

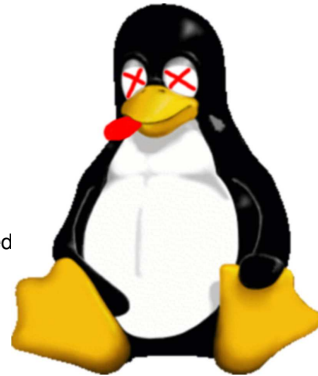
## The CS 5 Black Herald

### GIANT PENGUIN FOUND IN GALILEO CORRIDORS

Claremont (PPI): An enormous penguin, nearly six feet in length, was found in the corridors of Harvey Mudd College's Libra Complex late Wednesday evening. Scientists from Penguin Pleasures, a volunteer rescue group, said the animal appeared to have expired from an overdose of sugar. "Sadly, many people do not realize that penguins are terribly sensitive to sweets," stated Dr. D.I. Section as she examined the corpse. "I imagine that a well-meaning person must have intended to give it a treat."

The saddened campus plans to hold a moment of silence on the first day of final exams, since students are generally quiet and mournful during that time anyway.

Read 5.6-5.7!

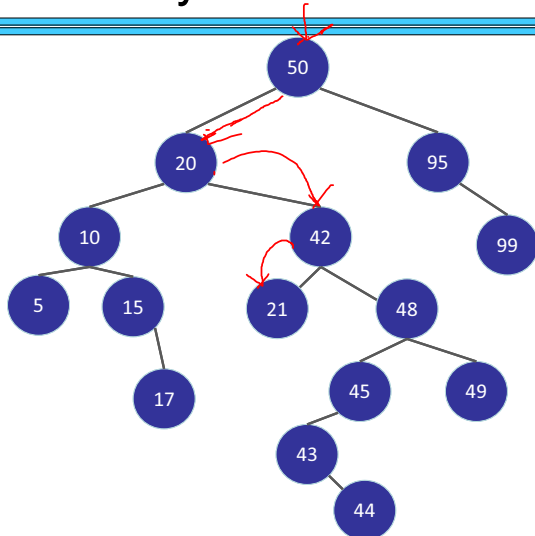


## Abstract Data Types (ADTs)

- Abstract set: insert(x), delete(x), find(x)
- Average and worst cases:

	Insert (avg)	Delete (avg)	Find (avg)	Insert (worst)	Delete (worst)	Find (worst)
Dictionary	$O(1)$	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Python List	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
BST	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

## Binary Search Trees



## Binary Search Trees!

```
>>> tree = None
>>> tree = insert(42, tree)
>>> tree
(42, None, None)
>>> tree = insert(47, tree)
>>> tree
(42, None, (47, None, None))
>>> find(27, tree)
False
>>> find(47, tree)
True
>>> inorder(tree)
[42, 47]
```

## inorder(tree)

```
def inorder(tree):
    if tree is None:
        return []
    else:
```

(42, 1/2, 2/2)



None

↓ ↓ ↓  
[1,2,3] + [42] + []  
~~[1,2,3]~~ [1,2,3,42]

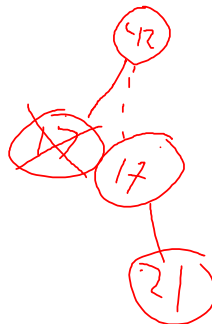
## find(x, tree)

```
def find(x, tree):
    if tree is None:
        return False
    else:
```



## insert(x, tree)

```
def insert(x, tree):
    if tree is None:
        return (x, None, None)
    else:
```



This will be handy when we save a tree to a file!



## Twenty Questions Game

- Demo!
- Playing the game gives back a new tree!
- How do we save the tree?
- How do we restore the tree? (Optional, but recommended!)

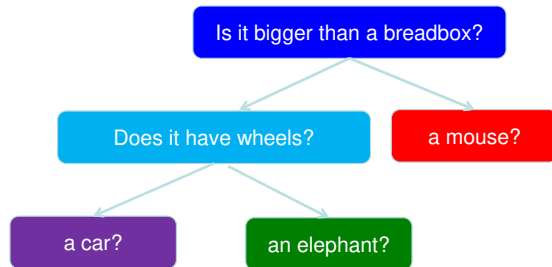
Is it bigger than a breadbox?

```
("Is it bigger than a breadbox?",
 ("an elephant", None, None),
 ("a mouse", None, None)
)
```

an elephant?

a mouse?

## Bigger Trees



```

("Is it bigger than a breadbox?",
 → ("Does it have wheels?", ("a car", None, None), ("an elephant", None, None)),
  ("a mouse", None, None)
)
  
```

## play(tree)

```

def play(tree):
    if leaf(tree):
        return playLeaf(tree)
    else:
        root, yesChild, noChild = tree
        if yes(root + " "):
            ...
        else:
            ...
  
```

Imagine a function called playLeaf(tree)!

Fill this in! Then talk with a neighbor about what playLeaf will need to do.

Worksheet!

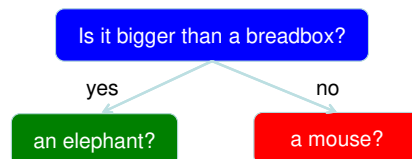


## Saving a Tree to a File

What does the file look like?

```

("Is it bigger than a breadbox?",
 ("an elephant", None, None),
 ("a mouse", None, None)
)
  
```



```

Is it bigger than a breadbox?
Internal node
an elephant
Leaf
a mouse
Leaf
  
```

## Writing to a File

```

saveFile = open("tqtreetree.txt", "w")
saveTree(saveFile, tree)
saveFile.close()
  
```

...

```

def saveTree(saveFile, tree):
    root, yesChild, noChild = tree
    if isLeaf(tree):
        print("Leaf", file = saveFile)
        print(root, file = saveFile)
    else:
        ...
  
```

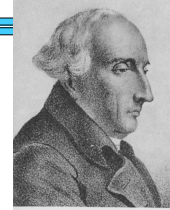
## Reading a Tree

```
try:
    loadFile = open(TREE_FILE, "r")
    tree = loadTree(loadFile)
    loadFile.close()
except FileNotFoundError:
    pass
...
def loadTree(loadFile):
    line = loadFile.readline().strip()
    if line == "Leaf":
        answer = loadFile.readline().strip()
        return (answer, None, None)
    else:
```

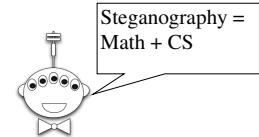
*Handwritten notes:*

- read* (circled around "r")
- default tree?* (with an arrow pointing to the `pass` statement)

## The Lagrange Polynomial Method!



Joseph Louis Lagrange.

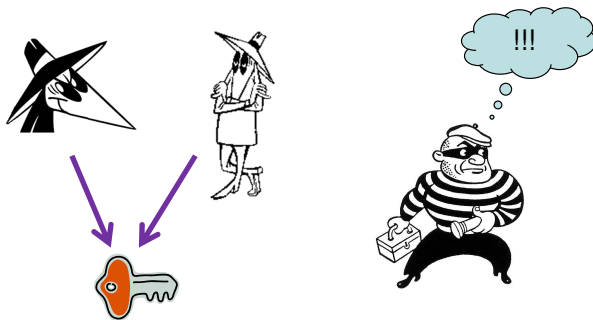


Suppose we have a secret...

And we don't trust *any* single person with it.

## The Lagrange Polynomial Method!

Wouldn't it be cool if we could split a secret into  $n$  parts, such that any  $k$  people could get it back?



## Lagrange Basis Functions

Consider the following *basis* function:

$$\frac{x - x_0}{x_1 - x_0} \times \frac{x - x_2}{x_1 - x_2}$$

What is its value at:

- $x = x_0$ ?
- $x = x_2$ ?
- $x = x_1$ ?
- Arbitrary  $x$ ?

## Lagrange Basis Functions

Let  $l_j$  be the basis function for  $x_j$ :

$$l_j = \prod_{m \neq j} \frac{x - x_m}{x_j - x_m}$$

What is its value at:

- $x = x_m$  (for any  $m \neq j$ )?
- $x = x_j$ ?
- Arbitrary  $x$ ?

## A Polynomial Through $k$ Points

Let  $l_j$  be the basis function for  $x_j$ :

$$l_j(x) = \prod_{m \neq j} \frac{x - x_m}{x_j - x_m}$$

Now define  $L(x) = \sum_{j=0}^k y_j l_j(x)$

By definition of the basis function,  $L(x) = y_j$  at all  $x_j$ .

We **don't care** what  $L(x)$  is at other values of  $x$ .

## A Concrete Example

Let  $x_0 = 3, x_1 = 4, x_2 = 5, y_0 = 1, y_1 = 4, y_2 = 3$ . Then:

$$l_0 = \left( \frac{x-4}{3-4} \right) \left( \frac{x-5}{3-5} \right) = \frac{(x-4)(x-5)}{2}$$

$$l_1 = \left( \frac{x-3}{4-3} \right) \left( \frac{x-5}{4-5} \right) = -(x-3)(x-5)$$

$$l_2 = \left( \frac{x-3}{5-3} \right) \left( \frac{x-4}{5-4} \right) = \frac{(x-3)(x-4)}{2}$$

$$L(x) = \frac{(x-4)(x-5)}{2} - 4(x-3)(x-5) + \frac{3(x-3)(x-4)}{2}$$

## Shamir's Secret Sharing



This brings us to Shamir's method for sharing a secret  $s$  such that any  $k$  of  $n$  people can reconstruct it:

1. Pick a polynomial of degree  $k-1$ , with random coefficients  $a_i$ :  

$$y = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x^1 + s$$
2. For each holder of the secret, pick a random  $x$  and use the polynomial to calculate a corresponding  $y$ .
3. Reconstruct the secret by creating a Lagrange polynomial of degree  $k-1$  and evaluating it at  $x = 0$ .



Wow! Is it really that easy?