Algorithms

What is an algorithm?

An algorithm is a procedure, recipe, process to accopmplish a task. It takes value as input and delivers a value as output.

How to develop an algortihm?

The development of an algorithm has four crucial steps:

1. **Specifiaction**: Be clear what the problem is
2. **Design**: Specify structure of the solution, usually in pseudcode
3. **Development**: Convert pseudocode in chosen language (C, Python, Java etc.)
4. **Testing**: if all inputs deliver all necessary outputs

Example: Linear search in Pseudocode

```
# Linear search last occurence
    p = NIL;
    for i = 1 to n do
        if A[i]==v then p=i;
    return p;
```

Here we are defining our variable p as 0. For the range of 1 to n in our array A, we are checking the values (v). If we find a value which matches our input, we set it equal to p and return it.

```
# Linear search first occurence
    i = 1;
    while i <= n and A[i] != v do i++;
    if i <= n then return i;
    else return NIL;
```

Here we are setting i to 1. While i is less or equal number of array elements and there is no match in value with the array, keeping adding 1. If i is less or equal n, return i else return 0.

```c
#include <stdio.h>

#define n 5

int i, v;
int a[] = { 11, 1, 4, -3, 22 };
int main() {
    i = 0; v = -2;
    while (i < n && a[i] != v) { i++; }
    if (i < n) { printf("%d\n", i); }
    else { printf("NIL\n"); }
}
```

Here we see how we would do the linear search task with C. As visible, the pseudocode comes very close to what we write in C. Pseudcode allows us to roughly structure out our code, before we write it any language. The Java code for this algorithm is also very close to pseudocode, making it a invaluable tool for construction of algorithms.

```
import java.io.*;
class search {
    static int n = 5;
    static int i, v;
    static int a[] = { 11, 1, 4, -3, 22 };
public static void main(String args[]) {
    i = 0; v = 22;
    while (i < n && a[i] != v) { i++; }
    if (i < n) { System.out.println(i); }
    else { System.out.println("NIL"); }
    }
}
```

Prime number filter in pseudocode

```
A[0] = False; A[1] = False;
for i = 2 to n do A[i] = True;
for i = 2 to floor(n/2) do
    for j = 2 to floor(n/i) do
        A[i*j] = False;
```

This is called the *Sieve of Eratosthenes*. Here we are given an array with all numbers from 0 to i. We set the first two elements, being `A[0] = 0` and `A[1] = 1`, to False. We know that these 2 numbers are not primes. Starting from `i = 2`, we set all the other elements to True. We then divide our number of array elements by 2 using `(floor(n/2))` - the floor function rounds the float to an integer. What we then do is check which of the remaining elements we have in our array are possible to reached as a multiple `A[i*j]` of our elements i and j. Here we can see how this happens .

Sorting algotihms

Bubble sort

Selection sort

Insertion sort