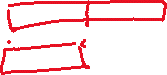
# Lecture 33: How Slow (or fast?) is Recursion?

In the *worst case,* if you use **linear search** to find a target item **x** in a list of **n** items, what is the running-time proportional to?



In the *worst case*, if you use **binary search** to find a target item **x** in a list of **n** items, what is the running-time proportional to?



In the *worst case*, if you use a fast sorting algorithm such as **mergesort**, or **Python’s built-in sort**,to sort a list of **n** items, what is the running-time proportional to?



## The contains Function

**contains(x, lst)** returns **True** if **x** is **Ist**, and **False** otherwise.



How could you **test** that **contains(x, lst)** works correctly?

**def** **test\_contains**(contains):



**print**(f'calling test\_contains({contains.\_\_name\_\_}) ... ', end**=**'')



**assert** contains(3, []) **==** False



**assert** contains(3, [3]) **==** True

**assert** contains(1, [3]) **==** False

**assert** contains(4, [3]) **==** False



**assert** contains(3, [1, 3, 5]) **==** True

**assert** contains(1, [1, 3, 5]) **==** True



**assert** contains(5, [1, 3, 5]) **==** True

**assert** contains(0, [1, 3, 5]) **==** False



**assert** contains(2, [1, 3, 5]) **==** False

**assert** contains(**-**1, [**-**1, 3, 51, 100]) **==** True

**assert** contains(100, [**-**1, 3, 51, 100]) **==** True

**assert** contains(2, [**-**1, 3, 51, 100]) **==** False

**assert** contains(200, [**-**1, 3, 51, 100]) **==** False

**print**('passed')



Why do this **not** guarantee that **contains(x, lst)** works correctly? What could we do to be sure it works correctly?

## Built-in



def contains\_builtin(x, lst):

return x in lst



## For-each Loop

def contains\_for\_each(x, lst):

for item in lst:

if item == x:

return True

return False

## For-each Loop with range

def contains\_for\_range(x, lst):

for i in range(len(lst)):



if lst[i] == x:



return True

return False

## While Loop

def contains\_while(x, lst):

i = 0

while i < len(lst):

if lst[i] == x:

return True

i += 1

return False



## Recursive Linear Search



def contains\_recursive\_linear(x, lst):

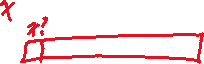


if lst == []:



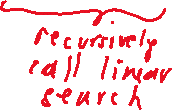
return False

elif lst[0] == x:



return True

else:



return contains\_recursive\_linear(x, lst[1:])



## Recursive Binary Search

def contains\_recursive\_binary(x, lst):



"""Assumes lst is sorted.

"""

if lst == []:



return False

else:



mid = len(lst) // 2

if lst[mid] == x:



return True

elif x < lst[mid]:



return contains\_recursive\_binary(x, lst[:mid])



else: # lst[mid] < x

return contains\_recursive\_binary(x, lst[mid+1:])



How can you test the **performance** of these different functions?

def test\_performance():  
 contains\_functions = [contains\_builtin, contains\_for\_each,  
 contains\_for\_range, contains\_while,  
 #contains\_recursive\_linear,  
 contains\_recursive\_binary]



lst = list(range(1000000))



print()

for contains in contains\_functions:



print(f'testing: {contains.\_\_name\_\_:25} ', end='')



start = time.time()



for i in range(10):



contains(-1, lst)



elapsed\_time = time.time() - start



print(f'{elapsed\_time:.2f} seconds')

