Cryptography and Network Security Lab

Assignment 9
Implementation and Understanding of Chinese Remainder Theorem

2019BTECS00058
Devang K

Batch: B2

Title: Implementation and Understanding of Chinese Remainder Theorem (CRT)

Aim: To Study, Implement and Demonstrate the Chinese Remainder Theorem (CRT)

Theory:

In mathematics, the Chinese remainder theorem states that if one knows the remainders of the Euclidean division of an integer n by several integers, then one can determine uniquely the remainder of the division of n by the product of these integers, under the condition that the divisors are pairwise coprime (no two divisors share a common factor other than 1).

For example, if we know that the remainder of n divided by 3 is 2, the remainder of n divided by 5 is 3, and the remainder of n divided by 7 is 2, then without knowing the value of n, we can determine that the remainder of n divided by 105 (the product of 3, 5, and 7) is 23. Importantly, this tells us that if n is a natural number less than 105, then 23 is the only possible value of n.

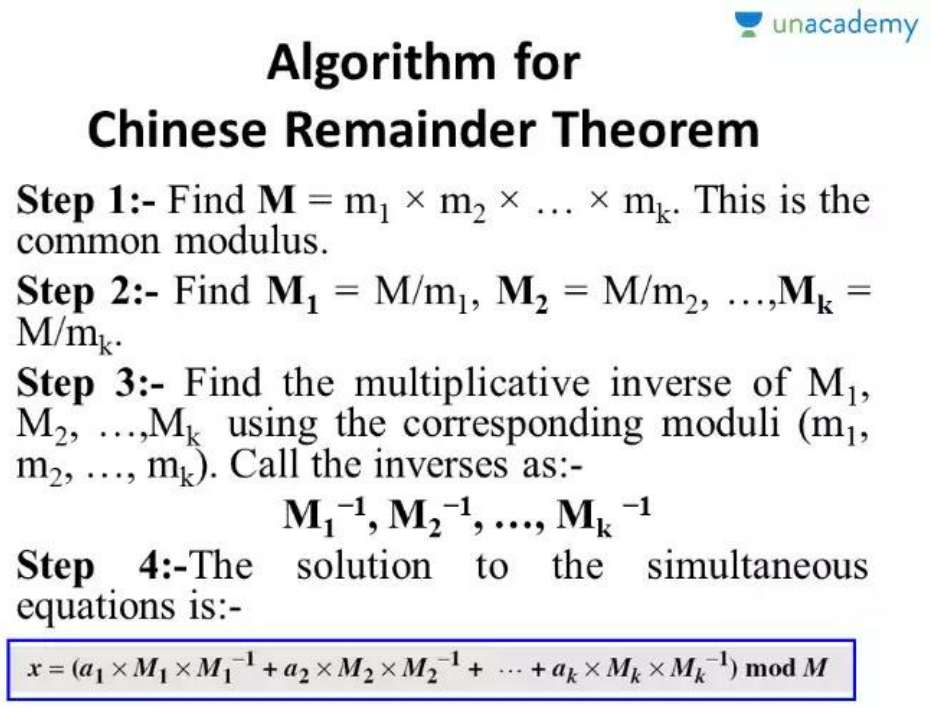
The earliest known statement of the theorem is by the Chinese mathematician Sun-tzu in the Sun-tzu Suan-ching in the 3rd century CE.

The Chinese remainder theorem is widely used for computing with large integers, as it allows replacing a computation for which one knows a bound on the size of the result by several similar computations on small integers.

CRT typically gives the smallest possible solution, but it's not the only solution. There are infinite number of solutions possible for CRT equations.

We make use of the Extended Euclid's algorithm of previous practical for the computation of CRT.

The algorithm to compute CRT is as follows:



We now therefore, code for the algorithm.

Code:

```
# We use the Extended Euclidean Function from last assignment
def computeInverse(a, b):
    # ensure that a is always the larger number
    if b > a:
        temp = a
        a = b
        b = temp
    amt = 1
    while(True):
        if((a*amt) % b == 1):
            return amt
        amt += 1
```

```
def computeCRT(lst):
    M = 1
    for i in range(len(lst)):
        M *= lst[i][1]
    valuesOfMs = []
    for i in range(len(lst)):
        print(lst[i][1])
        print(M//lst[i][1])
        print(computeInverse(lst[i][1], M//lst[i][1]))
        print()
        valuesOfMs.append([M//lst[i][1], computeInverse(lst[i][1],
M//lst[i][1])])
    print("\nValue of M: "+str(M)+"\n")
    for i in range(len(lst)):
        print("a"+str(i+1)+" = "+str(lst[i][0])+" M"+str(i+1)+" =
"+str(valuesOfMs[i][0])+", M"+str(i+1)+" -1 = "+str(valuesOfMs[i][1]))
    theModularValue = 0
    for i in range(len(lst)):
        theModularValue += (lst[i][0] * valuesOfMs[i][0] * valuesOfMs[i][1])
        theModularValue %= M
    return theModularValue, M

print("-- Chinese Remainder Theorem --\n")

print("Enter number of equations: ", end='')
n = int(input())

equations = []
for i in range(n):
    print("Enter number: ", end='')
    num = int(input())
    print("Enter modulo: ", end='')
    mod = int(input())
    print()
    equations.append([num, mod])
ans, M = computeCRT(equations)
print("\nOur M is "+str(M))
print("\nThe first solution is: "+str(ans))

print("\nTherefore our solutions are "+str(ans)+", "+str(ans+M)+",
"+str(ans+(2*M))+" and so on...")

# 2 3
# 3 5
# 2 7
```

We now illustrate with examples:

We illustrate first for 3 equations:

2 3

3 5

2 7

We get:

```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\CRT> py .\script.py
-- Chinese Remainder Theorem --

Enter number of equations: 3
Enter number: 2
Enter modulo: 3

Enter number: 3
Enter modulo: 5

Enter number: 2
Enter modulo: 7
```

```
Value of M: 105

a1 = 2 M1 = 35, M1 -1 = 2
a2 = 3 M2 = 21, M2 -1 = 1
a3 = 2 M3 = 15, M3 -1 = 1

Our M is 105

The first solution is: 23

Therefore our solutions are 23, 128, 233 and so on...
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\CRT>
```

All solutions of the form 23 + n*105 are valid.

Let’s take another example:

We take 2 equations:

3 5

6 8

```
PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\CRT> py .\script.py
-- Chinese Remainder Theorem --

Enter number of equations: 2
Enter number: 3
Enter modulo: 5

Enter number: 6
Enter modulo: 8
```

Solution:

```
 Value of M: 40

 a1 = 3 M1 = 8, M1 -1 = 2
 a2 = 6 M2 = 5, M2 -1 = 2

 Our M is 40

 The first solution is: 28

 Therefore our solutions are 28, 68, 108 and so on...
 PS C:\Users\marcus\Desktop\College\CNS-Lab-Archives\CRT>
```

We get solutions 28, 68, 108, 148 and so on.

Thus, we demonstrated the working of the code with examples.

Conclusion:

Thus, the Chinese Remainder Theorem algorithm was studied and demonstrated with the code.