Final Year B. Tech, Sem VII 2022-23
PRN – 2020BTECS00211
Name – Aashita Narendra Gupta
Cryptography And Network Security Lab
Batch: B4
Practical No – 8

Title: Implementation of Eucledian Algorithm and Extended Eucledian Algorithm.

Theory:

1. **Euclidian Algorithm:**
   Euclid's algorithm, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder. It is named after the ancient Greek mathematician Euclid, who first described it in his Elements (c. 300 BC). It is an example of an algorithm, a step-by-step procedure for performing a calculation according to well-defined rules, and is one of the oldest algorithms in common use. It can be used to reduce fractions to their simplest form, and is a part of many other number-theoretic and cryptographic calculations.

   Input: $a, b \in R$.

   Output: $g \in R$ a gcd of a and b.

   $r_0 := a$
   $r_1 := b$
   $i := 2$
   while $r_{i-1} \neq 0$ **repeat**
   $\quad r_i := r_{i-2} \text{ rem } r_{i-1}$
   $\quad i := i + 1$
   return r_{i-2}

2. **Extended Eucledian Algorithm:**
   The Extended Euclidean algorithm is arguably one of the oldest and most widely known algorithms. It is a method of computing the greatest common divisor (GCD) of two integers aa and bb. It allows computers to do a variety of simple number-theoretic tasks, and also serves as a foundation for more complicated algorithms in number theory.

Algorithm 3.4 Extended Euclidean algorithm

Input: Integers a and b

Output: Integers x, y, and d, where $d = \gcd(a, b) = ax + by$

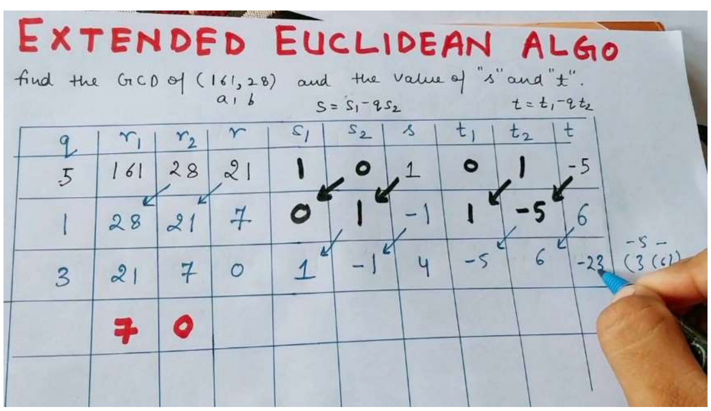
- **Set** $d_0 = a$ **Set** $x_0 = 1$ **Set** $y_0 = 0$
- **Set** $d_1 = b$ **Set** $x_1 = 0$ **Set** $y_1 = 1$
- **While** $d_1 \neq 0$ **Do**
  - **Set** $q = \lfloor d_0/d_1 \rfloor$
  - **Set** $d_2 = d_1$ **Set** $x_2 = x_1$ **Set** $y_2 = y_1$
  - **Set** $d_1 = d_0 - qd_1$
  - **Set** $x_1 = x_0 - qx_1$
  - **Set** $y_1 = y_0 - qy_1$
  - **Set** $d_0 = d_2$ **Set** $x_0 = x_2$ **Set** $y_0 = y_2$
- **End While**
- **Return** $[d, x, y] = [d_0, x_0, y_0]$

Example:

1. **Eucledian:**

Step	a	b	|a - b|
1	527	221	306
2	306	221	85
3	85	221	136
4	85	136	51
5	85	51	34
6	34	51	17
7	34	17	17
8	17	17	0

2. **Extended Eucledian:**



Code Snapshots:

1. **Eucledian:**

```cpp
#include <bits/stdc++.h>
using namespace std;

int findGcd(int r1, int r2)
{
    if (r2 == 0)
    {
        return r1;
    }

    int q = r1 / r2;
    int r = r1 % r2;

    cout<<"q "<<"r1 "<<"r2 "<<"r "<<endl;
    cout << q << " " << r1 << " " << r2 << " " << r << " "<< endl;

    return findGcd(r2, r);
}

int main()
{

    int num1, num2;
    cout << "Enter 2 numbers to find GCD" << endl;
    cin >> num1 >> num2;

    int gcd = findGcd(num1, num2);
    cout << "GCD is " << gcd << endl;


    return 0;
}
```

Output Snapshots:

```
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\CNS\Programs> cd "c:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\CNS\Programs\" ; if ($?) { g++ Eucledian.cpp -o Eucledian } ; if ($?) { .\Eucledian }
Enter 2 numbers to find GCD
120 7
q r1 r2 r
17 120 7 1
q r1 r2 r
7 7 1 0
GCD is 1
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\CNS\Programs>
```

Code Snapshots:

2. Extended Eucledian:

```
#include <bits/stdc++.h>
using namespace std;

int ansS, ansT;

int findGcdExtended(int r1, int r2, int s1, int s2, int t1, int t2)
{
    // Base Case
    if (r2 == 0)
    {
        ansS = s1;
        ansT = t1;
        return r1;
    }

    int q = r1 / r2;
    int r = r1 % r2;

    int s = s1 - q * s2;
    int t = t1 - q * t2;

    cout<<"q "<<"r1 "<<"r2 "<<"r "<<"s1 "<<"s2 "<<"s "<<"t1 "<<"t2 "<<"t
"<<endl;
    cout << q << " " << r1 << " " << r2 << " " << r << " " << s1 << " " << s2
<< " " << s << " " << t1 << " " << t2 << " " << t << endl;

    return findGcdExtended(r2, r, s2, s, t2, t);
}

int main()
{

    int num1, num2, s, t;
    cout << "Enter 2 numbers to find GCD" << endl;
    cin >> num1 >> num2;

    int gcd = findGcdExtended(num1, num2, 1, 0, 0, 1);
    cout << "GCD is " << gcd << endl;
    cout << "s = "<<ansS << ",  " << "t = "<< ansT << endl;

    return 0;
}
```

Output Snapshots:

```
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\CNS\Programs> cd "c:\Users\Ashitra\OneDrive\De
sktop\7th sem\Practicals\CNS\Programs\" ; if ($?) { g++ ExtendEucledian.cpp -o ExtendEucledian } ; if
($?) { .\ExtendEucledian }
Enter 2 numbers to find GCD
120 7
q r1 r2 r s1 s2 s t1 t2 t
17 120 7 1 1 0 1 0 1 -17
q r1 r2 r s1 s2 s t1 t2 t
7 7 1 0 0 1 -7 1 -17 120
GCD is 1
s = 1,  t = -17
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\CNS\Programs>
```

Conclusion:

1. The Euclidean algorithm helps us in finding the greatest common divisor of two non-negative integers.
2. The Extended Euclidean algorithm builds on top of the basic Euclidean algorithm. It can solve linear diophantine equations of the form: ax + by = c, where c is divisible by the greatest common divisor of a and b.