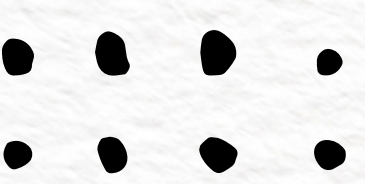


Felipe Colpo

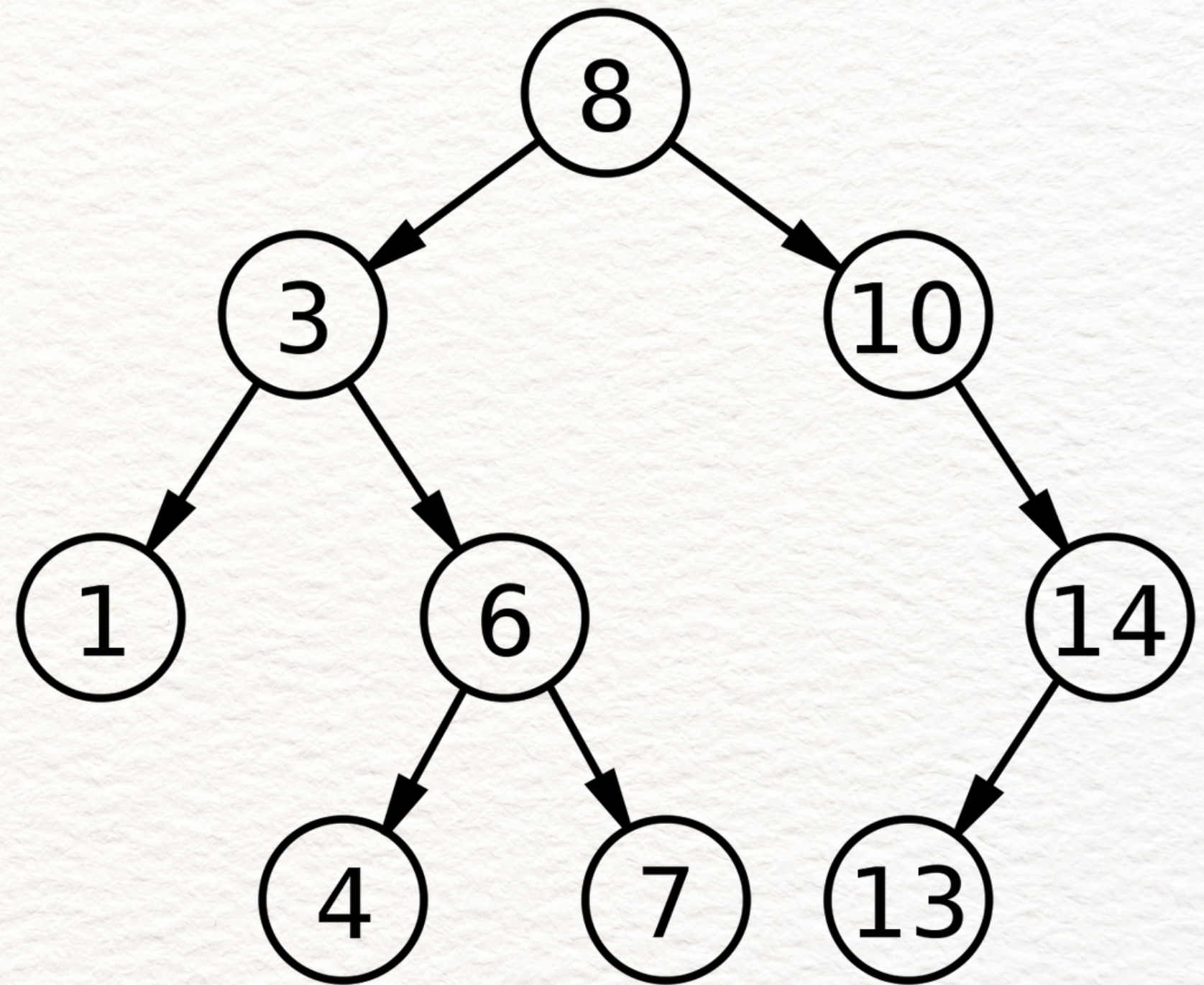
Árvores binárias e sua implementação em haskell

<https://replit.com/@Felipe-ColpoCol/BinaryTreesELC117#Main.hs>



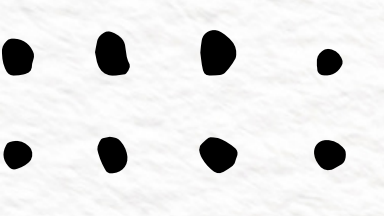


O que é uma árvore binária?



Como Funcionam Data Types em haskell

- Funcionam com structs de linguagens declarativas
- Permitem a criação de tipos diferentes dos padrões propostos pela linguagem(Int, Float, String)
- Serve como uma caixa para guardar diferentes tipos




```
data Person = Person String Int deriving (Show)
```

Keyword data permite inferir o nome do tipo a ser criado

Os valores String e Int Compõe os diferentes tipos armazenados na "struct" pessoa

O keyword deriving significa que tipo ira implementar o qualidade de Show


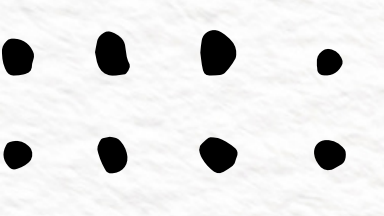


Implementando uma BST

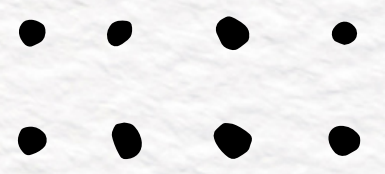

```
data Tree = Empty | Node Int Tree Tree

insert :: Int -> Tree -> Tree
insert x Empty = Node x Empty Empty
insert x (Node v left right)
  | x < v = Node v (insert x left) right
  | otherwise = Node v left (insert x right)

search :: Int -> Tree -> Bool
search x Empty = False
search x (Node v left right)
  | x == v = True
  | x < v = search x left
  | otherwise = search x right
```

**E Como funciona para outros
tipos?**




```
data Arvore a = Empty | No a (Arvore a) (Arvore a) deriving (Show, Eq)
```

```
insert:: Ord a => Arvore a -> a -> Arvore a
```

```
insert Empty val = No val Empty Empty
```

```
insert (No v left right) val = if val < v then No v (insert left val) right
    else No v left (insert right val)
```