Binary Trees

Introduction to Computer Science
Iris Howley
TODAY’S LESSON
Binary Trees

(A data structure for holding a different sort of data)
GAME: TWENTY QUESTIONS

• The "Knower" thinks of a noun
• The "Guesser(s)" ask yes/no questions in an attempt to guess the noun
• The Knower responds with only yes/no answers
• The Guesser aims to find the Knower's noun with fewer than 20 questions.
Hey Pixel.

Yeah, Dizzy?
I'm thinking of something. Can you guess it?

What is it?
Yes/No questions only.

Okay, is it a toy?
No. Toys are dumb.

Is it tasty?
Is it savory?

Yes.
Yes.

Does it come from the fridge?
Is it dog snacks?

No.
Is it from an animal?

No.
Is it chicken?

Yes.
No, but nice try, loser.

Is it tuna?
YASSSSSSSSSSSSSSSSSSSSSS.

I'm a winner?
Twenty Questions

Is it a toy? no

Is it tasty? yes

Is it savory? yes

Comes from fridge? no

Is it dog snacks? no

Is it chicken? no

Is it tuna? yes
Twenty Questions

Is it a toy?  no

Does it squeak? yes

Is it my squeaky goose? yes

Is it savory? yes

Comes from fridge? yes

Does it have 4 legs? yes

Is it dog snacks? no

Is it chicken? no

Is it tuna? no

Is a cat? yes

What does this structure resemble?
Twenty Questions

A more balanced game might be clearer...
Twenty Questions Tree

```python
__slots__ = [  
    _value,  
    _left,  
    _right  
]
```

Questions stored as the value

- Is it alive?
  - yes
  - Does it have 8 legs?
    - yes
      - Is it an octopus?
    - no
      - Is it food?
        - yes
          - Is it sweet?
        - no
          - Does it have 4 legs?
            - yes
              - Is it a pretzel?
            - no
              - Is it a table?

‘yes’ goes to the left
‘no’ to the right

If it’s a leaf, it’s a guess
QUESTIONS?

Please contact me!
Building Binary Trees

Introduction to Computer Science

Iris Howley
TODAY’S LESSON

Binary Trees

(A data structure for holding a different sort of data)
Twenty Questions Tree

```python
__slots__ = [value, left, right]
```

- **Is it alive?**
  - yes
  - no

- **Does it have 8 legs?**
  - yes
  - no

- **Is it food?**
  - yes
  - no

- **Is it sweet?**
  - yes
  - no

- **Does it have 4 legs?**
  - yes
  - no

- **Is it a pretzel?**
  - yes

- **Is it an octopus?**
  - yes

- **Is it a table?**
  - yes

‘yes’ goes to the left

‘no’ to the right

If it’s a leaf, it’s a guess

Questions stored as the value
Creating a Tree

- t2 = Tree('Does it have 8 legs?')
- t3 = Tree('Is it food?')
- mytree = Tree('Is it alive?', t2, t3)
Adding Nodes to the Tree

- Octopus?

- \( t_4 = \text{"octopus?"} \)

- \( \Rightarrow t_2 = \text{Tree('8 legs?',}t_4) \)
Accessing Nodes in a Tree

- `print(mytree.value)`  'Is it alive?'
- `print(mytree.left.value)`  'Does it have 8 legs?'
- `print(mytree.left.left.value)`  'Is it an octopus?'
What does this code do?

def mystery(self):
    if not self.right:
        return self
    return self.right.mystery()

See POGIL 45 on Binary Trees!
Twenty Questions Tree

- Is it alive?
  - no
  - yes
  - Does it have 8 legs?
    - yes
    - Is it an octopus?
      - yes
      - Is it a pretzel?
        - no
        - Is it a table?
          - yes
          - Is it sweet?
            - no
            - Does it have 4 legs?
              - yes
              - Is it food?
                - yes
                - Is it a table?
Binary Tree

• Let’s write a `contains( .. )` method for a tree
  ▪ (Application Question #3 from POGIL #45)

• `>>> from tree import *`
• `>>> mytree = Tree(99, Tree(33), Tree(66))`
• `>>> 66 in mytree`
• `True`  # `__contains__()` is implicitly called with “if ___ in <sequence>”
• `>>> 24 in mytree`
• `False`
Steps for Recursion

1. Know when to stop.
2. Decide how to take one, repeated step.
3. Break the journey down into that step plus a smaller journey.

Recursive Approach

- **REDUCE** the problem to smaller subproblem(s) (smaller version(s) of itself)
- **DELEGATE** the smaller problems to the recursion fairy *(formally known as induction hypothesis)* and assume they're solved correctly
- **COMBINE** the solution(s) of the smaller subproblems to reach/return the solution
Contains Method for Tree

• Stopping/Base Case:
  1. We’ve found the value
  2. Or we’re a leaf!

• Small step
  ▪ Check if we’re the value

• Break the journey down
  ▪ Check the left child, then the right (if it’s not in the left side)
Contains Method for tree

# __contains__() is implicitly called with “if ___ in <sequence>”
def __contains__(self, v):
    # Base case
    if self.value == v:
        return True

    l = v in self.left if self.left else False
    r = v in self.right if not l and self.right else False

    # if not l lets us skip the right side, if we found it in the left already
    return l or r
QUESTIONS?

Please contact me!
Using Binary Trees

Introduction to Computer Science
Iris Howley
TODAY’S LESSON
Using a Binary Tree

(Let's use our new data structure for something fun!)
Is it a toy?
Is it tasty?
Does it have 4 legs?
Is it savory?
Does it squeak?
Is it my squeaky goose?
Comes from fridge?
Is it dog snacks?
Is it chicken?
Is it tuna?
Is a cat?

Store our questions/answers so we can play the game with the computer.
q20.py

A program to play 20 Questions, using our tree data structure

See q20.py on the course website.
QUESTIONS?

Please contact me!
Leftover Slides
Steps for Recursion

• Know when to stop.
• Decide how to take one step.
• Break the journey down into that step plus a smaller journey.
**__str__** versus **__repr__**

- **__str__** returns a *human*-readable string representation of the object
  - Implicitly called with `print(object)` or `str(object)`
  - Also called with `'!s' . format(object)` in a format string

- **__repr__** produces a *machine*-readable string representation of the object
  - Implicitly called in interactive python: `>>> object`
  - Also called with `'!r' . format(object)` in a format string
OBJECT PERSISTENCE

Storing objects for future use.
POGIL Activity #41 – object persistence

- Find a partner and spend a few minutes discussing your responses to the POGIL worksheet, Question 1-3.

- Be prepared to report out your responses!

This is a brand new POGIL activity, let me know if you encounter any issues, typos, etc.
Time’s up!
Report out!
HOW MIGHT THIS BE USEFUL FOR OUR GAME OF TWENTY QUESTIONS?