Problems

In computer science, a *problem* refers to a general class of problems where in mathematics it might refer to a specific example. Thus our solutions are algorithms which are more generally applicable to a class of problems. Some examples of problem classes include

- Sorting problems, where an instance of a problem is a sequence of items.
- Graph colouring, where an instance is a graph.
- Equation solving problems, where an instance is a system of equations.

An *algorithm* is a finite sequence of instructions with

- No ambiguity and each step precisely defined
-

Euclid's Algorithm

With the observation that `gcd(m, n)== gcd(max(m, n)- min(m, n), min(m, n))` we have a recursive relation to find the greatest common denominator of two values.

```
int gcd(int m, int n) {
    if (m == n)
        return m;
    else
        return gcd(max(m, n) - min(m, n), min(m, n));
}
```

However, for large numbers this algorithm requires a very large number of iterations. Luckily, we have an operator that allows for easy repeated subtractions. The modulo operator allows us to instead use the algorithm `gcd(m, n)= gcd(n, n mod m)`.

```
int gcd(int m, int n) {
    if (n == 0)
        return m;
    else
        return gcd(n, n % m);
}
```

Although these two algorithms accomplish the same task, one performs it with much greater efficiency. Rather than perform this algorithm recursively we can do it iteratively, saving us the overhead of a function call.

```
int gcd(int m, int n) {
    while n != 0 {
        int r = m mod n;
        m = n;
        n = r;
    }
    return m;
}
```