Extended Euclid Algorithm

Extended Euclidean algorithm helps us to find two integers x and y such that ax+by = gcd(a,b), given a and b. The existence of two such integers x and y is given by **Bezout's Identity.**

- **Bezout's Identity:**

  The bezout's identity states that if d = gcd(a,b), then there always exists integers x and y such that ax+by = d, for non-zero integers a and b. Here d is also the smallest positive integer for which ax+by = d has a solution with integral values of x and y.

  From bezout's identity, the following lemmas also hold:
  - If a,b and c are integers and a | bc and gcd(a,b) = 1, then a | c.
  - If a,b and c are integers and a | b, a | c and gcd(a,b) = 1 then ab | c.

So, finding integers x and y such that ax+by = gcd(a,b) can be done by applying the Extended Euclidean algorithm.

The Extended Euclidean algorithm can be viewed as reversing the steps of the Euclidean algorithm having the GCD and the numbers a and b, and working recursively backward.

- Let us say that
  $$ax + by = GCD(a,b),$$
  given non-negative integers a and b we need to find x and y.
- As we know that
  $$GCD(a,b) = GCD(b,a \bmod b),$$
  hence the above equation can be re-written as
  $$bx_1 + (a \bmod b)y_1 = GCD(a,b)$$
  for some x_1 and y_1.
- We can write
  $$(a \bmod b) = a - b*floor(a/b).$$
  Substituting in the above equation we get
  $$bx_1 + (a - b*floor(a/b))y_1 = GCD(a,b)$$
  or simplifying the equation as
  $$b(x_1 - floor(a/b))y_1 + ay_1 = GCD(a,b)$$
- Comparing the coefficients in our initial equation ax + by = GCD(a,b) with $b(x_1 - floor(a/b).y_1) + ay_1$ we get the relation between $\{x,x_1\}$ and $\{y,y_1\}$ as -
  - $y = x_1 - floor(a/b).y_1$
  - $x = y_1$

Hence, we call GCD(b, a mod b) recursively to obtain the values of x_1 and y_1, which are used to compute the values of x and y.

Example: Let say a = 16 and b = 10

First, let's calculate GCD using the Euclid algorithm

$$16 = 1 * 10 + 6 \qquad -(1)$$
$$10 = 1 * 6 + 4 \qquad -(2)$$
$$6 = 1 * 4 + 2 \qquad -(3)$$
$$4 = 2 * 2 + 0 \qquad -(4)$$

From this, the last non-zero remainder (GCD) is 2. Now using the Extended Euclid algorithm
From eq (3)

$$2 = 6 + (-1)*4 \qquad -(5)$$

From eq (2)

$$4 = 10 + (-1)*6 \qquad -(6)$$

Substituting the value of 4 from eq (6) in eq (5)

$$2 = 6 + (-1)\{10 + (-1)*6\}$$
$$2 = 2*6 + (-1)*10 \qquad -(7)$$

Substituting value of 6 from eq(1) in eq(7)

$$2 = 2*\{16 + (-1) * 10\} + (-1)*10$$

On simplifying

$$2 = 2*16 + (-3)*10$$

Since we now write the GCD as a linear combination of two integers a and b, we compare and conclude the values of x and y as

$$x = 2$$
$$y = -3$$

Pseudocode:

```
/*
        Input a and b are integers, while the solutions x and y to be found corresponding to the
        coefficients a and b are passed as a reference to the function
        Returns the gcd(a,b) and solves for x and y : ax + by = gcd(a,b)
*/
function extendedEuclid(a, b, ref(x), ref(y))

        //   If b equals 0 assign the solutions x = 1 and y = 0, and return 'a' as the GCD
        if b equals 0
```

```
        x = 1
        y = 0
        return a

    /*
        Otherwise, recursively call GCD to get x1, y1 which helps us compute
        x and y for the original equation
    */
    x1, y1
    d = gcd(b, a mod b, x1, y1)
    x = y1
    y = x1 - y1 * (a / b)

    return d
```

Time complexity: O($\log_2$max(a,b)), where a and b are the given integers

The algorithm is useful in finding the modular multiplicative inverse of x under modulo m, given x and m are co-prime.