Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

Chinese Remainder Theorem

Theory:

The Chinese Remainder Theorem (CRT) is a mathematical concept that provides a method for solving a system of simultaneous linear congruences. In essence, it allows you to find a unique solution to a set of modular equations by combining solutions from simpler, individual modular equations.

Here's a brief theory of CRT:

1. System of Congruences:

CRT is used to solve a system of congruences of the form:

```
x ≡ a_1 (mod n_1)

x ≡ a_2 (mod n_2)

...

x ≡ a_k (mod n_k)
```

2. Coprime Moduli:

For CRT to work, the moduli (n_1, n_2, ..., n_k) should be pairwise coprime, meaning that their greatest common divisors (GCD) are all 1.

3. Unique Solution:

CRT guarantees a unique solution for the value of "x" within a specific range.

4. Solution Calculation:

- Calculate the product of all moduli: N = n_1 * n_2 * ... * n_k.
- For each congruence, compute N_i = N / n_i.
- Find the modular inverses (x_i) of each N_i modulo n_i.
- The solution "x" is given by:

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
x = a_1 * N_1 * x_1 + a_2 * N_2 * x_2 + ... + a_k * N_k * x_k (mod N)
```

5. Applications:

CRT has applications in number theory, modular arithmetic, cryptography, and computer science. It is used in algorithms for efficient arithmetic in finite fields and modular integer operations.

CRT is a powerful tool for solving systems of modular equations, and it plays a crucial role in various computational and cryptographic applications.

Code:

```python
def extended_gcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = extended_gcd(b % a, a)
        return (g, y - (b // a) * x, x)

def mod_inverse(a, m):
    g, x, _ = extended_gcd(a, m)
    if g != 1:
        raise ValueError("The modular inverse does not exist.")
    return x % m

def chinese_remainder_theorem(n, a):
    N = 1
    N_i = []
    x_i = []
    b_i = []

    for ni in n:
        N *= ni

    for ni in n:
        N_i.append(N // ni)
        x_i.append(mod_inverse(N_i[-1], ni))

    for i in range(len(n)):
        b_i.append(a[i] * N_i[i] * x_i[i])

    print("Intermediate Steps:")
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```python
    for i in range(len(n)):
        print(f"N_{i} = {N_i[i]}, x_{i} = {x_i[i]}, b_{i} = {b_i[i]}")

    result = sum(b_i) % N

    return result

# Input from the user
n = []
a = []

num_congruences = int(input("Enter the number of congruences: "))

for i in range(num_congruences):
    n_i = int(input(f"Enter the modulus (n_{i}): "))
    a_i = int(input(f"Enter the remainder (a_{i}): "))
    n.append(n_i)
    a.append(a_i)

result = chinese_remainder_theorem(n, a)

print("\nSystem of Congruences:")
for i in range(len(n)):
    print(f"x = {a[i]} (mod {n[i]})")

print(f"The solution is x = {result}")
```

Screenshot:

```
PS F:\D drive\0Walchand_Sem7\CNS lab> python -u "f:\D drive\0Walchand_Sem7\CNS lab\CRT\CRT.py"
Enter the number of congruences: 3
Enter the modulus (n_0): 3
Enter the remainder (a_0): 2
Enter the modulus (n_1): 4
Enter the remainder (a_1): 3
Enter the modulus (n_2): 5
Enter the remainder (a_2): 1
Intermediate Steps:
N_0 = 20, x_0 = 2, b_0 = 80
N_1 = 15, x_1 = 3, b_1 = 135
N_2 = 12, x_2 = 3, b_2 = 36

System of Congruences:
x = 2 (mod 3)
x = 3 (mod 4)
x = 1 (mod 5)
The solution is x = 11
PS F:\D drive\0Walchand_Sem7\CNS lab>
```