"""Marc Ojalvo and Daniel Licht
CMPS 1500
Thursday 3:30-4:45
Lab 10 Part 1
04/26/2018"""

lab10pr1

a) Insert time:

If n = len(L), insert time = 0.022881031036376953, because the insert function creates a tree with every item in the list, since the total length is equal to the amount of items in the list.

If n= len(L)//2, insert time = 0.005978822708129883, because the insert function creates a tree with half the amount of the items in the list. This reduces the run time to ¼ of the running time, when using the complete list.

If n= len(L)//5, insert time = 0.001233816146850586, because the insert function creates a tree with a fifth of the number of items from the list. This reduces the run time to $\frac{1}{20}$ of the run time it takes when using the complete list.

If n= len(L)//10, insert time = 0.000492095947265625, because the insert function creates a tree with a tenth of the number of items from the list. The reduces the run time to $\frac{1}{40}$ of the run time it takes when using the complete list.

If n= len(L)//20, insert time = 0.00025272369384765625, because the insert function creates a tree with a twentieth of the number of items form the list. This reduces the run time to $\frac{1}{80}$ of the run time it takes when using the complete list.

If you notice, the run time decreases to a ¼ of the run time when the list is cut in half. This remains constant for every time the list is cut in halves again. This shows that when the list is reduced by a fifth, tenth and twentieth, the run times is equal to ¼ times the reciprocal of the number the list is being reduced by.

b) Find time:
If n = len(L), find time = 0.05400705337524414
If n= len(L)//2, find time = 0.028384923934936523
If n= len(L)//5, find time = 0.012158870697021484
If n= len(L)//10, find time = 0.005866050720214844
If n= len(L)//20, find time = 0.003551959991455078

Every time the list is cut in half, run time is reduced by a half. This makes sense because the find function, during its worst case, searches each level of the tree until it reaches the bottom and does not find the item. By reducing the length of the list by half, the find function takes half as long to run.

c) Our operations simulated the worst-case scenario, because we searched for an item that does not exist in the tree. If we were to search for the root of the tree 500 times, we would end up with the best-case scenario. An average of best and worst-case scenario would give us a good estimate of the typical run time; we could achieve this by using random integer function to index random words throughout the list.