Lecture 5

Fibonacci Series

In continuation with the theme of the last lecture we define another infinite series — the fibonacci series. Ofcourse all of you know the Fibonacci Series. It is defined by the linear recurrence relation $F_{i+2} = F_i + F_{i+1}$. We assume that $F_0 = 1$ and $F_1 = 1$ to begin with.

The defintion is straight forward; it is just a one liner, but we use this as an excuse to introduce two standard list functions

5.1 Ziping a list

The zip of the list $x_0, ..., x_n$ and $y_0, \ldots, y_m$ is the list of tuples $(x_0, y_0), \ldots, (x_k, y_k)$ where k is the minimum of n and m.

```
> zip :: [a] -> [b] -> [(a,b)]
> zip  []       _    = []
> zip   _      []    = []
> zip (x:xs) (y:ys) = (x,y) : zip xs ys
```

The function `zipWith` is a general form of ziping which instead of tupling combines the values with an input functions.

```
> zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
> zipWith f xs ys = map (uncurry f) $ zip xs ys
```

We can now give the code for fibonacci numbers.

```
> fib = 1:1:zipWith (+) fib (tail fib)
```

Notice that the zip of `fib` and `tail fib` gives a set of tuples whose caluse are consecutive fibonacci numbers. Therefore, once the intial values are set all that is required is zipping through with a `(+)` operation.