Unit 3 – Algorithms

Recursion

What is recursion? Why use recursion? What are the types of recursion?

1. Recursion – a function calls itself within its definition.
   1. Recall: We need a stopping condition (base case).
   2. Common traits between iterative/recursive cases:
      1. Initial values
      2. A changing initial value
      3. And a terminating condition.
   3. Imperative vs Declarative code
      1. Imperative caters to the computer’s understanding (mental model)
      2. Whereas declarative tends to be more “readable (human mental model)
      3. Recursion tends to be more declarative.
   4. Recursive solutions tend to describe *what* the solution is.
   5. Recursive functions forces you to write declaratively, rather than imperatively.

const factorial = n => {

*if* (n===0) *return* 1;

*else* *return* n\*factorial(n-1);

}

Vs

const factorialTailCall = (n, product = 1) => {

*if* (n===0) *return* product;

*return* factorial (n-1, product\*n)

}

1. Tail-call recursion
   1. In tail call recursion, because the only thing returned is a call to a function, it does not need to store the original function. Instead, it will use **stack frame replacement** and pull the result directly from the parameters.
2. Mutual Recursion
   1. Functions call one another.

function isEven(n) {

*if* (n === 0) *return* true;

*return* isOdd(n - 1);

}

function isOdd(n) {

*if* (n === 0) *return* false;

*return* isEven(n - 1);

}

1. Tree Recursion
   1. This is kind of the biggest topic for today.
   2. AKA Multiple recursion.
   3. Recall Fibonacci code.