Session for lecture 10: Understanding Program Efficiency: 1

```
#    *** Just an Example ***
import time
import random
def generate_list(length):
    L = []
    for i in range(length):
        L.append(random.randint(1,1000))
    return L

def linear_search(L, e):
    found = False
    for i in range(len(L)):
        if e == L[i]:
            found = True
    return found

testList = [1, 3, 4, 5, 9, 18, 27]
# testList = generate_list(1000000)
# testList = generate_list(1000)

starting_time = time.clock()
print(linear_search(testList,50))
ending_time = time.clock()
total_time = ending_time - starting_time
print("Execution time is ", total_time)

# write your own tes code for this function,
# try to use different size of lists, and notice the execution time for each one.
# Evaluate efficiency of programs by:
#     1- measure with a timer
#     2- count the operations
#     3- What is the order of growth?
##########################################################################################
# Q.1 [1 mk]
def search(L, e):
    for i in range(len(L)):
        if L[i] == e:
            return True
        if L[i] > e:
            return False
    return False

# write your own tes code for this function,
# try to use different size of lists, and notice the execution time for each one.
# Evaluate efficiency of programs by:
# 1- measure with a timer
# 2- count the operations
# 3- What is the order of growth?
##########################################################################################
```

 # Q.2 [2 mks]

```
def isSubset(L1, L2):
    for e1 in L1:
        matched = False
        for e2 in L2:
            if e1 == e2:
                matched = True
                break
        if not matched:
            return False
    return True

testSet = [1, 2, 3, 4, 5]
testSet1 = [1, 5, 3]
testSet2 = [1, 6]
```

 # write your own tes code for this function,
 # try to use different size of lists, and notice the execution time for each one.
 # Evaluate efficiency of programs by:
 # 1- measure with a timer
 # 2- What is the order of growth?

##

 # Q.3 [2 mks]

```
def intersect(L1, L2):
    tmp = []
    for e1 in L1:
        for e2 in L2:
            if e1 == e2:
                tmp.append(e1)
    res = []
    for e in tmp:
        if not(e in res):
            res.append(e)
    return res
```

 # write your own tes code for this function,
 # try to use different size of lists, and notice the execution time for each one.
 # Evaluate efficiency of programs by:
 # 1- measure with a timer
 # 2- What is the order of growth?

##

 # Q.4 [Bonus]

Write some code to plot the runtime of the function in Q1 above. To do this, generate lists of sizes 1,10,100,1000…100000000 and then measure the runtime of the function for each size. A helper function which takes two lists – one of sizes, the other of times – has been provided for your convenience below:

```
def plot(sizes, times):
    import matplotlib.pyplot as plt

    plt.figure()
    plt.plot(sizes, times)
    plt.xlabel("Sizes")
    plt.ylabel("Times(s)")
    plt.grid(True, which="both")
    plt.show()
```

Hint: Since we're interested in worst-case complexity, be sure to ensure that the entire list is used during the operation. To do this, you will need to modify the chosen question's code in some way to ensure this.

Questions:
-What do you expect the graph to look like?Why?
-Does the graph match your expectations?
-If you were to do this for Q2 and Q3, what do you think the graph would look like instead?

Good hunting