Discrete Task 4 Anaylsis

Name: Kareem Gaber Abu Hashem El Halaby

Code: 2101545

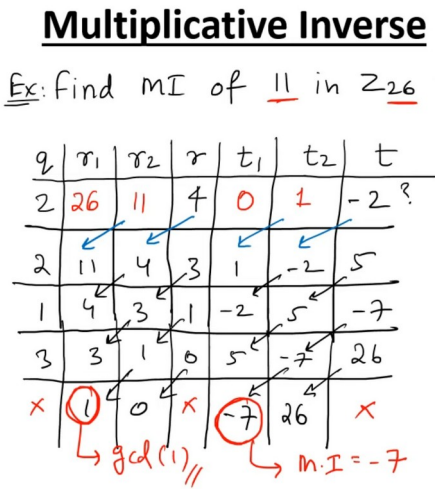
Group: 28

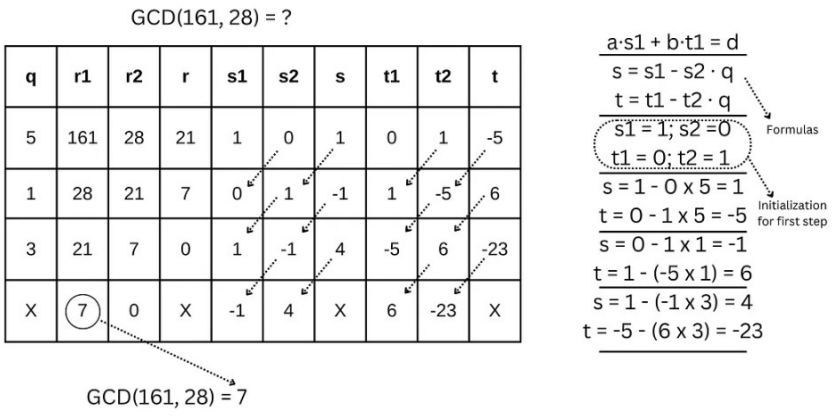
Affine Decryption Rule:

D = a^-1 (c - b) mod m
c: position of ciphered letter in alphabet
a: 1st key
b: 2nd key
m: number of letters in alphabet
D: Deciphered letter

Mod Inverse

Calculated with Extended Euclidean algorithm:





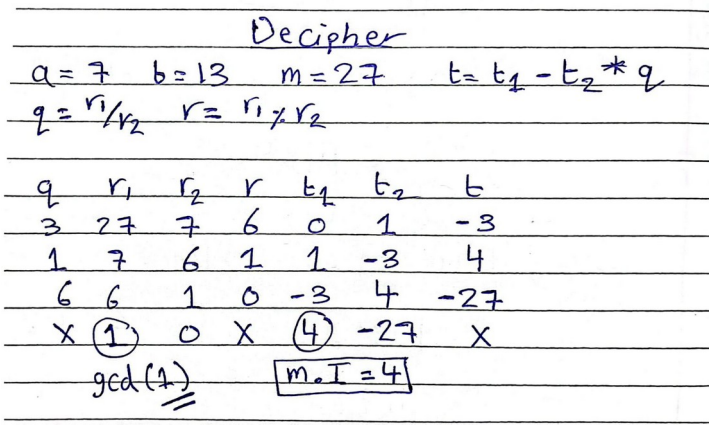
Analytical Calculation for my case:

a=7 b=13 m=27

language: English

Modulus Inverse = 4

gcd(27 , 7) = 1



Implementation Steps:

- here m and a will be copied to r1 and r2
- initialization of t1 and t1: t1=0 , t2=1
- quotient q=r1/r2 and remainder r=r1%r2 calculated
- calculate t= t1 – t2 * q
- shift r2 to r1 , remainder to r2 and t2 to t1
- Keep Looping until r2 reaches 0

At the end:

- r1 will be gcd , it must be 1 for multiplicative inverse to be present , the 2 numbers must be coprime with each other
- (gcd(a,m) must be 1)
- t1 will be the multiplicative inverse I am looking for but the number might be negative and it should be between 0 and m-1
- We loop and add m till it is inside range.
- **Last block I check if gcd (r1) equals 1 if not then no multiplicative inverse so returns -1**

```
int modInverse(int a,int m){
    int t1=0,t2=1,t,r1=m,r2=a;
    while(r2!=0){ // applying extended euclidean algorithm to get t1 wh
        int q,r;
        q=r1/r2;
        r=r1%r2;
        t=t1-t2*q;
        r1=r2;
        r2=r;
        t1=t2;
        t2=t;
    }
    if(r1==1){
        while(t1<0){ // mod inverse must be bewteen 0 & m-1
            t1=t1+m;
    }
        return t1%m; // returning mod inverse of a and m with gcd of 1
    }
    else return -1; // not coprime so no mod inverse
}
```

Main function

Input Prompts: Message to be Deciphered , a and b
a=7 , b=13 , message = “ROVKMWUKKTHUMVKMRJQMKUGDUR”

Language: English Alphabet with space at position 0

m: 27

```cpp
int main() {
    string cipherText,alphabet;
    int a, b;
    cout << "Enter the affine ciphered message: ";
    getline(cin, cipherText);
    cout << "Enter a: ";
    cin >> a;
    cout << "Enter b: ";
    cin >> b;
    alphabet=" ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //  SPACE is added
    int m = alphabet.size();

    // Decrypt the message
    string decryptedMessage = affineDecrypt(cipherText, a, b, m,alphabet);

    if (!decryptedMessage.empty()) {
        cout << "The decrypted message is: " << decryptedMessage << endl;
    }

    return 0;
}
```

Decipher Function

1) Modulus inverse of a is calculated using function described above
2) Checks if valid modulus inverse or not
3) Loops over each letter of ciphered message
4) Calculates Deciphered letter offset in alphabet array using :

   a_inv * (position of letter in alphabet – b) % m

5) Position calculated can be negative so we have to make it between 0 and m-1 by adding m until value in range
6) Access position of deciphered letter in our alphabet array (alphabet[] array of 27 letter)
7) Add letter to decrypted message

```
// Function to decrypt the Affine ciphered message
string affineDecrypt(string cipherText, int a, int b, int m,string alphabet) {
    string decryptedText = "";
    int a_inv = modInverse(a, m);

    if (a_inv == -1) {
        cout << "Modular inverse of 'a' does not exist!" << endl;
        return "";
    }

    for (char c : cipherText) {
        int L;
        L=(a_inv * (search(alphabet,c)-b))%m;
        while(L<0){
                L=L+m;
            }
        decryptedText += alphabet[L];
    }
    return decryptedText;
}
```

Example for 1st letter in message:

1st letter of ciphered message : R
4 (18 – 13) % 27 = 20
T=20

Output:

```
Enter the affine ciphered message: ROVKMWUKKTHUMVKMRJQMKUGDUR
Enter a: 7
Enter b: 13
The decrypted message is: THIS MESSAGE IS TOP SECRET
```