Functional Programming

Laziness

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2019

Laziness

- Haskell does not evaluate function arguments unless they are demanded to calculate the function's result
- Data structures are only evaluated as much as they are needed

Infinite lists

```
nat :: [Integer]
nat = [0..] -- cheating!

ones :: [Integer]
ones = undefined

-- |create an infinite list of argument value
repeat' :: a -> [a]
repeat' a = undefined
```

Fibonacci numbers

```
fib :: [Integer]
fib = undefined
```

Sieve of Eratosthenes

```
primes :: [Integer]
primes = undefined
```

The minimum tree

A binary tree type

A variant of binary trees has information only at the nodes.

```
data BTree = Leaf Int | Branch BTree BTree
```

The minimum tree

A binary tree type

A variant of binary trees has information only at the nodes.

```
data BTree = Leaf Int | Branch BTree BTree
```

Task

The minimum tree

A binary tree type

A variant of binary trees has information only at the nodes.

```
data BTree = Leaf Int | Branch BTree BTree
```

Task

- Write a function `mintree` that replaces the value at each leaf by the minimum value of all leaves in the tree.

The minimum tree

A binary tree type

A variant of binary trees has information only at the nodes.

```
data BTree = Leaf Int | Branch BTree BTree
```

Task

- Write a function `mintree` that replaces the value at each leaf by the minimum value of all leaves in the tree.
- Only one traversal of the tree is allowed!