```
#James Gardner
#Will Schwarzer
import matplotlib.pyplot as plt
import numpy as np
import math
import time
import random

# Sorts the list of numbers in place.
# Input: List of numbers.
# Output: The same list of numbers, sorted in place.
def bubbleSortInPlace(l):
    for nForSorting0ToN in range(len(l) - 1, 0, -1):
        for i in range(nForSorting0ToN):
            if l[i] > l[i + 1]:
                temp = l[i + 1]
                l[i + 1] = l[i]
                l[i] = temp
    return l

# Sorts the list of numbers, returning an independent copy.
# Input: List of numbers.
# Output: List of numbers.
def bubbleSort(l):
    # Python trick: l[:] is a _copy_ of all of l.
    return bubbleSortInPlace(l[:])

# Sorts part of the list of numbers (from index a onward).
# Input: List of numbers. Integer index into that list.
# Output: The same list of numbers, sorted in place.
def selectionSortInPlace(l, a):
    while len(l) - a >= 2:
        # Scan to find smallest element.
        least = a
        for i in range(a + 1, len(l)):
            if l[i] < l[least]:
                least = i
        # Swap the list elements in positions a and least.
        temp = l[a]
        l[a] = l[least]
        l[least] = temp
        # The ath element is correct; sort the rest of the list.
        a += 1
    return l

# Sorts the list of numbers, returning an independent copy.
# Input: List of numbers.
# Output: List of numbers.
def selectionSort(l):
    return selectionSortInPlace(l[:], 0)

# Sorts the list of numbers, returning an independent copy.
# Input: List of numbers.
# Output: List of numbers.
def mergeSort(l):
    if len(l) <= 1:
        # The list is so short that it is already sorted.
        return l
    else:
        # Sort the front and back halves of the list.
        middle = len(l) // 2
        front = mergeSort(l[:middle])
        back = mergeSort(l[middle:])
        # Merge the two halves into a single sorted list.
        result = []
```

```
        i = 0
        j = 0
        while i < len(front) and j < len(back):
            if front[i] < back[j]:
                result.append(front[i])
                i += 1
            else:
                result.append(back[j])
                j += 1
        # Include any remaining elements from the lists.
        while i < len(front):
            result.append(front[i])
            i += 1
        while j < len(back):
            result.append(back[j])
            j += 1
        return result

def randomList(cap):
    unmixed = list(range(cap))
    mixed = []
    while len(unmixed) > 0:
        j = random.randint(0, len(unmixed) - 1)
        mixed += [unmixed[j]]
        unmixed = unmixed[:j] + unmixed[(j + 1):]
    return mixed

def testBubble(numTrials, cap):
    l =[]
    for i in range(numTrials):
        l+=[randomList(cap)]
    start = time.clock()
    for j in range(len(l)):
        bubbleSort(l[j])
        finish = time.clock()
    return (finish - start)

def testSelect(numTrials, cap): #same concepts as gcd
    l =[]
    for i in range(numTrials):
        l+=[randomList(cap)]
    start = time.clock()
    for j in range(len(l)):
        selectionSort(l[j])
        finish = time.clock()
    return (finish - start)

def testMerge(numTrials, cap): #same concepts as gcd
    l =[]
    for i in range(numTrials):
        l+=[randomList(cap)]
    start = time.clock()
    for j in range(len(l)):
        mergeSort(l[j])
        finish = time.clock()
    return (finish - start)




def sortPlot(bit): #Will Schwarzer
    x = np.arange(1,bit+1, 1)
    y1 = np.array([testBubble(trials,x)/trials for x in range(1,bit+1)])
    y2 = np.array([testMerge(trials,x)/trials for x in range(1,bit+1)])
    y3 = np.array([testSelect(trials,x)/trials for x in range(1,bit+1)])
```

```
    plt.plot(x,y1,x,y2,x,y3)
    plt.show()

#Example test runs
trials = 40
listlen = 200
sortPlot(listlen)
'''
print(testBubble(trials,listlen)/trials)
print(testMerge(trials,listlen)/trials)
print(testSelect(trials,listlen)/trials)'''


'''
def main():
    l = [5, 2, 9, 1, 7, 3, 4, 6, 0, 8]
    print("The unsorted list is", l, ".")
    print("The result of bubbleSort is", bubbleSort(l), ".")
    print("The result of selectionSort is", selectionSort(l), ".")
    print("The result of mergeSort is", mergeSort(l), ".")
    print("The original list is still intact:", l, ".")

if __name__ == "__main__":
    main()'''
```