Exercise 3

3/3 points (graded)

For these questions, you'll be asked to give the big-O upper bound runtime for some Python functions. In your answer, please omit the "O( )" portion of the answer and simply write a mathematical expression.

Use +, -, / signs to indicate addition, subtraction, and division. Explicitly indicate multiplication with a \* (ie say "6\*n" rather than "6n"). Indicate exponentiation with a caret (^) (ie "n^4" for n4). Indicate base-2 logarithms with the word log2 followed by parenthesis (ie "log2(n)").

What this all means is if an answer is, for example, O(n4), please simply write "n^4" in the box.

1. What is the big-O upper bound runtime for the function look\_for\_things, where *n* is defined as the number of elements in myList?

NUMBER = 3

def look\_for\_things(myList):

"""Looks at all elements"""

for n in myList:

if n == NUMBER:

return True

return False



Loading

1. What is the big-O upper bound runtime for the function look\_for\_other\_things, where *n* is defined as the number of elements in myList?

NUMBER = 3

def look\_for\_other\_things(myList):

"""Looks at all pairs of elements"""

for n in myList:

for m in myList:

if n - m == NUMBER or m - n == NUMBER:

return True

return False



1. What is the big-O upper bound runtime for the function look\_for\_all\_the\_things, where *n* is defined as the number of elements in myList?

You do not need to account for the runtime of get\_all\_subsets; the code is provided so you can see how many subsets it generates, as that **will** be a factor in your answer.

def get\_all\_subsets(some\_list):

"""Returns all subsets of size 0 - len(some\_list) for some\_list"""

if len(some\_list) == 0:

# If the list is empty, return the empty list

return [[]]

subsets = []

first\_elt = some\_list[0]

rest\_list = some\_list[1:]

# Strategy: Get all the subsets of rest\_list; for each

# of those subsets, a full subset list will contain both

# the original subset as well as a version of the subset

# that contains first\_elt

for partial\_subset in get\_all\_subsets(rest\_list):

subsets.append(partial\_subset)

next\_subset = partial\_subset[:] + [first\_elt]

subsets.append(next\_subset)

return subsets

NUMBER = 3

def look\_for\_all\_the\_things(myList):

"""Looks at all subsets of this list"""

# Make subsets

all\_subsets = get\_all\_subsets(myList)

for subset in all\_subsets:

if sum(subset) == NUMBER:

return True

return False

