## CS 61A Topical Review Session: Tree Recursion

May 6, 2019

[Slides](https://docs.google.com/presentation/d/1qRYWt-4xAi4gisrK44Mst3Gj1HnCEWKRCudhAgXX28g/edit?usp=sharing) (with solutions)

*Tree Recursion is like regular recursion, but involves* ***2 or more recursive calls!***

Common tree recursion problems:

* Explore the best of two or more possible options
* Count the total number of ways to achieve something

Tips for solving tree recursion problems:

* At each recursive step, think about what options you have to continue solving the problem
  + Each option should have its own recursive call!
* Think about how to combine your recursive calls
  + Eg: sum, max, min

**Problem 1**

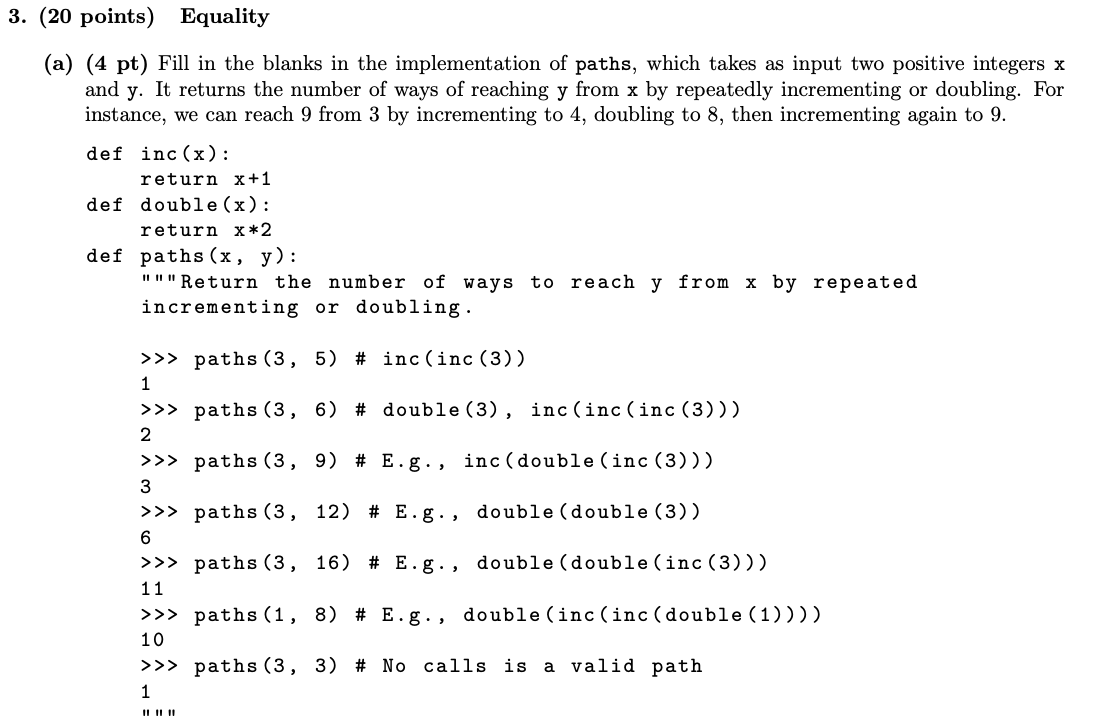
def sum\_largest(n, k):

"""Given non-negative ints n, k, return the sum of the k largest digits of n.   
>>> sum\_largest(2018, 2) # 2 and 8 are the two largest digits (larger than 0 and 1).   
10   
>>> sum\_largest(12345, 10) # There are only five digits, so all are included in the sum.   
15   
"""

if\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:   
 return 0  
a = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
return \_\_\_\_\_\_\_\_\_\_\_\_\_\_(a, b)

**Spring 2013 Final Q3A**

****

if x > y:

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

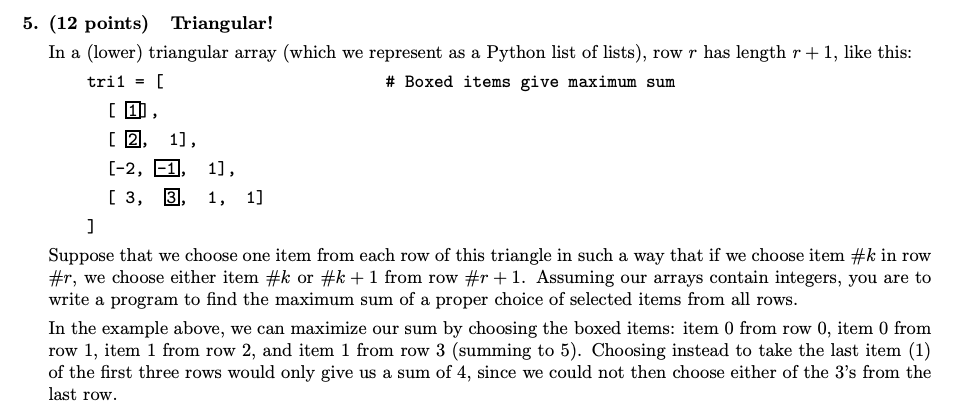
elif x == y:

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

else:

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Spring 2016 Final Q5A**



Fill in the blanks in the following recursive program to find the maximum sum:

def triangle\_sum(tri):

""" Given that tri is a triangular array , return the maximum sum attainable by selecting one item from each row , where if item #k is selected from row #r , either item #k or item #k +1 is selected from row #r +1. """

rows = len(tri)

def partial\_sum(r, k):

""" The maximum partial sum of items from rows #R , R +1 , ... starting from selecting item #K in row #R . """

if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ :

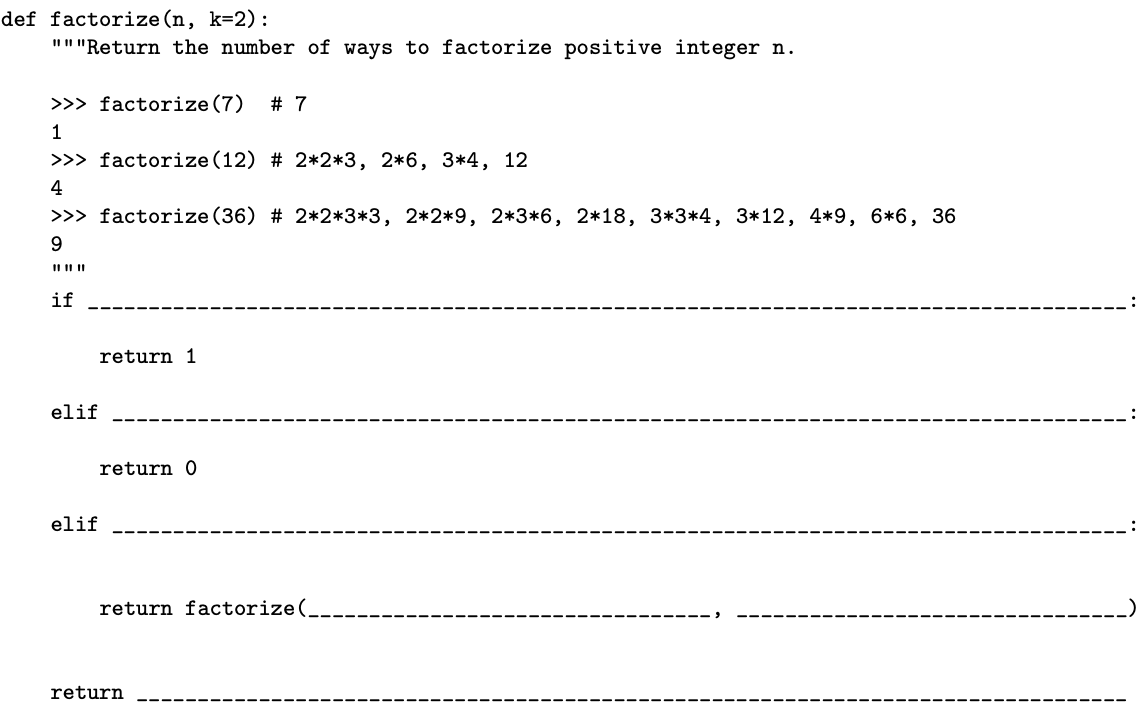
return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

else:

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return partial\_sum(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Spring 2018 Final Q5B**



**Summer 2017 Final Q7**

