

Haskell - A Search Tree Application

Functional Programming

Jens Egholm Pedersen and Anders Kalhauge



Spring 2018

Search Trees

- Binary Search Tree

- Balanced Search Trees

Haskell IO

- Some basic examples

- Chapter 8 and 9

Assignment

Search Trees

Binary Search Tree

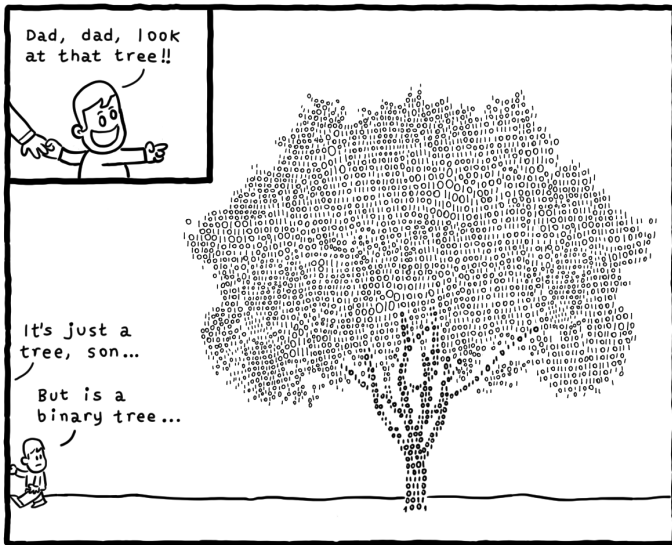
Balanced Search Trees

Haskell IO

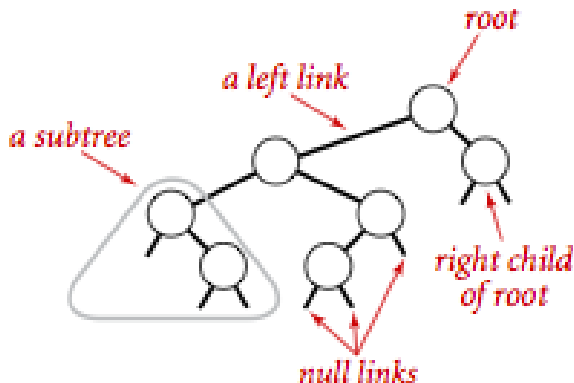
Some basic examples

Chapter 8 and 9

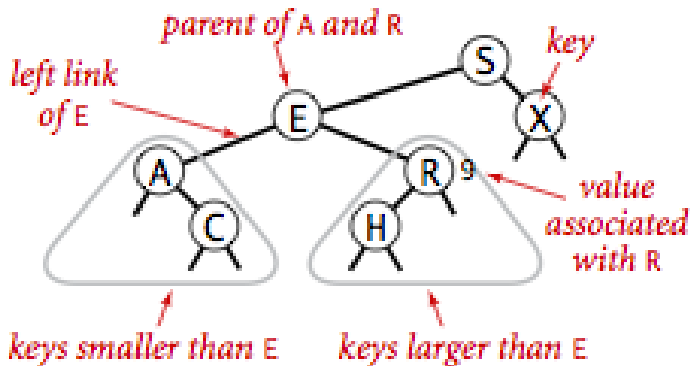
Assignment



Daniel Stori {turnoff.us}



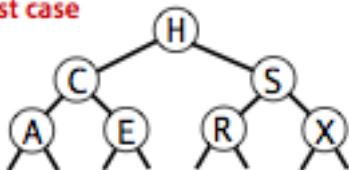
Anatomy of a binary tree



Anatomy of a binary search tree

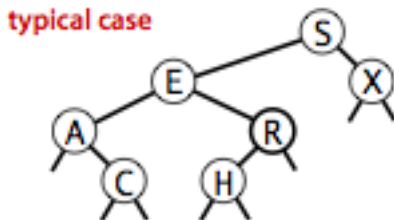
Analysis

best case



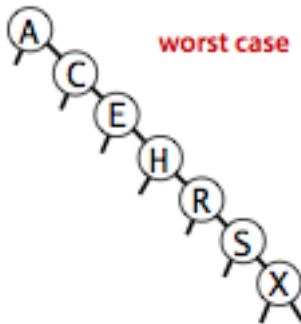
$$O(\log n)$$

Analysis



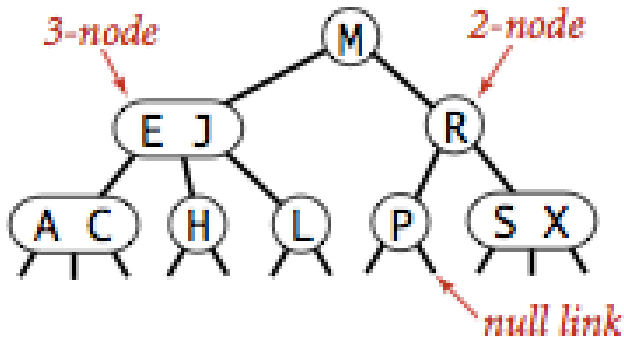
$$O(\log n)$$

Analysis



$$O(n)$$

Anatomy

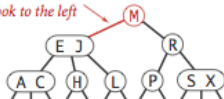


Anatomy of a 2-3 search tree

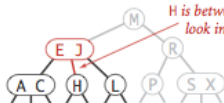
Searching

successful search for H

H is less than M so
look to the left



H is between E and J so
look in the middle

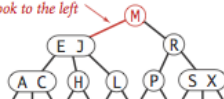


found H so return value (search hit)

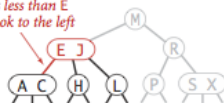


unsuccessful search for B

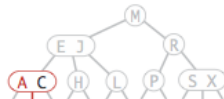
B is less than M so
look to the left



B is less than E
so look to the left

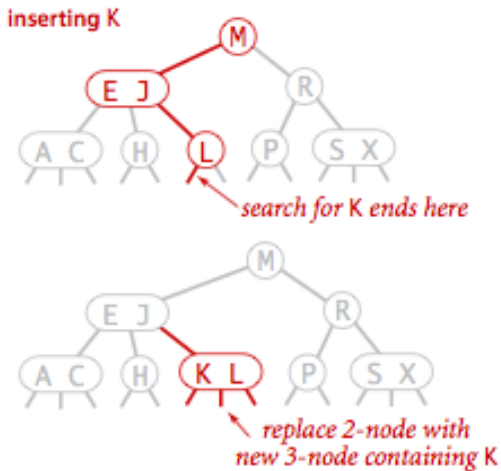


B is between A and C so look in the middle
link is null so B is not in the tree (search miss)



Search hit (left) and search miss (right) in a 2-3 tree

Inserting



Insert into a 2-node

Inserting

Inserting S



← no room for S



← make a 4-node

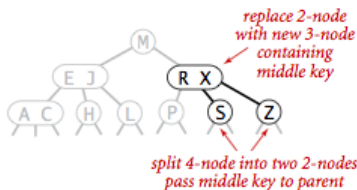
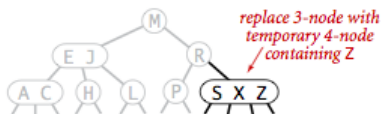
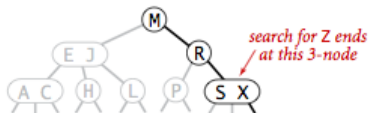


split 4-node into
this 2-3 tree

Insert into a single 3-node

Inserting

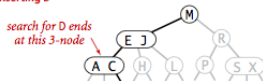
inserting Z



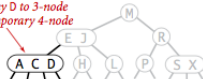
Insert into a 3-node whose parent is a 2-node

Inserting

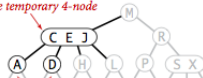
inserting D



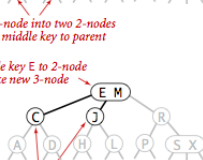
add new key D to 3-node to make temporary 4-node



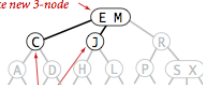
add middle key C to 3-node to make temporary 4-node



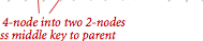
split 4-node into two 2-nodes pass middle key to parent



add middle key E to 2-node to make new 3-node



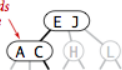
split 4-node into two 2-nodes pass middle key to parent



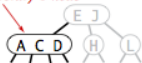
Insert into a 3-node whose parent is a 3-node

inserting D

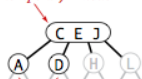
search for D ends at this 3-node



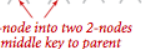
add new key D to 3-node to make temporary 4-node



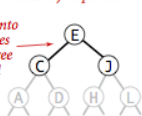
add middle key C to 3-node to make temporary 4-node



split 4-node into two 2-nodes pass middle key to parent



split 4-node into three 2-nodes increasing tree height by 1



Splitting the root

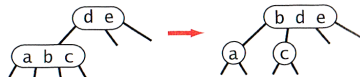
Summing up

root



parent is a 3-node

left



parent is a 2-node

left



middle



right



right



Splitting a temporary 4-node in a 2-3 tree (summary)

Search Trees

Binary Search Tree

Balanced Search Trees

Haskell IO

Some basic examples

Chapter 8 and 9

Assignment

```
main :: IO ()
main = do
  putStrLn "Hi, what's your name?"
  name <- getLine
  putStrLn ("Hello " ++ name)
```

In GHCi

```
ghci> :t getLine
getLine :: IO String
ghci> :t putStrLn
putStrLn :: String -> IO ()
ghci>
```

```
main :: IO ()
main = do
  putStrLn "Hi, what's your age?"
  line <- getLine
  let age = (read line :: Int)
  putStrLn ((show age)++" years")
```

```
import System.IO

main :: IO ()
main = do
  withFile "numbers.txt" ReadMode (\handle -> do
    contents <- hGetContents handle
    let numberLines = lines contents
    putStr (contents++(numberLines!!2)++ "\n")
  )
```

```
fact :: Integer -> Integer
fact 0 = 1
fact n = n * fact (n - 1)

main :: IO ()
main = do
    putStrLn ("50! = " ++ (show (fact 50)))
```

```
50! = 30414093201713378043612608166064768844377641568
      960512000000000000
```

Consult chapter 8 and 9 in Learn You a Haskell for Great Good!

Search Trees

Binary Search Tree

Balanced Search Trees

Haskell IO

Some basic examples

Chapter 8 and 9

Assignment

Create a Haskell application that:

- ❑ Implements a balanced search tree¹ with functions for finding items, inserting items, and traversing the tree.
- ❑ Implements a console interface with:
 - ❑ Search item
 - ❑ Insert item
 - ❑ List all items in order
 - ❑ Load items from text file

Start with integer numbers as items, expand with any item deferring **Ord**

Can be done individually or in groups.

¹or at least a binary search tree