```
#James Gardner
#Will Schwarzer
import matplotlib.pyplot as plt
import numpy as np
import math
import time
time.time() / (60 * 60 * 24 * 365.25)

start = time.clock()
# Put some commands here, that you'd like to time.
finish = time.clock()
#print(finish - start)

import random
#random.randint(13, 72)
def randomIntOfBits(numBits):
    return random.randint(2 ** (numBits - 1), 2 ** numBits - 1)

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 gcd(a, b):
    if min(a, b) == 0:
        return max(a, b)
    else:
        for d in range(min(a, b), 0, -1):
            if a % d == 0 and b % d == 0:
                return d

def euc(a, b):
    while b != 0:
        rem = a % b
        a = b
        b = rem
    return a

def numTrials(a, numBits):
    a = []
    while len(a) <= len(numBits):
        i = random.randint(0, len(numBits) -1)
        a += [a[i]]
    return a

def gcdTest(numTrials, numBits):
    a =[] #Empty List A
    b =[] #Empty List B
    for i in range(numTrials):
        a+=[randomIntOfBits(numBits)] #Generates random list A from number of bits
        b+=[randomIntOfBits(numBits)] #Generates random list B from number of bits
    start = time.clock() #start time outside of for loop
    for j in range(len(a)):
        gcd(a[i],b[i]) #computes gcd
    finish = time.clock() #finish time after computation
    return (finish - start) #return the calculation time

def eucTest(numTrials, numBits):
    a =[] #Empty List A
    b =[] #Empty List B
    for i in range(numTrials):
        a+=[randomIntOfBits(numBits)] #Generates random list A from number of bits
        b+=[randomIntOfBits(numBits)] #Generates random list B from number of bits
    start = time.clock() #start time outside of for loop
    for j in range(len(a)):
        euc(a[i],b[i]) #computes gcd with euc
    finish = time.clock() #finish time after computation
    return (finish - start) #return the calculation time


def gcdPlot(bit): #Will Schwarzer
    x = np.arange(1,bit+1, 1)
    y1 = np.array([gcdTest(trials,x)/trials for x in range(1,bit+1)])
    y2 = np.array([eucTest(trials,x)/trials for x in range(1,bit+1)])
    plt.plot(x,y1,x,y2)
    plt.show()


#Example test runs
trials = 100
bits = 10
gcdPlot(bits)
'''
print(gcdTest(trials,bits)/trials)
print(eucTest(trials,bits)/trials)'''
```