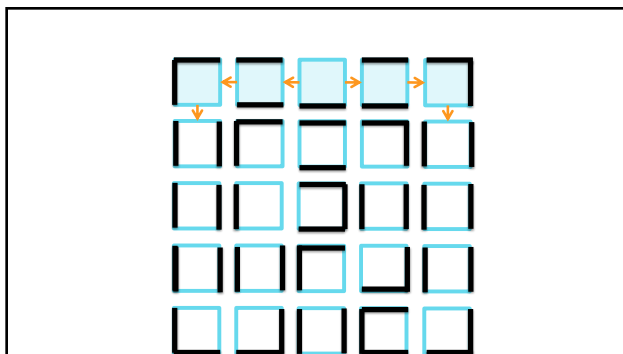
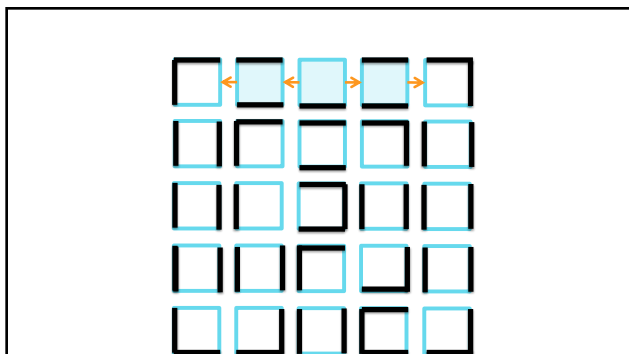
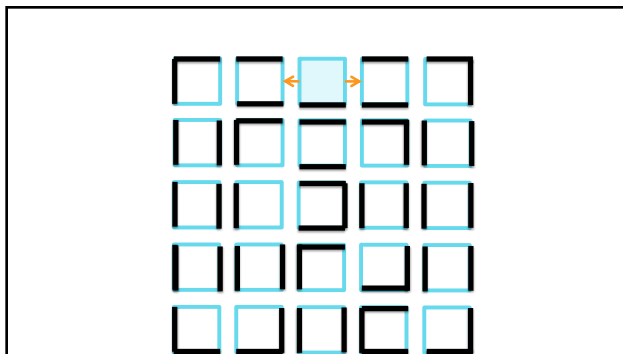
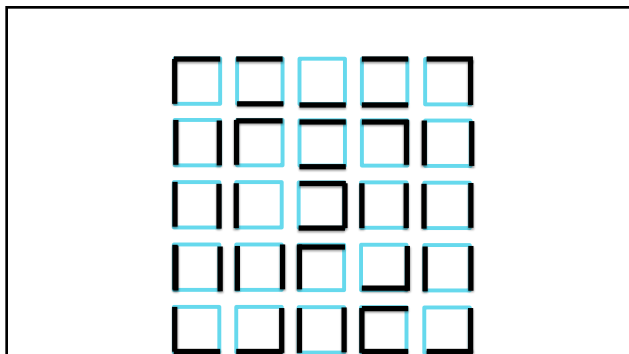
record LEC-02

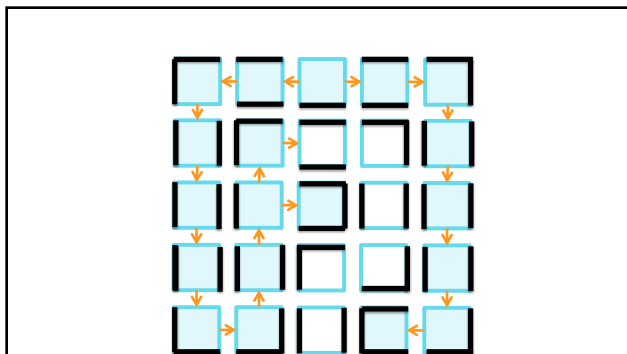
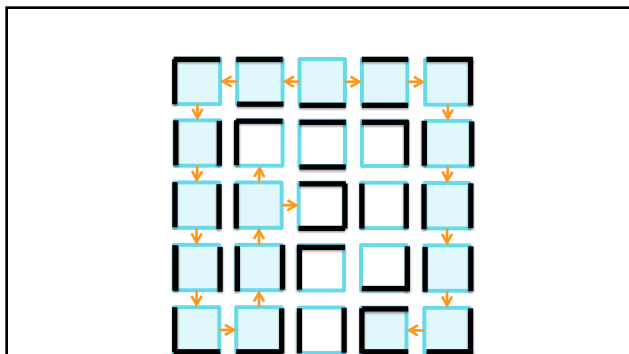
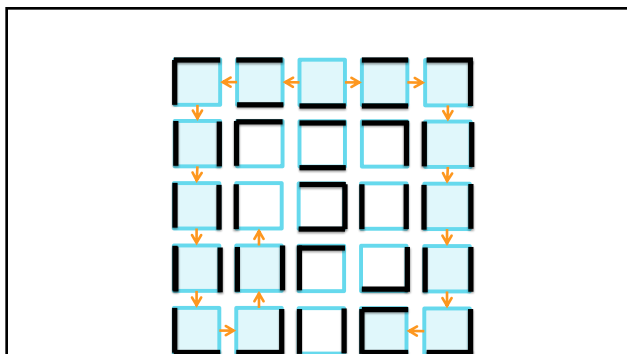
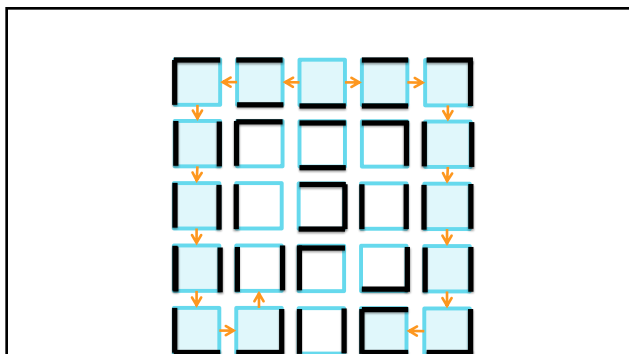
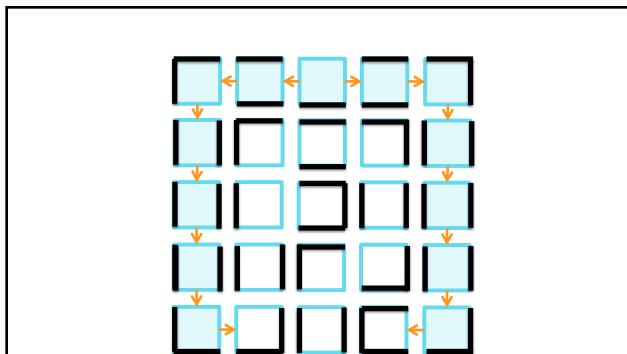
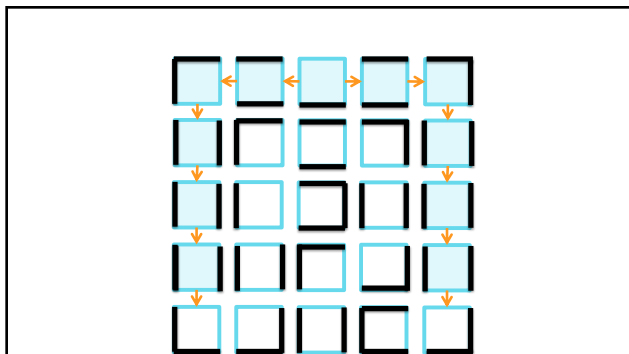
MAZE GAME

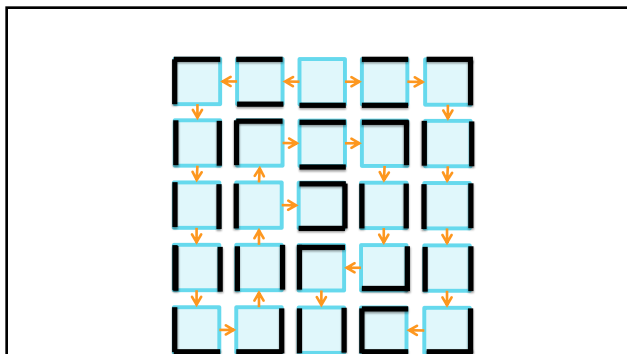
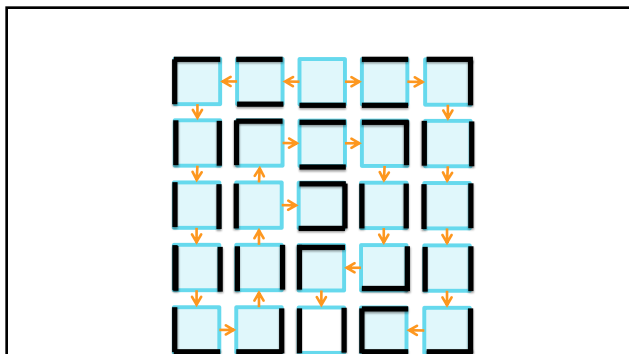
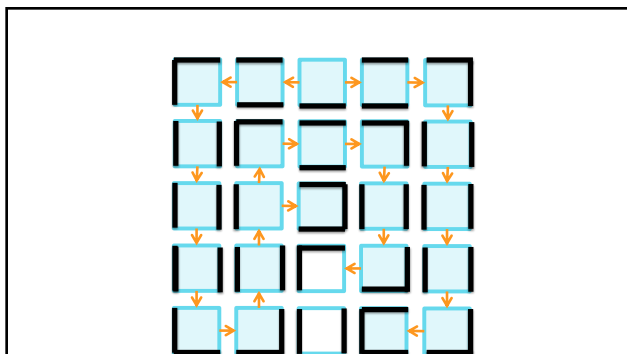
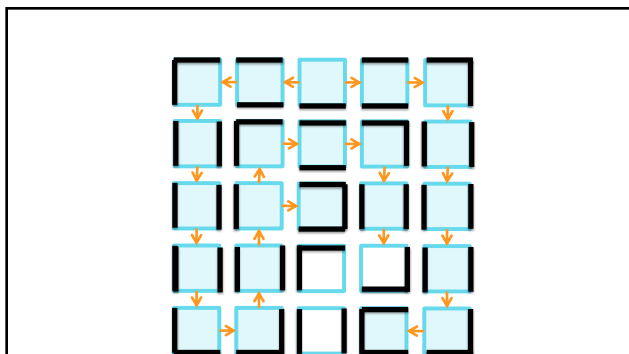
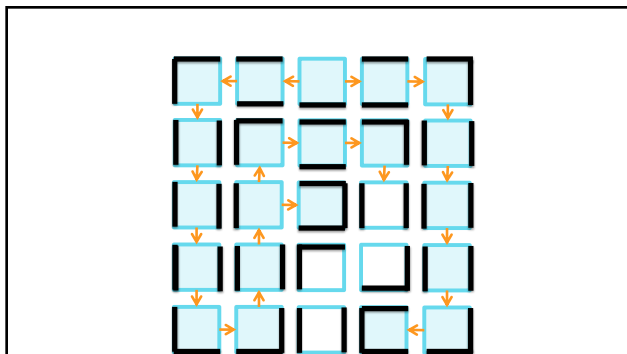
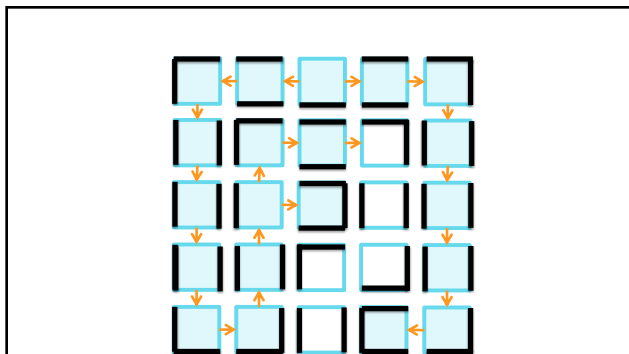


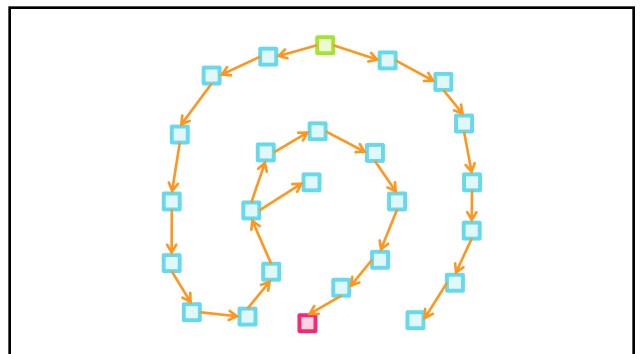
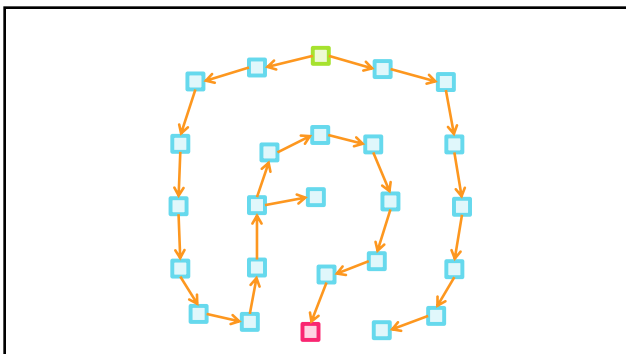
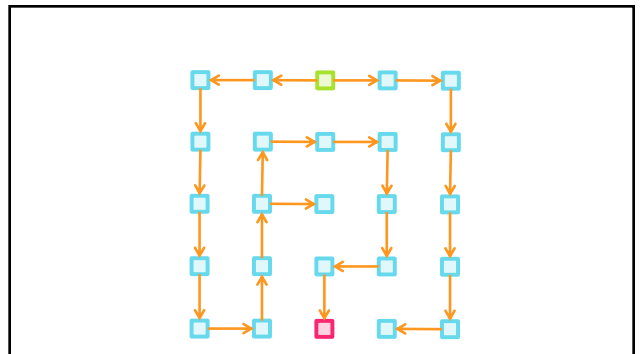
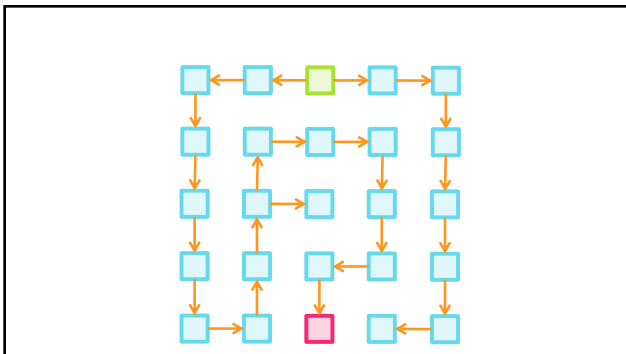
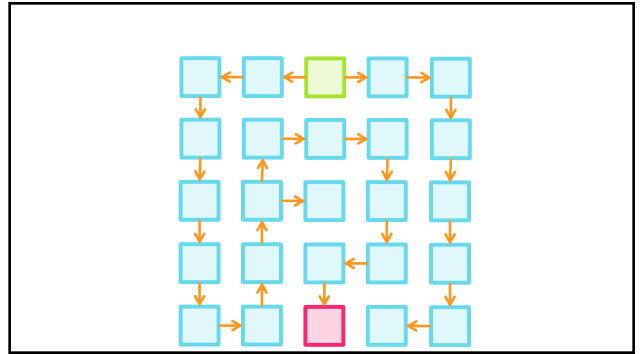
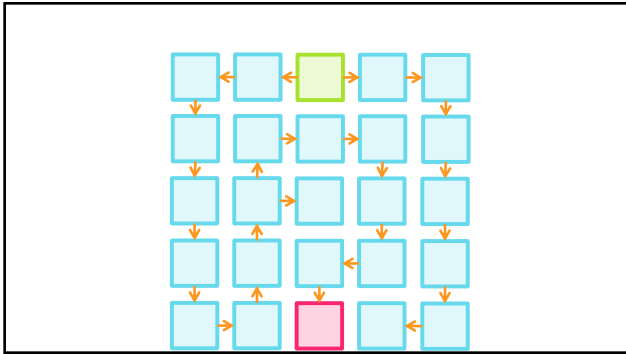
MAZE GAME

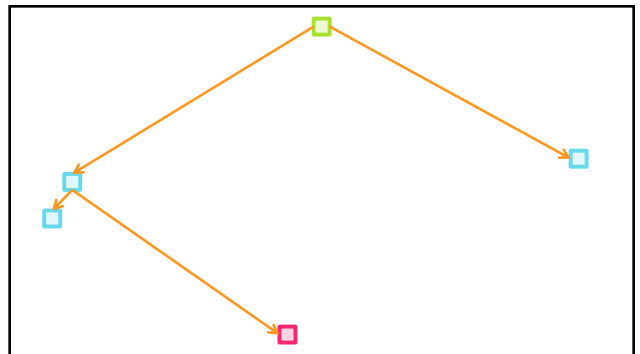
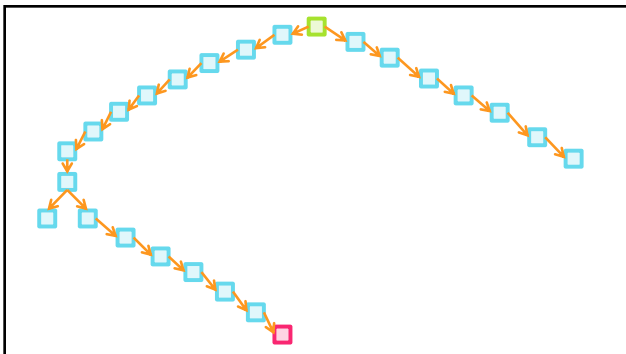
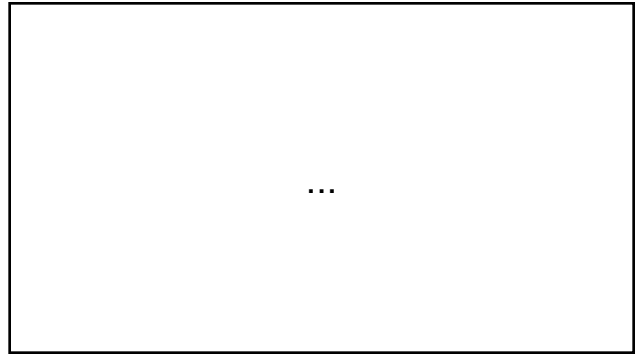
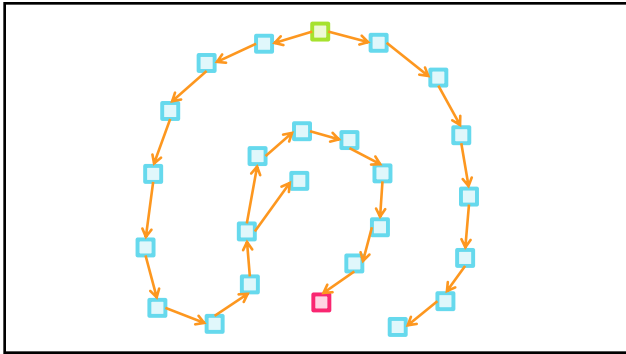












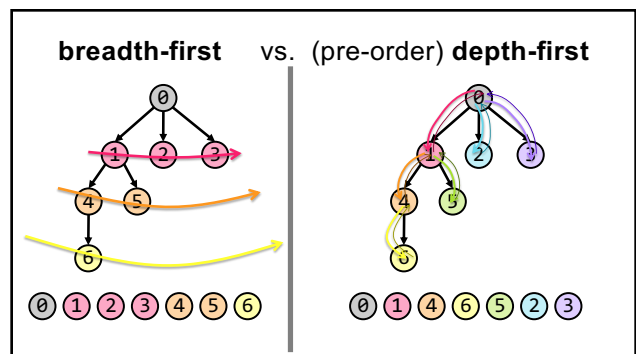
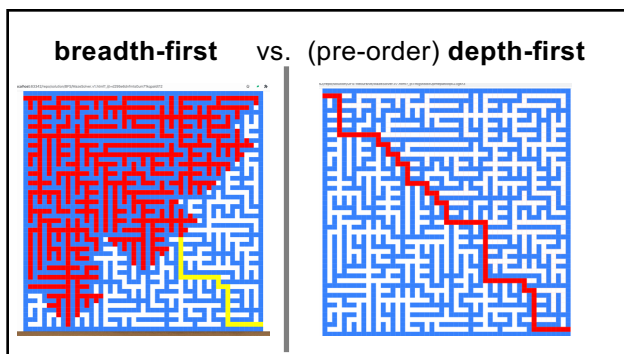
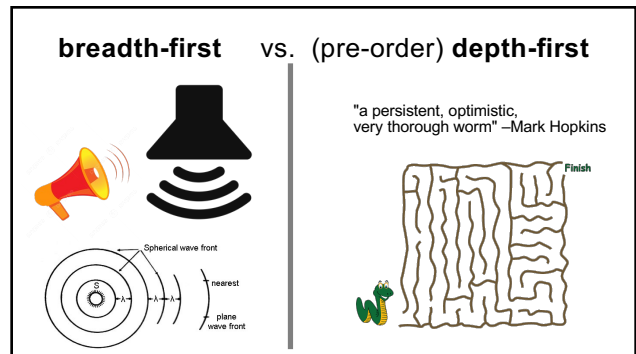
ANNOUNCEMENTS
today is Fun Friday?

WARMUP
vibes seem low, so here is
an extra a-MAZE-ing warmup
draw this maze as a tree

TODAY
tree traversal wrapup; quadtree reading

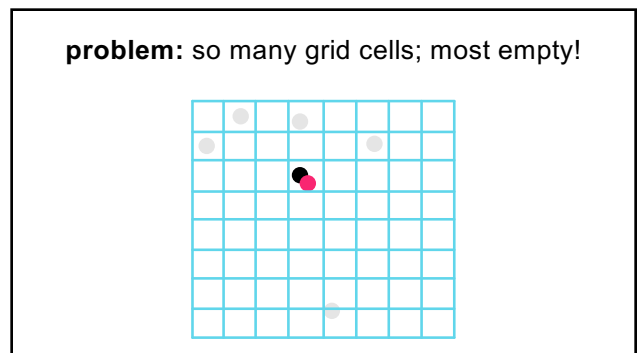
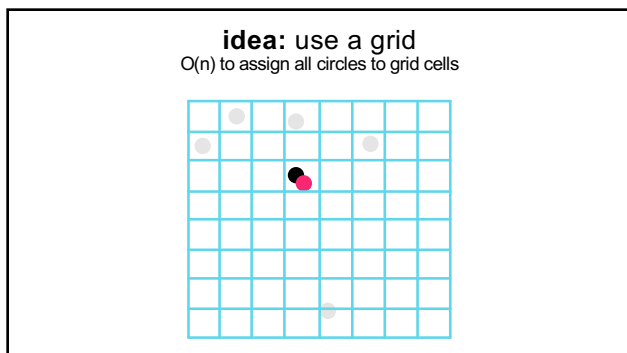
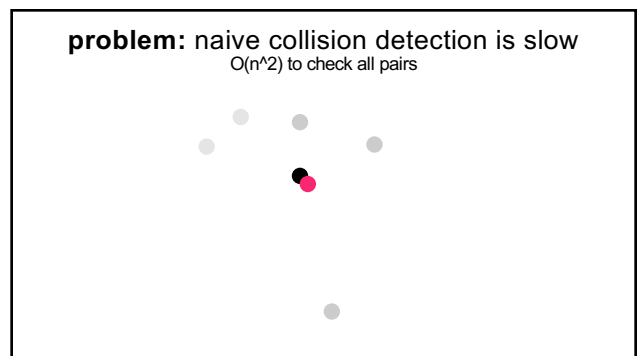
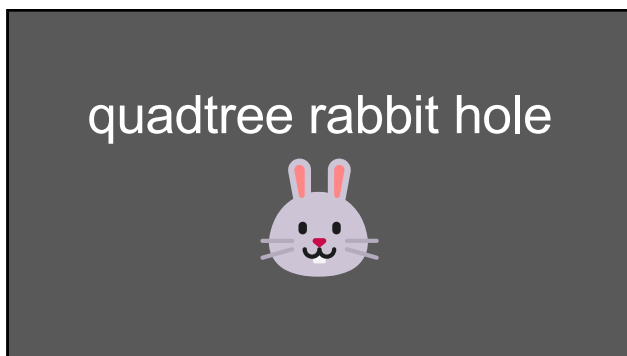
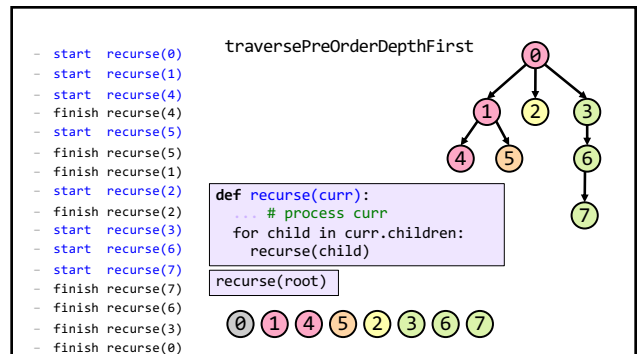
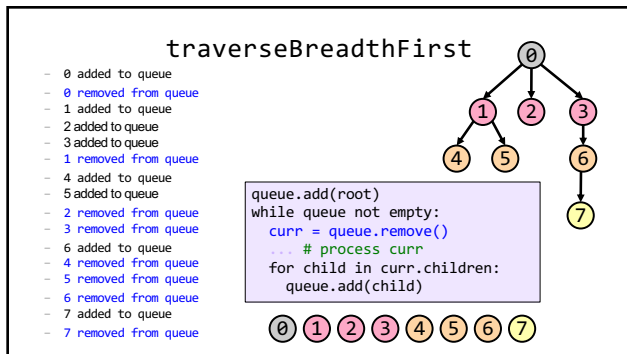
worms world party example

tree traversal

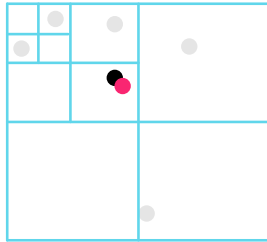


(switch slide deck)

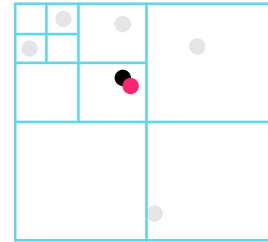
Tree Tutorial continued



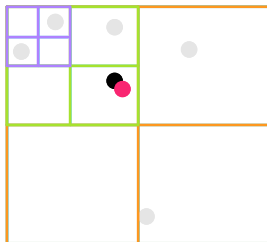
idea: "only have grid where you need it"



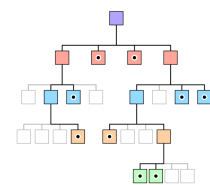
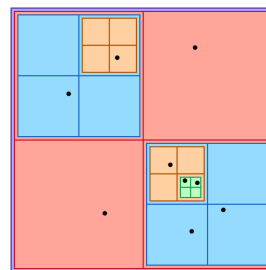
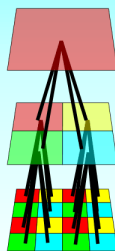
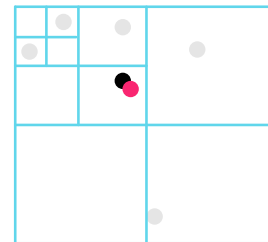
problem: how do we store this?



idea: a tree



idea: a tree



quadtree reading