Problem Statement : Given a number , identifying if a given number is a Carmichael number

Anirudh Sathish

```
# Defining Packages
# Used for accesign the sqrt function
import math

class carmiachelNumbers(object):

    # Initilizing values

    def __init__(self,inputNumber):

        self.NumberToBeTested = inputNumber

    def isOdd(self):

        #Function checks if given number is odd

        if(self.NumberToBeTested %2) == 0:

            return False

        else:

            return True

    def isCompositeMethod1(self):

        #Function to check if the number is composite or prime

        rootOfNumber = math.sqrt(self.NumberToBeTested)

        i = 2

        while( i <= rootOfNumber):

            if(self.NumberToBeTested%i == 0):

                return True

            i+=1
```

```
        return False
    def FindGCD(self,numberForTest ,iteratingNumber):
        #Finds gcd of two numbers
        if(iteratingNumber == 0):
            return numberForTest
        else:
            return
self.FindGCD(iteratingNumber,numberForTest%iteratingNumber)
    def appropriateGCD(self,HCF_test):
        #Function checks if the HCF of two numbers is 1
        #Which is a criterion in fermat's test
        if(HCF_test == 1):
            return True
        return False
    def FermatTestResult(self , iteratingNumber , raisedPower , modvalue):
        #Performing Fermat's test
        if(raisedPower == 0):
            return 1
        #Keep on reducing the number
        RHS =
self.FermatTestResult(iteratingNumber,raisedPower//2,modvalue)%modvalue
        RHS = (RHS*RHS) % modvalue
        if(raisedPower %2 == 1):
            RHS = (RHS*iteratingNumber) % modvalue
        return RHS
```

```
    def isPassedFermatTest(self,iteratingNumber , raisedPower ,modvalue):
        RHS = self.FermatTestResult(iteratingNumber,raisedPower,modvalue)
        if(RHS ==1):
            return True
        return False
    def FinalTest(self):
        number = self.NumberToBeTested
        for i in range(2,number):
            HCF = self.FindGCD(number,i)
            TheResult = self.appropriateGCD(HCF)
            fermatResult = self.isPassedFermatTest(i,number-1, number)
            if(TheResult == True):
                if(fermatResult == False):
                    return False
        return True
    #Printing on basis ->
    def Print(self,status):
        if(status == 1):
            print("The number ",self.NumberToBeTested," is a carmiachel
number ")
        elif(status == 0):
            print("The number ",self.NumberToBeTested," is not a carmiachel
number")
#Running the body
```

```
#Taking input

inputNumber = int(input("Enter the number "))



#Initalizing the object

numberSystem = carmiachelNumbers(inputNumber)



level1Test = numberSystem.isOdd()

if(level1Test == False):

    numberSystem.Print(0)

else:

    level2Test = numberSystem.isCompositeMethod1()

    if(level2Test == False):

        numberSystem.Print(0)

    else:

        final = numberSystem.FinalTest()

        if(final == False):

            numberSystem.Print(0)

        else:

            numberSystem.Print(1)
```

The estimated time complexity of the following code is *O(sqrt(N))* , where N is the number which is being tested if Carmichael or not

The Algorithm goes like follows:

1. The number is checked if odd : *if even , the number is not Carmichael , so that is ignored . If odd it proceeds further
2. The number is checked if composite : *If number is prime , it can be ignored since it will anyway satisfy Fermat's test and hence cannot be Carmichael . If number is composite , we take it to the next stage
3. Over here we check for Fermat's test: *We check for each a, does the number satisfy the required property

*The complexity of the algorithm includes the complexity of each of the process :

1. *O(1) for Odd testing
2. *O(sqrt(N) for testing if number is composite
3. *O(sqrt(N) for finding HCF
4. *2log(N) for testing the rest part of Fermat's test

Since we are looking for an upper bound . The Worst case running time is going to be *O(sqrt(N))
Here N is the number itself

The output for certain values

```
In [1]: runcell(0, 'C:/Users/aniru/Project_S/
CarmiachelTestBrute.py')

Enter the number 561
The number  561  is a carmiachel number

In [2]: runcell(0, 'C:/Users/aniru/Project_S/
CarmiachelTestBrute.py')

Enter the number 825265
The number  825265  is a carmiachel number

In [3]: runcell(0, 'C:/Users/aniru/Project_S/
CarmiachelTestBrute.py')

Enter the number 10091
The number  10091  is not a carmiachel number
```

**Note

1. *The algorithm presented here is a brute force scenario
2. *This can be further optimized , by using the quadratic sieve method , which could bring down the asymptomatic complexity to (d)^1/3 , where d is the number of digits
3. *The above method is planned to be implemented in the further iterations
4. *Current algorithm , works for all inputs . But it is able to calculate till 8 digit numbers within reasonable time