Diophantine Equations

A diophantine equation is of the form ax + by = c, where a, b and c are constants and the values of x and y are to be found that can be positive, negative, or zero. However, each number in the equation must be an integer.

A diophantine equation can be solved if and only if c is divisible by gcd(a,b), otherwise, we cannot find any integral solutions x and y that satisfy the equation ax + by = c.
The equation can be solved because we can reduce our equation to the form ax + by = gcd(a,b) if c is divisible by gcd(a,b) and hence find x and y such that ax + by = c from **Extended Euclidean algorithm.**

For Example:
Given the equation:
21x + 39y = 9, when Euclid's algorithm recursively calculated the GCD(39,21), it gives GCD(39,21) = GCD(21,18) = GCD(18,3) = GCD(3,0) = 3.

So, the equations followed which calculating GCD can be written as -
39 - 21*1 = 18
21 - 18*1 = 3

So, using substitution we get the equation as 39*1 - 18*2 = 3, and multiplying the equation by 3, we get -
39*3 - 18*6 = 9. Hence the solutions for the equations are x = 3 and y = 6. This is how we can obtain the solutions to the given diophantine equation from the steps used in the Extended Euclidean algorithm.

A solution to the diophantine equation is not unique, we can have an infinite number of solutions to the equation if we know just one solution. Let us say we have solution {x,y} to a diophantine equation, then -
{x + (k * b / gcd(a,b)), y - (k * a / gcd(a,b))} are all possible solutions to the equation where k is any integer.

Here is the Pseudocode for finding any solution to the equation ax + by = c

Pseudocode:

```
/*
        a, b, c are inputs to the function, and x0, y0 will be the solutions. g will be the
        extended gcd of a and b. All of x0, y0, and g will be passed by reference
*/
function find_any_solution(a, b, c, ref(x0), ref(y0), ref(g)) {
        //   finding the gcd using extended euclid's gcd algorithm
        //   g stores the extended gcd and x0 , y0 and g are passed by reference
        g = extendedEuclid(abs(a), abs(b), x0, y0)

        //   checking if any solution exists
        if (c % g) {
                return false
        }

        //   scale up x0 and y0 by c / g
        x0 *= c / g
        y0 *= c / g

        //   adjust for signs of x0 and y0
        if (a < 0)
                x0 = -x0
        if (b < 0)
                y0 = -y0

        return true
}
```

Time complexity: O($\log_2$max(a,b)), where a and b are the given integers. This is bounded by the time for calculating the extended Euclid gcd after which only a constant number of operations are performed.

Space Complexity: O($\log_2$max(a,b)) which is equal to the space complexity of extendedEuclid function.