Recursion

Stefan Zorcic

Recursion has a multitude of uses within the world of programming from graph theory (dfs, cycle detection) to dynamic programming, recursion is at the basis of many algorithms. Recursion algorithms have two fundamental parts:

- The base case(s)
- Recursion (calling itself)

Recursion follows the stack data structure as task (calls to the method) at the top of the stack execute first.

A classic dynamic programming programming that uses recursion is calculating the xth number in the Fibonacci sequence. The naïve method that many beginner programmers do is using a recursion algorithm with a time complexity of $O(2^n)$ where n is the index of the number searched for. The example of such a solution is as follows.

```
public static int fib(int n){
    if ( n <= 1 ) { return 1; }
    return fib(n-1) + fib (n-2);
}
```

**The recursive algorithm above utilizes any values below 1 as the base case*

This solution can be improved to a time complexity of O(n) using memoization. Memoization is a process where computed values are stored in a cache to be called later to prevent resources being wasted on re-computation.

```
static int n = 5;
static int [] dp = new int [n+1];

public static int fib(int n){
    if ( dp[n] > 0 ) { return dp[n]; }
    if ( n <= 1 ) { return 1; }
    dp[n]=fib(n-1) + fib (n-2);
    return dp[n];
}
```