Bin. strom

```
data GeneralTree a b = Leaf b | Tree (GTree a b) a (GTree a b)
depth :: Tree a -> Int
depth Ext = 0
depth (Node l x p) = (max (depth l) (depth p)) + 1
```

```
find x Ext = False
find x (Node l y r) = case compare x y of
    LT = find x l
    EQ = True
    GT = find x r
```

Multimnožiny
Tree (a, Int)

```
find Ext = 0
    : compare x (fst y) of
    EQ -> snd y
```

foldTree ... size = ft (\l x r -> l + r + 1) 0

depth T = ft (\l x r -> (max l r) + 1) 0

```
find x = foldTree (\l y r -> l || (y == x) || r) False
```

$\Theta(n)$

```
foldTree (\l x r = x : (l ++ r))
```

a => b

"Constraint

---

akce -- 1) dostaneš hodnotu, na její základě něco uroudí
2) dostaneš akci, jak na ni reagují

Funktor

```
fmap :: (a -> b) -> (f a -> f b)
(<$>)
```

```
liftA3 f x y z = f <$> x <*> y <*> z
```

Aplikativní funktory

```
Lift A2 :: (a -> b -> c) -> (A a -> b -> c)
pure :: a -> (f a)
(<*>) :: f (a -> b) -> f a -> f b
```

Monáda

```
(a, a -> b -> b)
(a -> a -> b)
bind :: a -> (a -> b) -> b
>>=
```