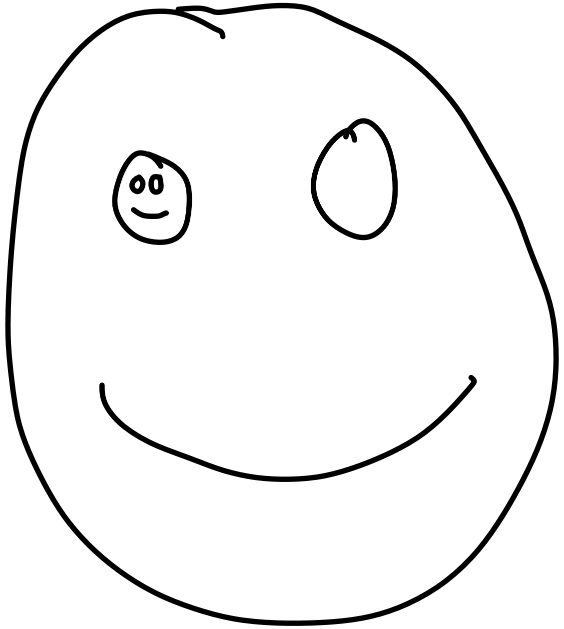
Algorithms and Data Structures for Data Science
lab_recursion

CS 277
Brad Solomon

February 23, 2024



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

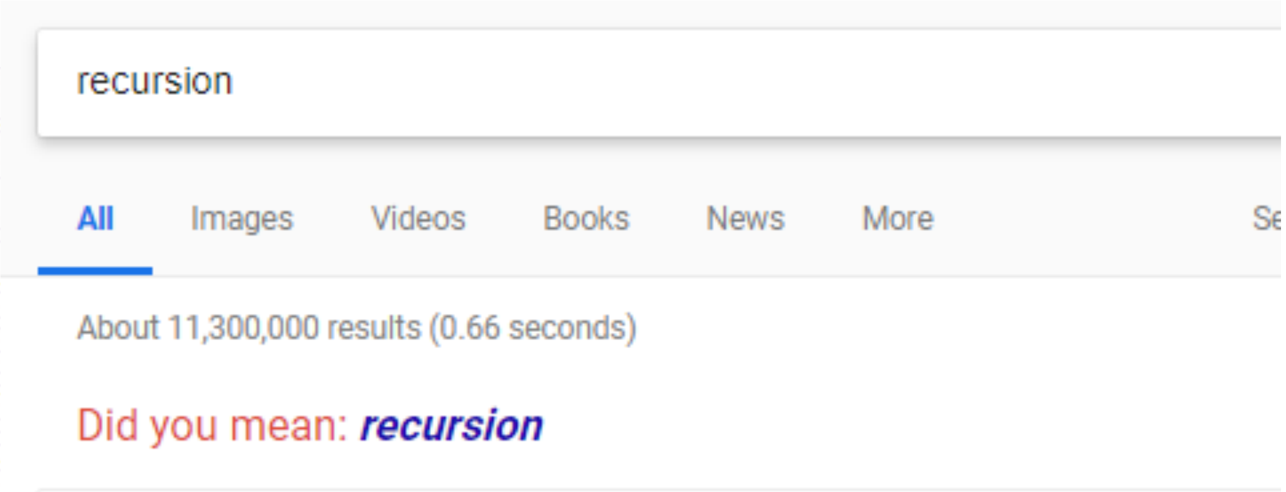
Learning Objectives

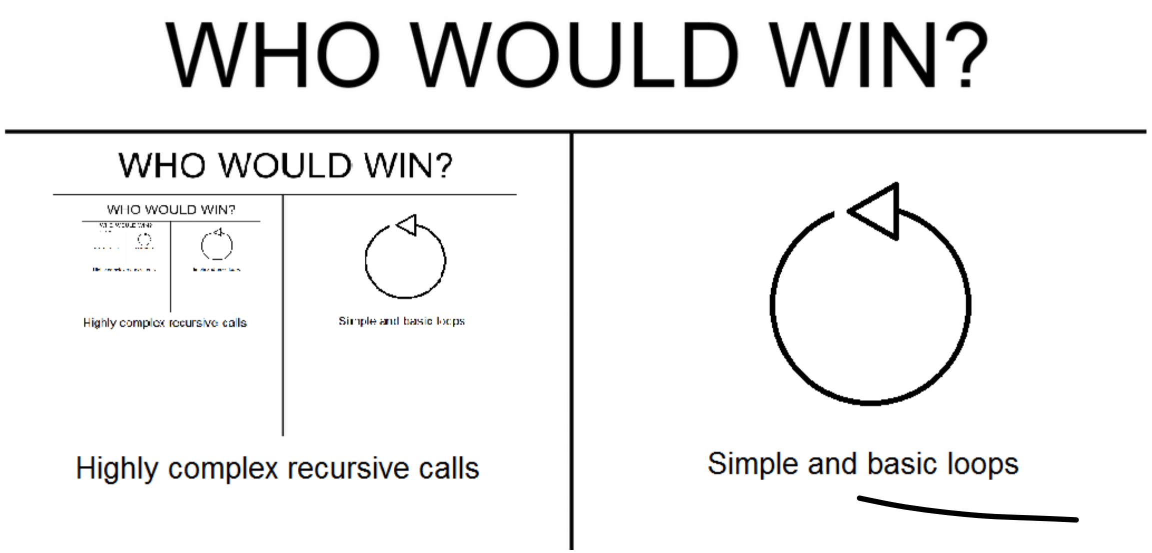
Review fundamentals of recursion

Implement recursive functions to handle a variety of tasks

Recursion


SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION
SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION


recursion
All
Images
Videos
Books
News
More
About 11,300,000 results (0.66 seconds)
Did you mean: recursion


WHO WOULD WIN?
WHO WOULD WIN?
WHO WOULD WIN?
Highly complex recursive calls
Simple and basic loops
Highly complex recursive calls
Simple and basic loops

Recursion

The success or failure of this lab (and the time it takes you) depends on your ability to answer the following:

Base Case: What is the smallest sub-problem? What is the trivial solution?

Recursive Step: How can I reduce my problem to an easier one?

Combining: How can I build my solution from recursive pieces?

Lets work together to brainstorm some of the following functions!

Each exercise a fun new twist!

Sum of Digits:

Triangle:

Palindrome:

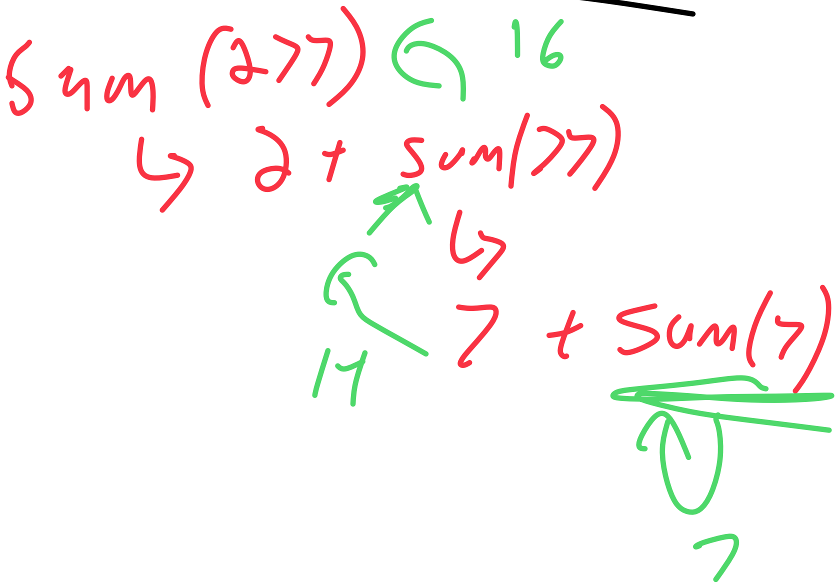
Fibonacci:

Bonus List Partitioning:

Recursion Practice: Sum of Digits

Given a number, return the numerical value of summing each digit.

277 2 + 7 + 7 = 16



111 1 + 1 + 1 = 3

Recursion Practice: Sum of Digits

Given a number, return the numerical value of summing each digit.

Base Case: Any length 1 digit → return the digit!

n % 10 == n

↳ x < 10   len(str(n)) == 1

Recursive Step: Separate one digit

call function again [sum( )]

Combining: One # + the rest | one digit + sum(<the rest>) |

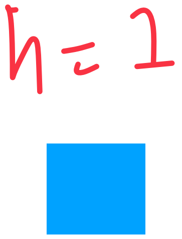
277

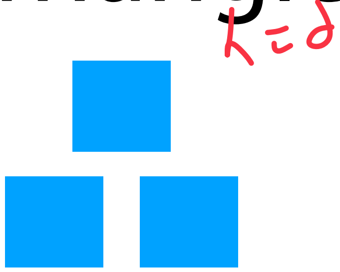
2 + sum(77) = 16

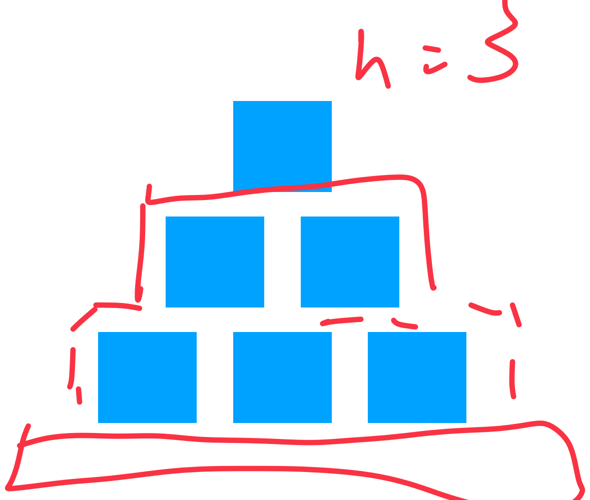
7 + sum(27) = 16

ABC → BC → C

Recursion Practice: Triangle







Given the height of a triangle, how many total blocks were used to make it?

Base Case: h = 0 → 0 blocks

h = 1 → 1 block

Recursive Step: Triangle of height one smaller

Combination Step: ???

Recursion Practice: String Palindrome

Given a string, return whether it is a palindrome or not (True or False)

AAA Yes

racecar

racetrack

Recursion Practice: String Palindrome

"A B A"

Given a string, return whether it is a palindrome or not (True or False)

Base Case:

0 letters - palindrome!

2 letters - may be palindrome

"AA" ↓ ""

1 letter - is a palindrome!

"AXYA"

Recursive Step: Remove first & last letter

"ABC"

↳ Look at first & last letter. If same recurse!

If diff return False

Combining: ??

Recursion Practice: Recursive Fibonacci

0, 1, 1, 2, 3, 5, 8 ...

Given a number *n*, return the *nth* Fibonacci number:

$$Fib(n) = Fib(n-1) + Fib(n-2)\,,\ n > 1$$

Fib(2)
↳ Fib(0) & Fib(1)
↳ No Fib(-1) & Fib(-2)
↳ No Fib(-1)

Base Case:

n = 0 → 0

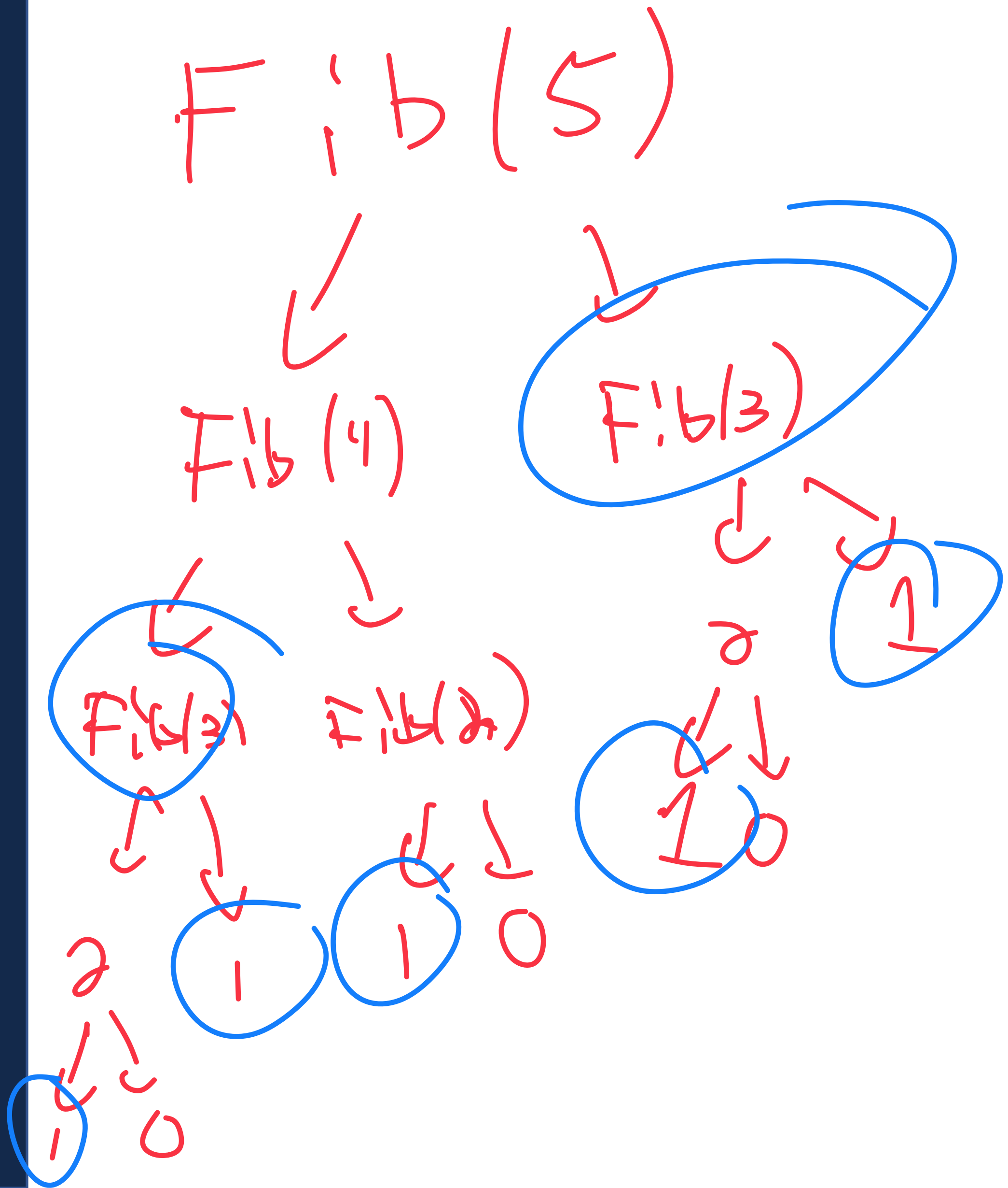
???

Recursive Step: # n

↳ Fib(n-1) & Fib(n-2)

Combining:

nth # = Fib(n-1) + Fib(n-2)



Fib(5)
Fib(4)
Fib(3)
Fib(3)
Fib(2)
2
1
1
0
1
0
2
1
0

Using all elements in a list, can we make two lists which have equal sums?

Input

[4, 3, 1]	([], [])			
[3, 1]	([4], [])		([], [4])	
[1]	([3, 4], [])	([4], [3])	([3], [4])	([], [3, 4])
[]	([1, 3, 4], [])	([1, 4], [3])	([1, 3], [4])	([1], [3, 4])
	([3, 4], [1])	([4], [1, 3])	([3], [1, 4])	([], [1, 3, 4])

Recursive List Partitioning

Base Case: When my input list is empty, I have tried every permutation

Recursive Step: Given list L, pop() L[0] to left ***and*** right and recurse on both

Combination Step: If either partition recursion is True, return True