9. Explain Euclidean algorithm for finding the greatest common divisor.

Ans.: The *Euclidean algorithm* (also called *Euclid's algorithm*) is an efficient algorithm for finding the *greatest common divisor* (*GCD*) of two positive integers. This algorithm was invented by the Greek mathematician Euclid and is hence named after him. Given two positive integers `x` and `y`, then another positive number (say, `a`) is called the gcd of `x` and `y` if and only if the following conditions are satisfied:

(i) `a` divides both `x` and `y`.

(ii) Any other common divisor of `x` and `y` also divides `a`.

In other words, `gcd(x,y)=a` if `a` is the largest integer that divides both `x` and `y`.

Euclidean's algorithm computes the gcd of two positive integers, `x` and `y`, based on the following facts:

(i) `gcd(x,0)=x`, that is, if the second integer is zero, then the gcd is the first integer.

(ii) `gcd(x,y)=gcd(y,r)`, where `r` is the remainder obtained on dividing `x` by `y`.

Algorithm

The following are the steps to find the gcd of two positive integers `x` and `y`, where `x>y>0` using Euclidean's algorithm, are as follows:

```
1. a:=x
   2. b:=y
   3. while (b>0)
      {
       q:=a/b
       r:=a-q*b
       a:=b
       b:=r
      }
   4. gcd(x,y):=a
```

In this algorithm, we have used two variables `a` and `b` to hold the remainders produced during the reduction process. To start with, variables `a` and `b` are initialized with `x` and `y`, respectively. During each step in the reduction process, we calculate the remainder of `a` divided by `b` and then store it into the variable `r`. Then, `a` and `b` are replaced with `b` and `r`, respectively. This process is continued until the value of `b` becomes zero. Eventually, we get the `gcd(x, y)` as `a`.

16. Find the GCD of 2740 and 1760 using the Euclidean algorithm.

Ans.: Using the Euclidean algorithm as explained in Question 9, we have `x` = 2740 and `y` = 1760.

Now, initializing `a = x` and `b = y`, we get `a` = 2740 and `b` = 1760. As `b` > 0, we move to the first iteration of the while loop.

Algorithm

First iteration

`q`=2740/1760=1

`r`=2740-1*1760=980

`a`=1760

`b`=980

As 980 > 0, we move to the next iteration.

Second iteration

q=1760/980=1

r=1760-1*980=780

a=980

b=780

As 780 > 0, we move to the next iteration.

Third iteration

q=980/780=1

r=980-1*780=200

a=780

b=200

As 200 > 0, we move to the next iteration.

Fourth iteration

q=780/200=3

r=780-3*200=180

a=200

b=180

As 180 > 0, we move to the next iteration.

Fifth iteration

q=200/180=1

r=200-1*180=20

a=180

b=20

As 20 > 0, we move to the next iteration.

Sixth iteration

q=180/20=9

r=180-9*20=0

a=20

b=0

As the value of b has become zero, the while loop terminates.

Thus, gcd(x, y)=a

⇒gcd(2740, 1760)=20

12. Describe the extended Euclidean algorithm to find the multiplicative inverse.

Ans.: The extended Euclidean algorithm is an extension to the Euclidean algorithm. Besides finding the gcd of two positive integers `x` and `y`, it simultaneously finds the multiplicative inverses `a` and `b` such that:

```
m*x+n*y=gcd(x,y)
```

where `m` is the multiplicative inverse of `x mod y` and `n` is the multiplicative inverse of `y mod x`.

Algorithm

The following are the steps involved in the extended Euclidean algorithm to find the gcd of two positive integers along with the multiplicative inverses are as follows:

```
1. a:=x
   2. b:=y
   3. c:=1
   4. d:=0
   5. e:=0
   6. f:=1
   7. while (b>0)
      {
        q:=a/b
        r:=a-q*b
        a:=b
        b:=r

        m:=c-q*d
        c:=d
        d:=m

        n:=e-q*f
        e:=f
        f:=n
      }
   8. gcd(x,y):=a
   9. m:=c
  10. n:=e
```

Similar to the Euclidean algorithm, the extended Euclidean algorithm also uses the reduction process to find the gcd and multiplicative inverses. It uses three sets of variables, (`a,b`), (`c,d`), and (`e,f`) and during each step of the reduction process, three sets of calculations are made, one per each set of variables. To start with, the variables `a, b, c, d, e,` and `f` are initialized with `x, y, 1, 0, 0,` and `1`, respectively. In the while loop, variables `q` and `r` are used to hold the quotient and the remainder of `a` divided by `b`, respectively. Then, variables `a` and `b` are updated in a similar manner as in the Euclidean algorithm. The set of variables (`c,d`) and (`e,f`) are also updated on the basis of `q`'s value. This process continues until the value of `b` becomes zero. Finally, we obtain the `gcd(x,y)` as `a` as well as the values of `m` and `n`.

17. Find the greatest common divisor of 400 and 60 using the extended Euclidean algorithm. Also, find the values of *m* and *n*.

Ans.: Using the extended Euclidean algorithm as explained in Question 12, we have `x` = 400 and `y` = 60. Now, initializing `a` = `x` and `b` = `y`, we get `a` = 400 and `b` = 60. We also know that `c` = 1, `d` = 0, `e` = 0, and `f` = 1.

As `b`>0, we move to the first iteration of the while loop.

First iteration

`q` = 400/60=6

`r` = 400-6*60=40

`a` = 60

`b` = 40

`m` = 1-6*0=1

c = 0

d = 1

n = 0-6*1=-6

e = 1

f = -6

As 40 > 0, we move to the next iteration.

Second iteration

q = 60/40=1

r = 60-1*40=20

a = 40

b=20

m = 0-1*1=-1

c = 1

d = -1

n = 1-1*(-6)=7

e = -6

f = 7

As 20 > 0, we move to the next iteration.

Third iteration

q = 40/20=2

r = 40-2*20=0

a = 20

b = 0

m = 1-2*(-1)=3

c = -1

d = 3

n = (-6) -2*7=-20

e = 7

f = -20

As the value of b has become zero, the while loop terminates.

Now, gcd(x, y)=a, m = c, and n = e. Thus, gcd(400, 60)=20, m = -1, and n = 7.

10. Write a short note on modular arithmetic.

Ans.: In mathematics, to perform a division operation, we need two inputs, a divisor (say, m) and a dividend (say, x). After performing the operation we get two outputs, a quotient (say, q) and a remainder (say, r). That is, the division relationship can be expressed as follows:

x=m*q+r

However, in modular arithmetic, we are interested in only one output, that is, the remainder, while the other output (that is, the quotient) is not considered. Thus, in this case, the division operation can be expressed as a binary operator having two inputs, the integers x and m and only one output r. This binary operator is referred to as the *modulo operator* (written as mod). The input m (divisor) to the modulo operator is referred to as the *modulus*, while the output r is referred to as the *residue*. Thus, we can say that:

x mod m=r

where x is an integer from the set of integers Z={…,-3,-2,-1,0,1,2,3,…} and the modulus (m) and residue (r) are the positive integers. In case the value of x is negative, the value of r also comes out negative. Thus, to make it non-negative, the modulus m is added to r.

15. Find out the result of the following operations:

(a) 140 mod 10

(b) -73 mod 13

(c) 0 mod 7

Ans.: (a) When 140 is divided by 10, we get the remainder `r`=0. This means that 140 mod 10=0.

(b) When -73 is divided by 13, we get the remainder `r`=-8. To make *r* non-negative, we need to add modulus (13) to `r`. That is, `r` =-8+13=5. This means that -73 mod 13=5.

(c) When 0 is divided by 7, we get the remainder `r` =7. This means that 0 mod 7 = 7.

Example 2.6:

Find the result of 2^{90} mod 13.

Solution

Step 1: Split *x* and *y* into smaller parts using exponent rules as shown below:

$$2^{90} \bmod 13 = 2^{50} \times 2^{40}$$

Step 2: Calculate mod *n* for each part

$$2^{50} \bmod 13 = 1125899906842624 \bmod 13 = 4$$

$$2^{40} \bmod 13 = 1099511627776 \bmod 13 = 3$$

Step 3: Use modular multiplication properties to combine these two parts, we have

$$2^{90} \bmod 13 = \left(2^{50} \times 2^{40}\right) \bmod 13$$

$$= \left(2^{50} \bmod 13 \times 2^{40} \bmod 13\right) \bmod 13$$

$$= (4 \times 3) \bmod 13 = (12) \bmod 13 = 12$$

11. Explain the following with reference to modular arithmetic:

(a) Set of residues

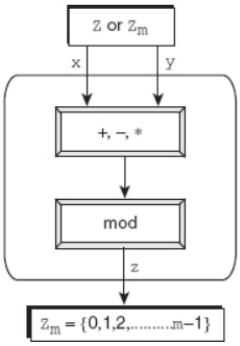
(b) Congruence

(c) Additive and multiplicative inverse

Ans.: (a) *Set of residues:* Consider a modulo operation `x mod m=r`, where `x` is an integer from a set of integers `Z` while `m` and `r` are positive integers. The result of this operation is always an integer less than `m`. That is, the value of `r` lies between 0 and `m`-1. Thus, it can be said that the modulo operation results in a set containing elements from 0 to `m`-1. In modular arithmetic, this set is called the *set of least residues modulo* `m` (denoted as Z_m) or simply the *set of residues*. There can be infinite possible instances of Z_m, one for each value of `m`. For example, Z_{11} can have 11 values {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, Z_4 can have four values {0, 1, 2, 3}, and so on.

Modular arithmetic allows three binary operations: addition, subtraction, and multiplication to be applied on the elements of Z_m. After applying each operation, the result obtained may need to be mapped to Z_m with the help of the modulo operator. To understand, consider three elements x, y, and z such that both x and y belong to Z (or Z_m) and z belongs to Z_m. Then the binary operations in Z_m can be expressed as (also see Figure 2.7):

$$(x+y) \bmod m=z$$
$$(x-y) \bmod m=z$$
$$(x*y) \bmod m=z$$



(b) *Congruence:* There is always a many-to-one relationship between Z and Z_m. That is, many elements of the set Z can map to a single element of Z_m. For example, modulo operations 3 mod 10, 13 mod 10, and 23 mod 10 result in the same value (equal to 3). Thus, these numbers (3, 13, and 23) are referred to as congruent mod 10 in modular arithmetic. To represent the congruence relationship between two integers, the *congruence operator* represented by the '≡' symbol is used. For example, we can write that $3 \equiv 13 \pmod{10}$, $13 \equiv 23 \pmod{10}$, and $3 \equiv 23 \pmod{10}$.

(c) *Additive and multiplicative inverse:* While working with modular arithmetic, we often need to determine the inverse of an element with respect to some operation. Two commonly required inverses are additive and multiplicative inverses. The former is the inverse with respect to the addition operation, while the latter is the inverse with respect to the multiplication operation.

Each element in modular arithmetic has only one additive inverse, which is always unique; sometimes, the additive inverse of an element is the element itself. Let x and y be two elements of the set Z_m. Now, x is said to be the *additive inverse* of y and vice versa if:

$$x+y \equiv 0 \pmod{m}$$

Simply put, the additive inverse of any element, say x in Z_m is equal to $m-x$. For example, the additive inverse of 11 in $Z_{15}=\{0,1,2,\ldots,13,14\}$ is 4 (15-11).

On the other hand, an element may or may not have a multiplicative inverse. Let x and y be two

elements of the set Z_m. Now, x is said to the multiplicative inverse of y and vice versa if:

$x*y \equiv 1 \pmod{m}$

For example, the multiplicative inverse of 7 in $Z_{15}=\{0,1,2,\ldots,13,14\}$ is 13, as $7*13 \equiv 1 \pmod{15}$.

The simple method to determine whether or not a number (x) in Z_m has a multiplicative inverse is to compute the GCD of x and m. If $\gcd(x,m)$ comes out to be one, x has a multiplicative inverse; otherwise, the multiplicative inverse for x in Z_m does not exist. For example, there does not exist a multiplicative inverse for number 5 in Z_{15} because $\gcd(5,15) \neq 1$. Notice that if $\gcd(x,m)=1$, x and m are said to be *relatively prime*.

Examples 2.2:

2.2.1 $10 \equiv 2 \pmod{4}$, Because $\frac{10-2}{4} = \frac{8}{4} = 2$

2.2.2 $38 \equiv 3 \pmod{5}$, Because $\frac{38-3}{5} = \frac{35}{5} = 7$

2.2.3 $-46 \equiv 4 \pmod{10}$, Because $\frac{-46-4}{10} = \frac{-50}{10} = -5$

2.2.4 $38 \not\equiv 6 \pmod{5}$, Because $\frac{38-6}{5} = \frac{32}{5} = 6.4$