



$2^{(n-1)}$
guys

Question:

Consider an arbitrary binary tree of height n . What is the maximum number of leaves that a binary tree of height n can have? What is the minimum number of leaves that a binary tree of height n can have?

Hint: Consider when the maximum amount of subtrees is achieved, and when the minimum amount of subtrees is achieved.

Leaf /
Maximum:
rounded up (n
/ 2)

Max:
 $2^{(n-1)}$

min: 1 leaf,
max:
 $2^{(n-1)}$
leaf

Harambee
is alive

min: 1
(the
root)

2^n

$n/1 =$
 $n/2 =$ max:
size $n-1$

ibra java
men
shuned

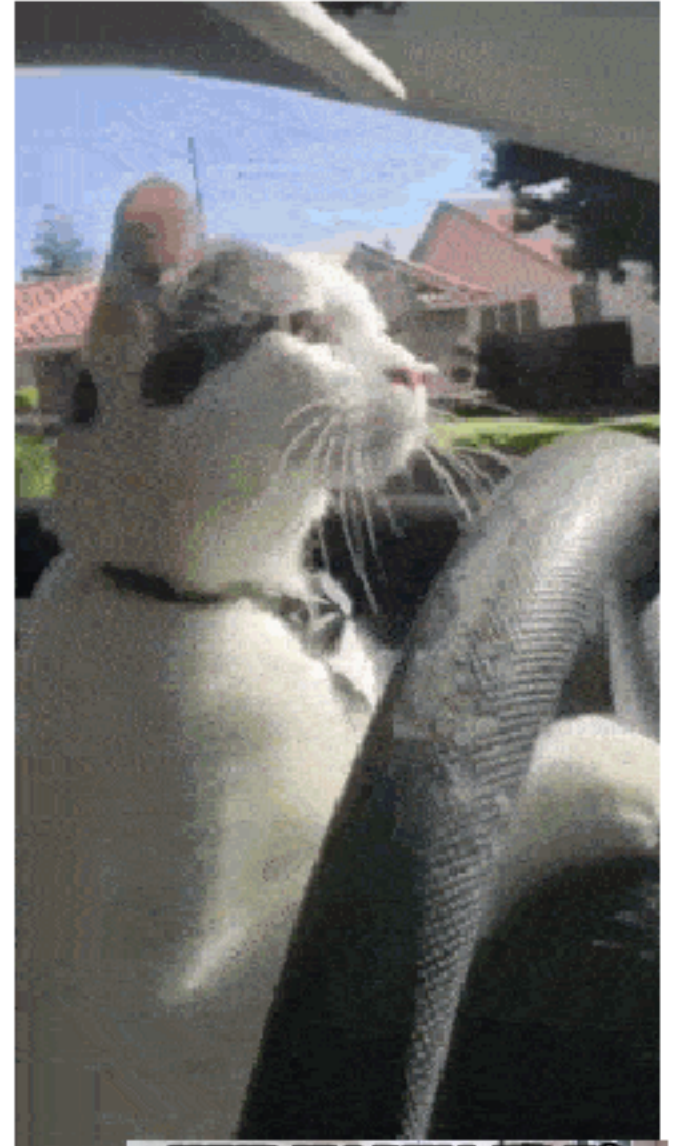
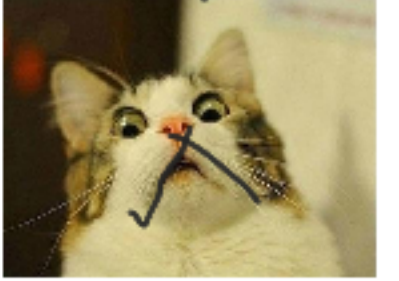
height n ,
height
 $\log_2(n-1)$



at
my is im
still the
family guy

ibrajava mentioned $\{ \leq$

Scallops



WYD IF I PULL UP TO



YOU'RE CRIB IN THIS



Question:

Consider an arbitrary Binary Tree of size n . What is the maximum height h_1 that can be achieved? What is the minimum height h_2 that can be achieved? Justify your answer.

Hint: Consider modelling a few examples of arbitrary size n and see if you can find a pattern.

Max:
n

Mjn:
 ~~$\log_2(-1)$~~



Question:

Consider an arbitrary Binary Tree of size n . What is the maximum height h_1 that can be achieved? What is the minimum height h_2 that can be achieved? Justify your answer.

Hint: Consider modelling a few examples of arbitrary size n and see if you can find a pattern.

**max
height n ,
min size 2**

**max =
 n for h_1**

**min: 2
if $n \neq 1$**

**Max: n / Min:
 $\text{floor}(\log_2(n))$
+ 1**

**Max height n ,
min height
 $\text{floor}(\log_2(n))$
+ 1**

**max
 $h_1: n$**

**Max: n /
Min:
 $\log_2(n) + 1$**

**max: n , min:
 $\text{ceil}(\log_2(n+1))$**

slide 8 →

Question:

Consider the search operation for a *Binary Search Tree*. What is the **worst case** time complexity for the search operation? When does it happen? Give an example of a tree that would cause this to happen.

Are BST operations *in general* always $\mathcal{O}(\log n)$? Why or why not?

Hint: Recall our discussion about the Balancing Factor and AVL Trees earlier in the slides.

Worst case is $\mathcal{O}(n)$, it happens when every node only has one child and you search for the last item in the tree.

$\mathcal{O}(n)?$

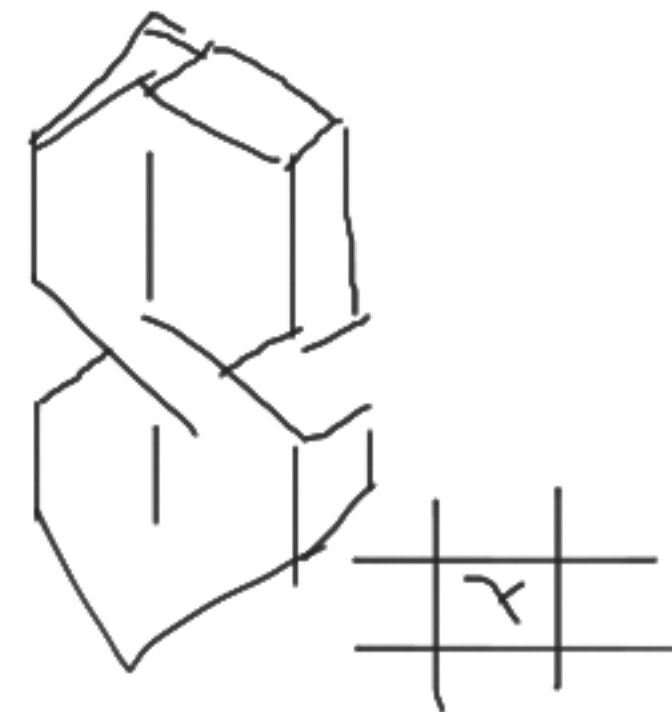
$\mathcal{O}(\text{height})$

Worst case is when the element isn't even in the tree

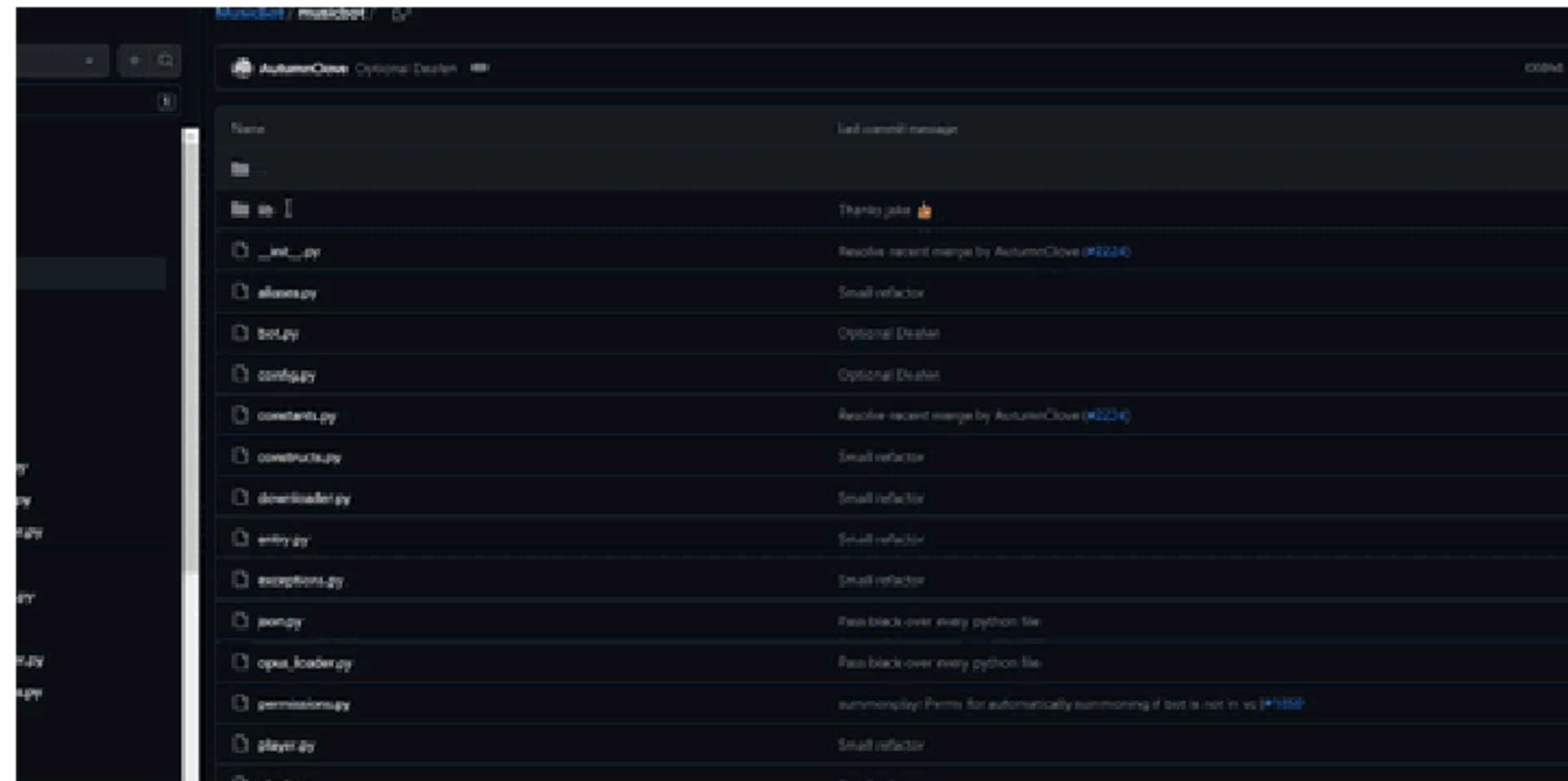
When it's more than the max or less than min or not even found in the leaves

No because BSTs are not necessarily balanced.

When you have a tree where each node is smaller than the rest, so a linear one, and you search for something smaller than everything in the tree.



$\mathcal{O}(n)$



Name	Last commit message
..	
..	Thanks jake 🍷
__init__.py	Resolve recent merge by AutumnClove (#2234)
aliases.py	Small refactor
bot.py	Optional Dealer
config.py	Optional Dealer
constants.py	Resolve recent merge by AutumnClove (#2234)
constructs.py	Small refactor
downloader.py	Small refactor
entry.py	Small refactor
exceptions.py	Small refactor
pongy	Pass black over every python file
opus_loader.py	Pass black over every python file
permissions.py	summonplayn: Perms for automatically summoning if bot is not in vc (#1888)
player.py	Small refactor
plugins.py	Production ready

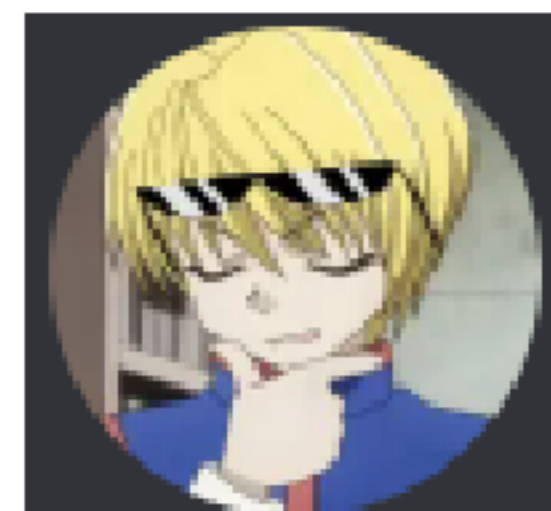
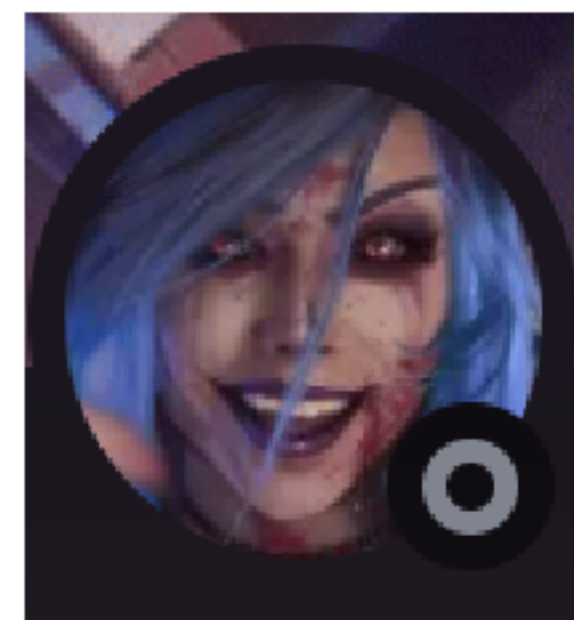
thanks for inspiration







easter egg



binary_tree of sam198

!?!?!?!?!?
!?!?!?!?!?

this is worrying
- Ibrahim











csc148 was
here