Euclid's Algorithm:

Euclid's algorithm is a very efficient method to find the GCD of two numbers 'a' and 'b', instead of the naive approach of finding the factor of the numbers and then finding the common factors and finally taking their product.

The Euclidean algorithm is based on the equation given as GCD(a,b) = GCD(b,a mod b) and we recursively call the function until a%b is non-zero. It is based on the simple concept that GCD (a, b) = GCD (a, b - a) if a > b and vice versa.

So the Algorithm looks as follows :-

- if b = 0 GCD(a, b) = a
- Otherwise GCD(a, b) = GCD(b, a mod b)

Pseudocode:

```
/* Input a and b are integers, returns the gcd(a,b) */
function GCD(a,b)

        //   Check if b equals 0 then return 'a' as the GCD
        if b equals 0
                return a
        //   Otherwise recursively call GCD(b,a mod b)
        else
                return GCD(b,a mod b)
```

Time complexity: O($\log_2$max(a,b)), where a and b are the given integers. The exact proof for time complexity is out of the scope of this lecture. The space complexity is also O($\log_2$max(a,b)) since that is the maximum depth of recursion stack possible.