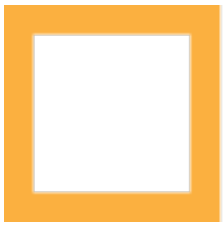
Haskell - A Search Tree Application

Functional Programming

Jens Egholm Pedersen and Anders Kalhauge



cphbusiness

Spring 2018

Outline

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Search Trees

- Binary Search Tree
- Balanced Search Trees

Haskell IO

- Some basic examples
- Chapter 8 and 9

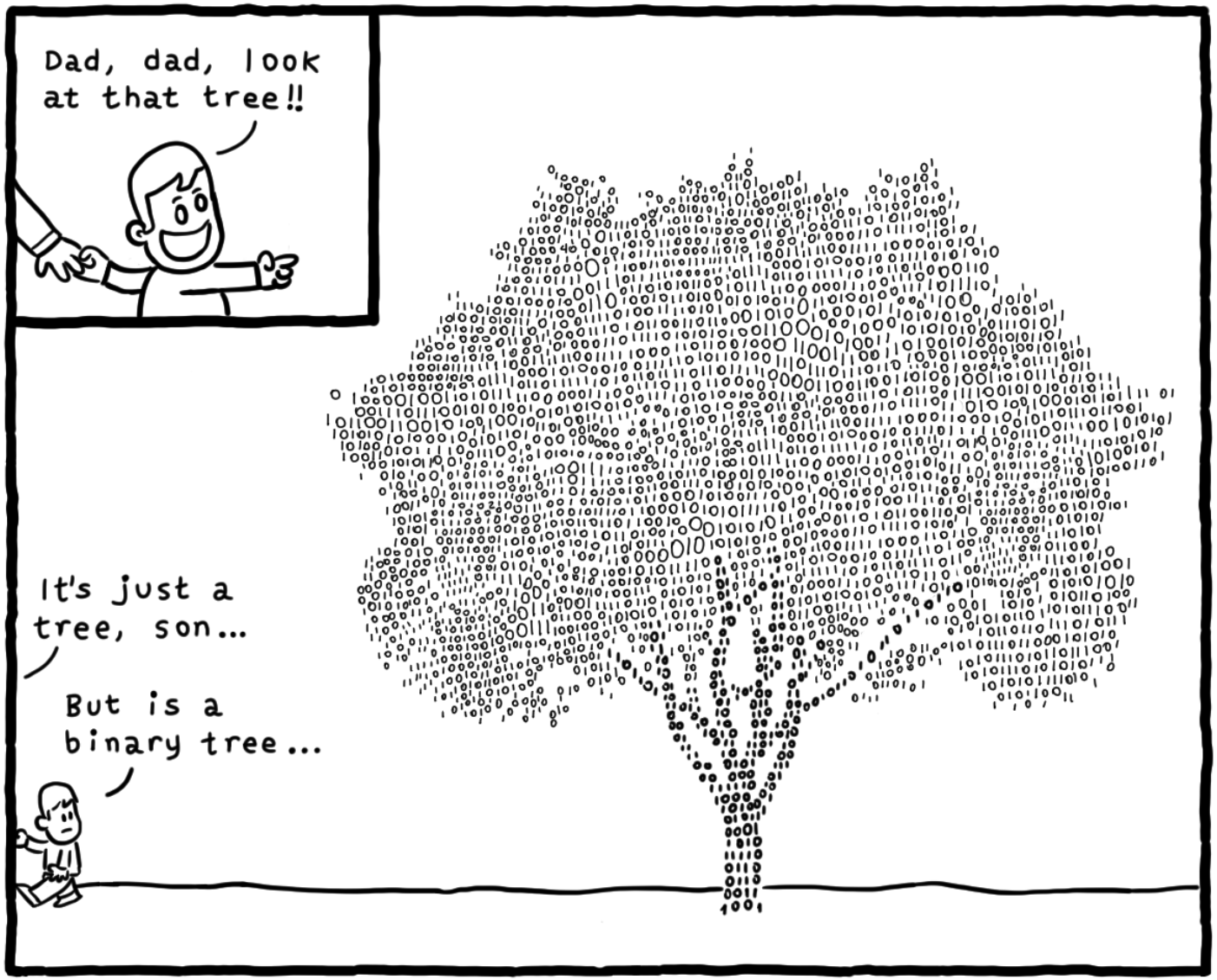
Assignment

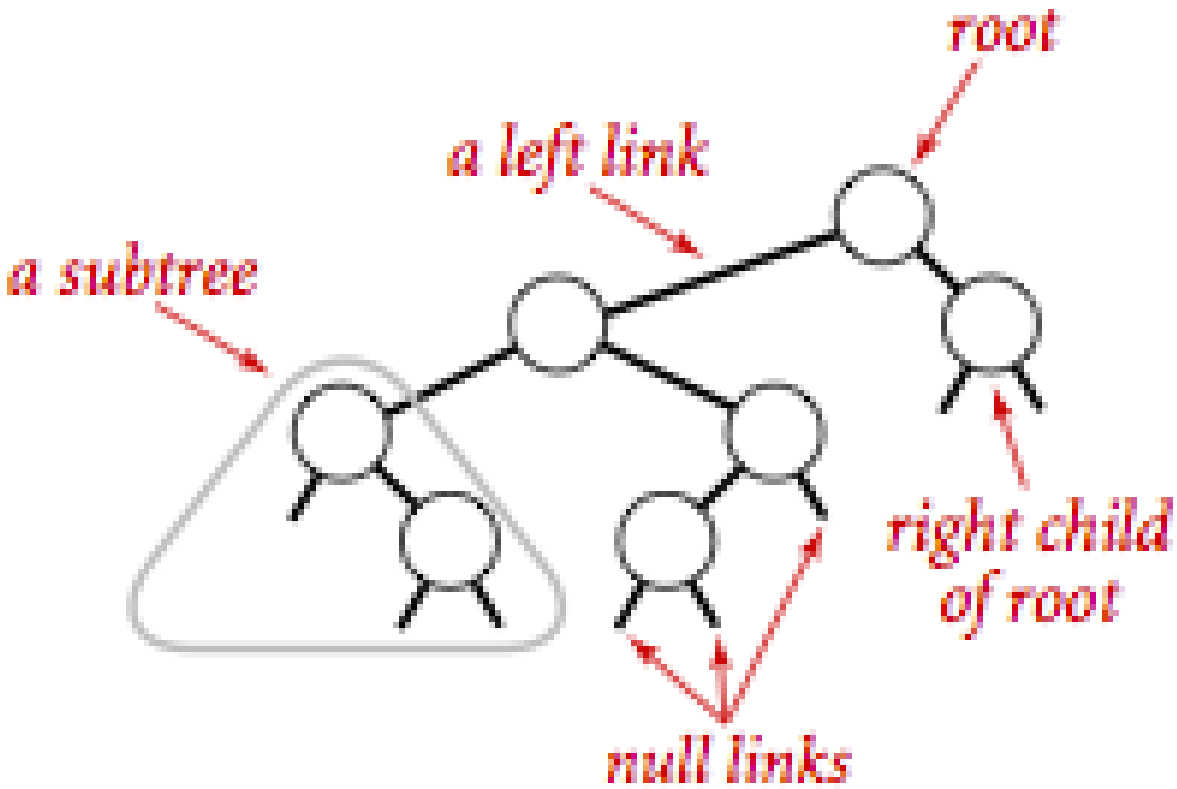
Search Trees
- Binary Search Tree
- Balanced Search Trees

Haskell IO
- Some basic examples
- Chapter 8 and 9

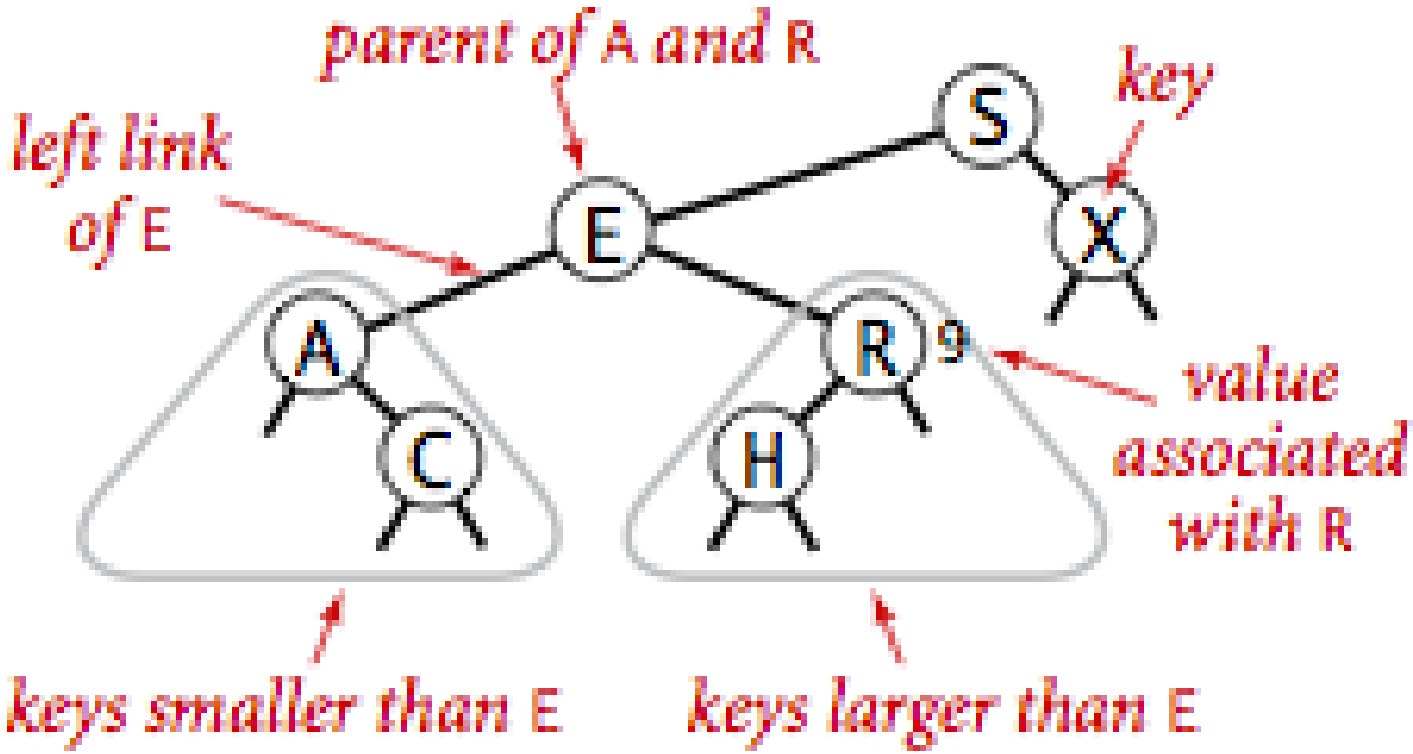
Assignment

Binary Trees





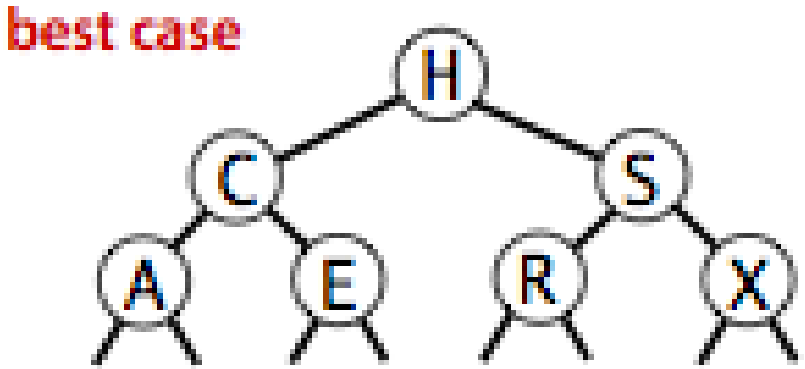
Anatomy of a binary tree



Anatomy of a binary search tree

cphbusiness
COPENHAGEN BUSINESS ACADEMY

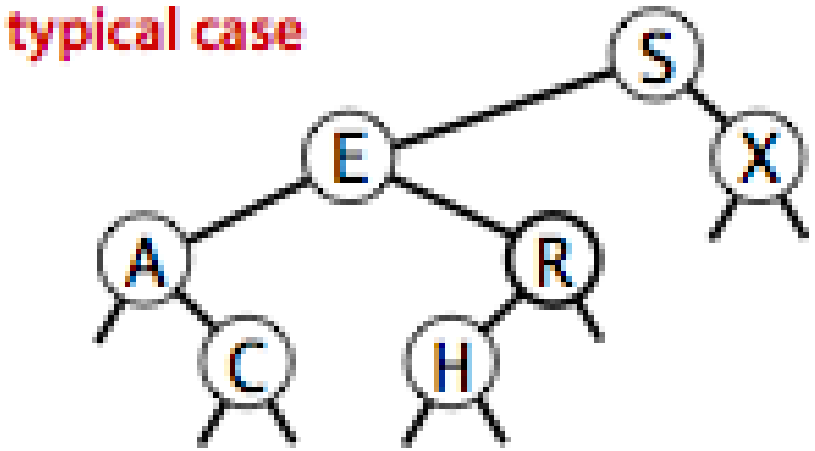
Analysis



$$O(\log n)$$

cphbusiness
COPENHAGEN BUSINESS ACADEMY

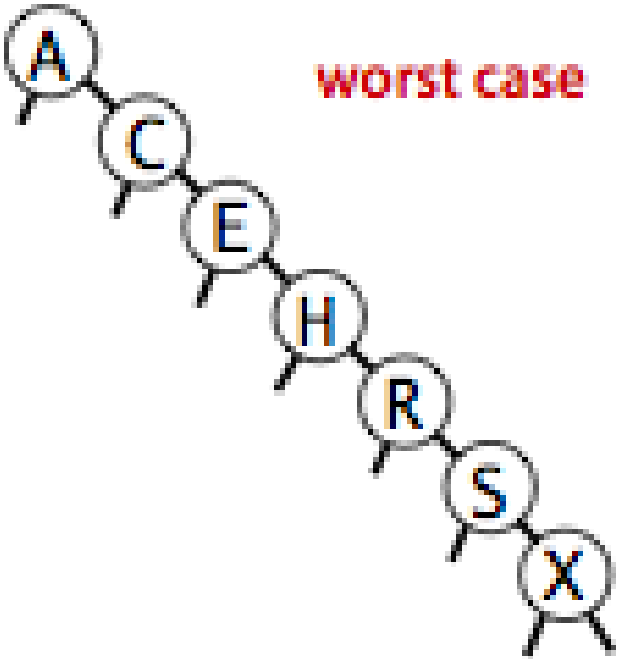
Analysis



$$O(\log n)$$

cphbusiness
COPENHAGEN BUSINESS ACADEMY

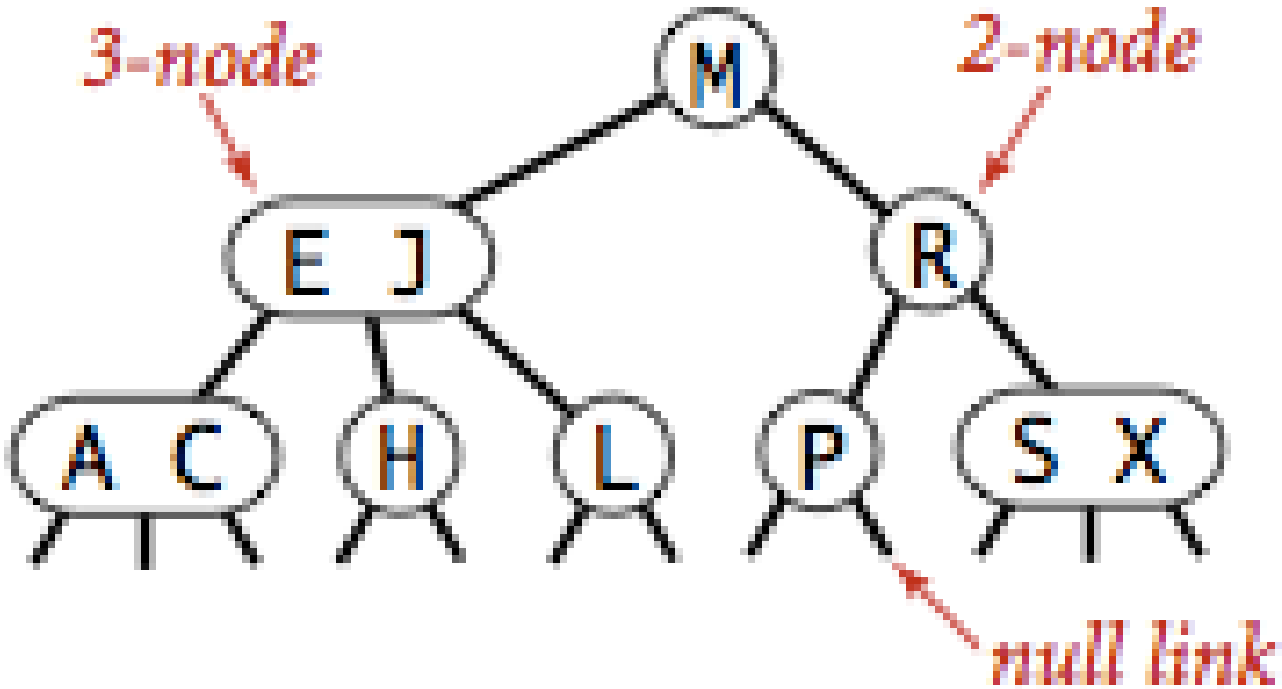
Analysis



$$O(n)$$

cphbusiness
COPENHAGEN BUSINESS ACADEMY

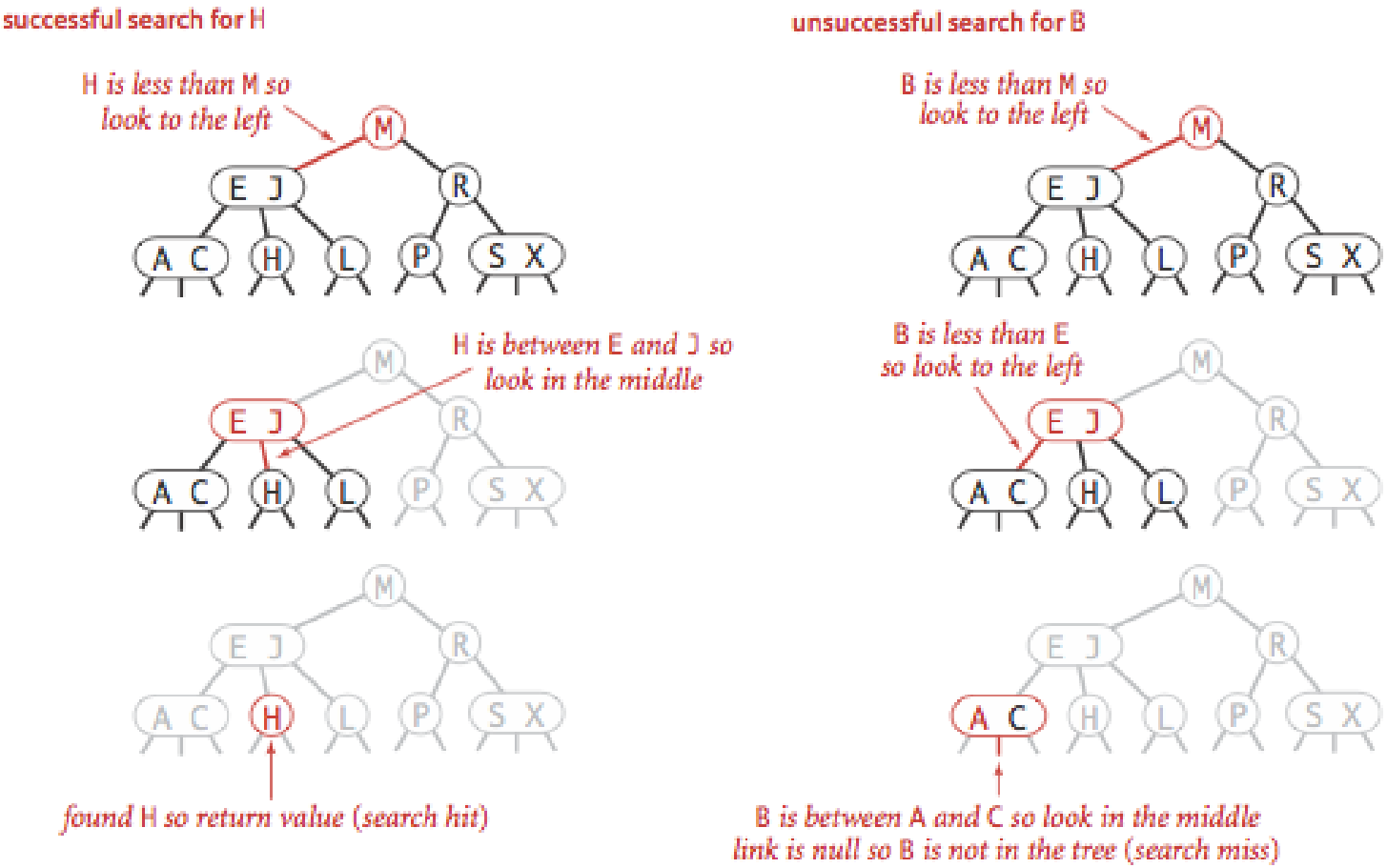
Anatomy



Anatomy of a 2-3 search tree

cphbusiness
COPENHAGEN BUSINESS ACADEMY

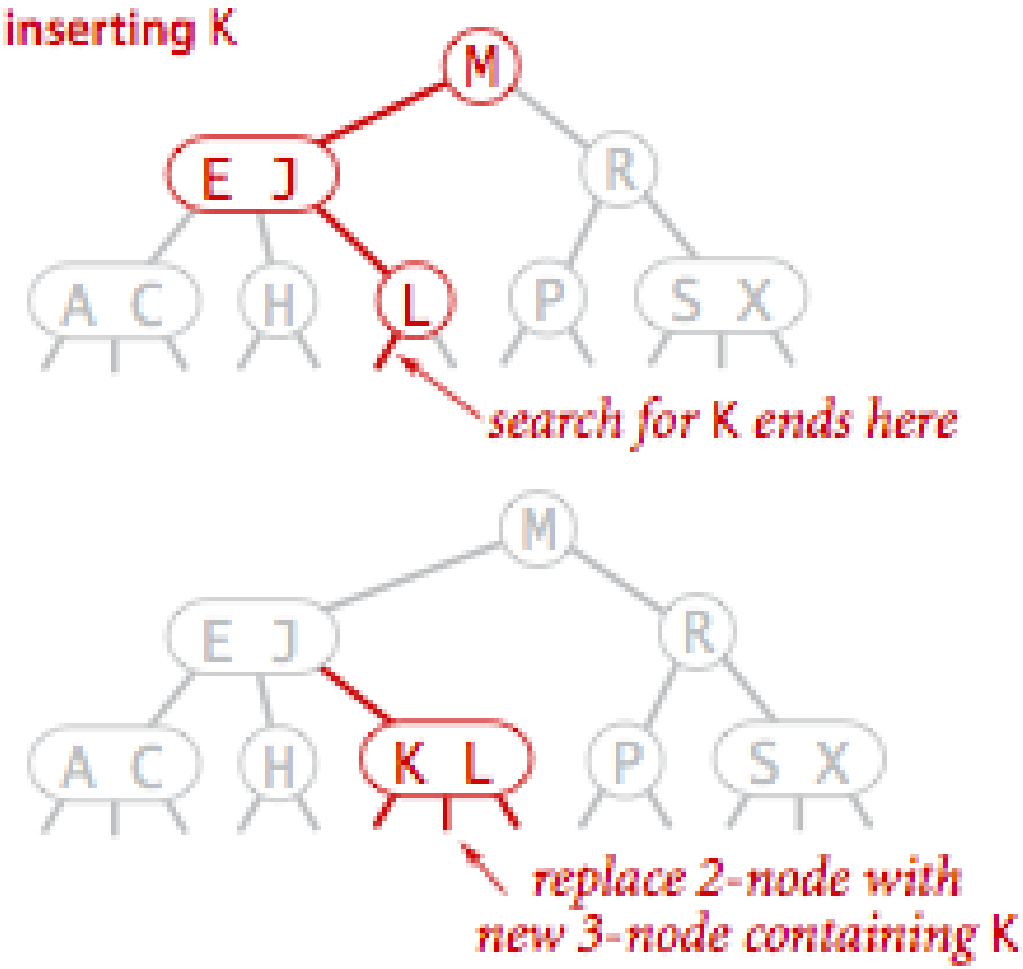
Searching



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

cphbusiness
COPENHAGEN BUSINESS ACADEMY

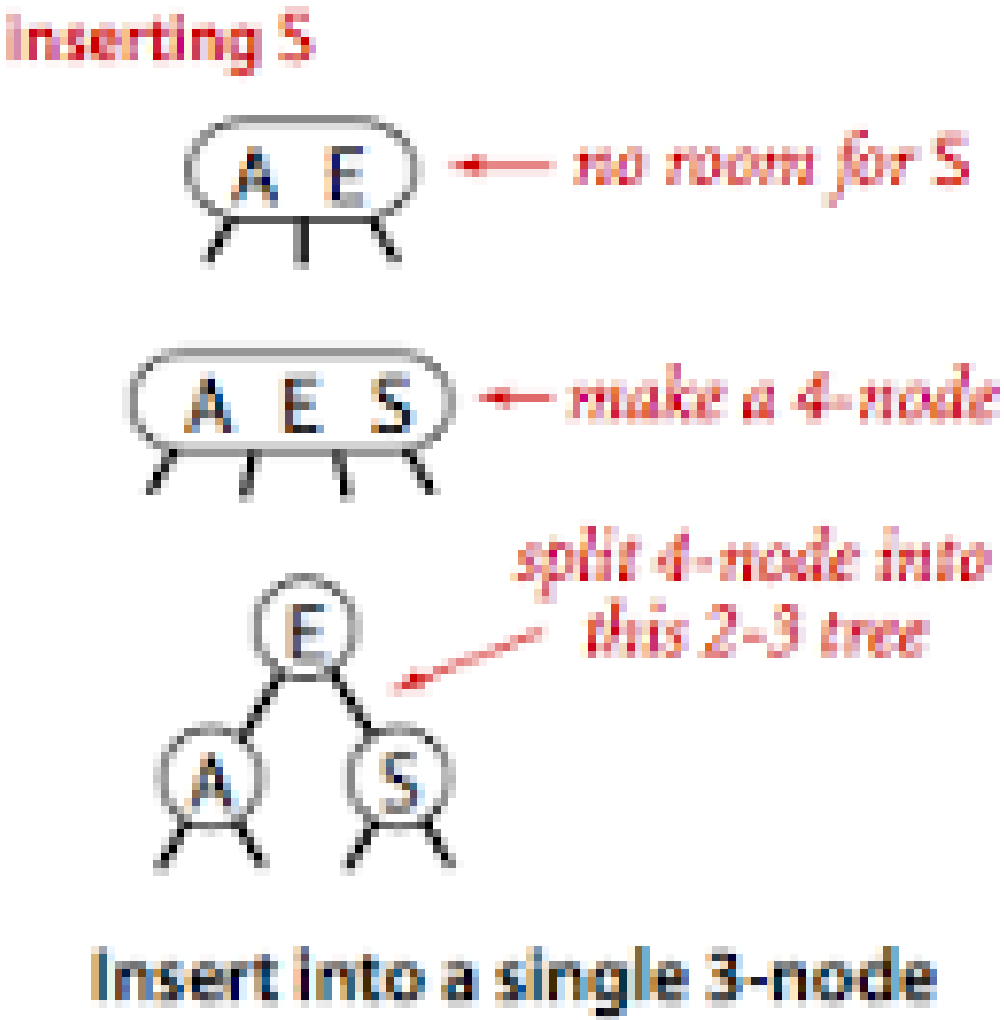
Inserting



Insert into a 2-node

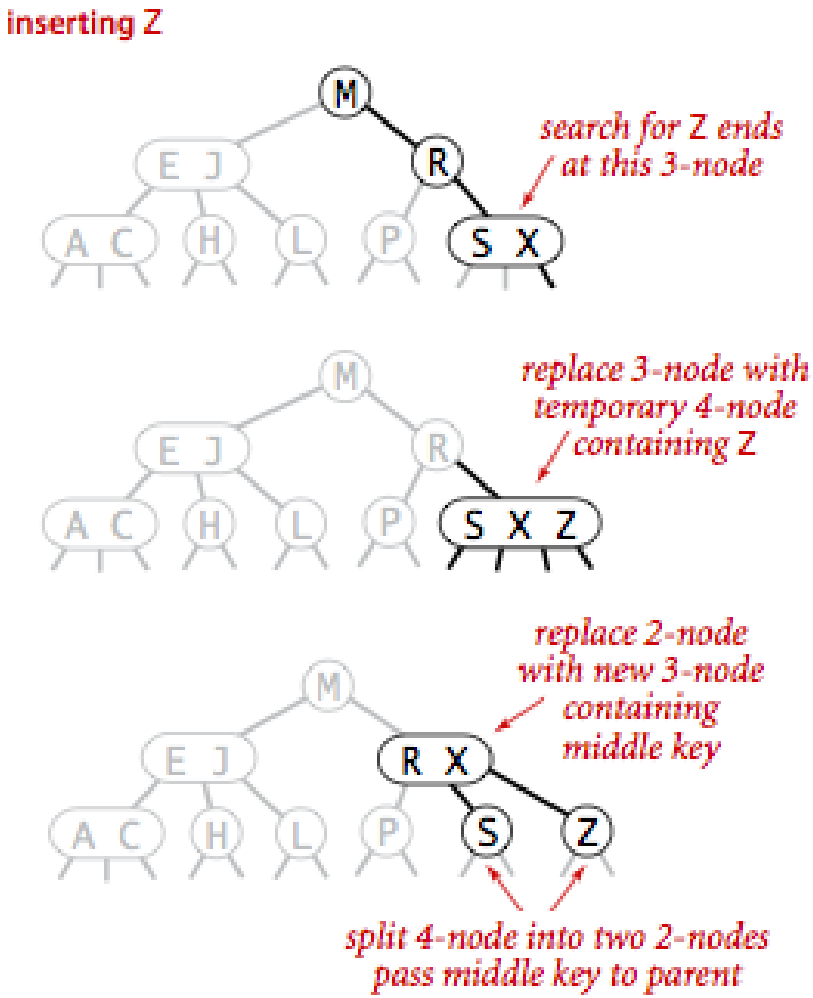
cphbusiness
COPENHAGEN BUSINESS ACADEMY

Inserting



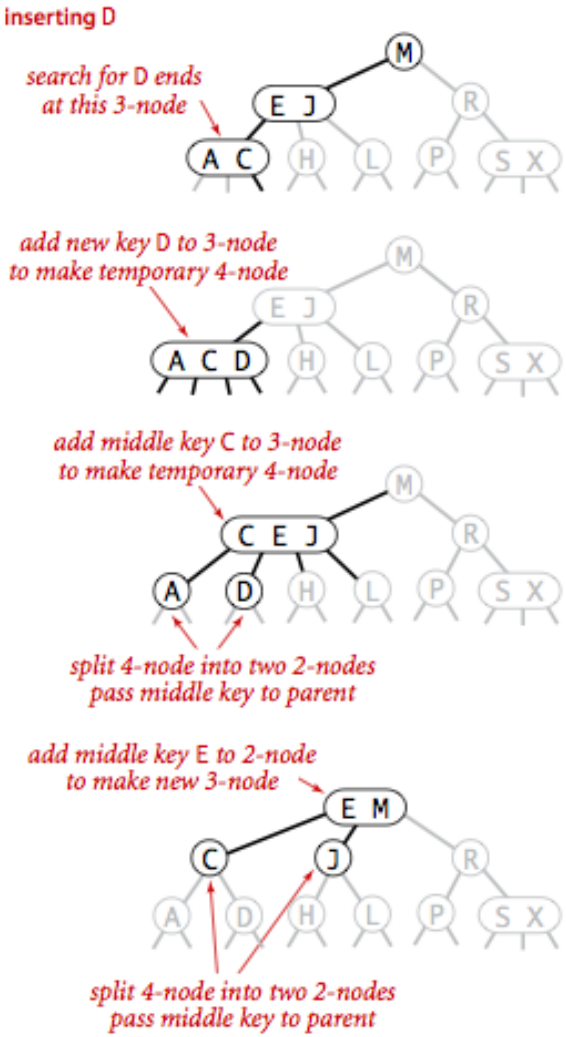
cphbusiness
COPENHAGEN BUSINESS ACADEMY

Inserting

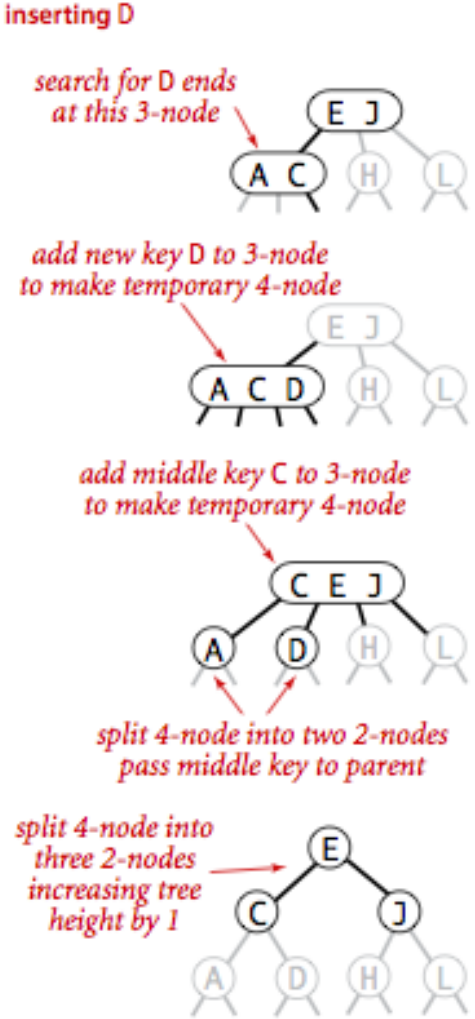


Insert into a 3-node whose parent is a 2-node

Inserting



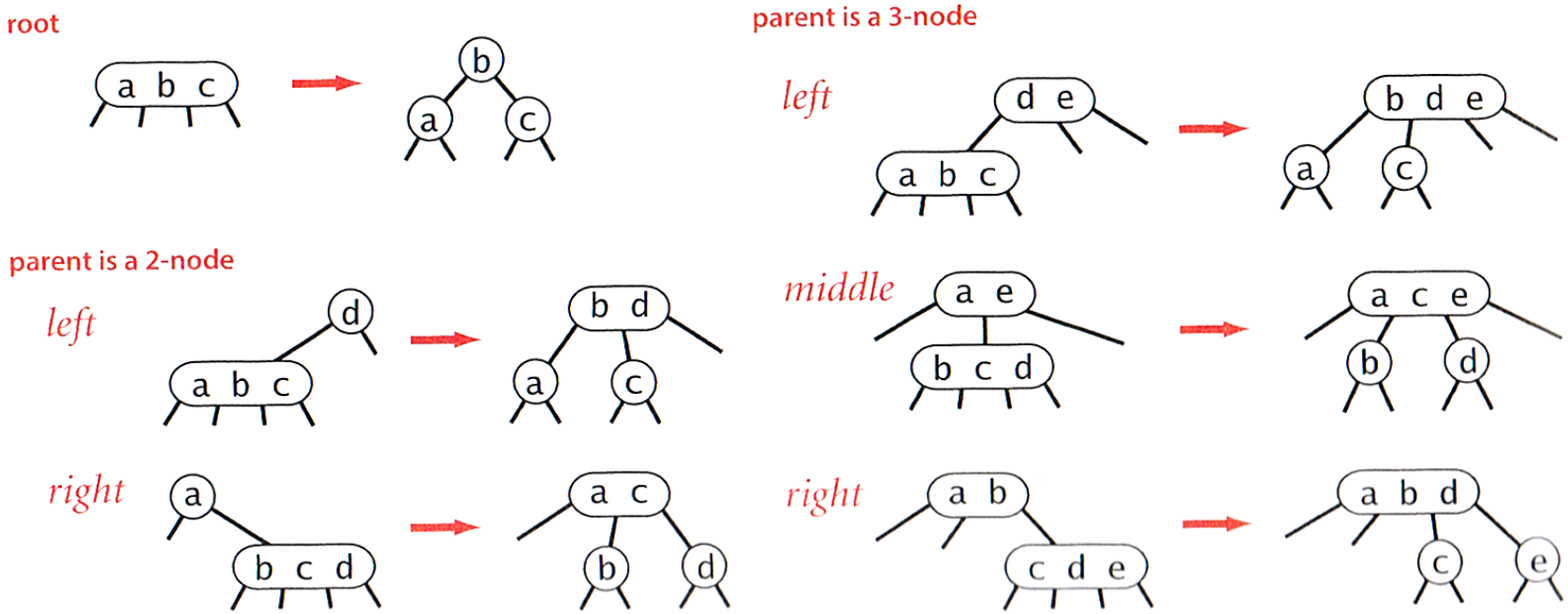
Insert into a 3-node whose parent is a 3-node



Splitting the root

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Summing up



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

Read and Write to console

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
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>
```

Show and Read

cphbusiness
COPENHAGEN BUSINESS ACADEMY

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

Reading lines from file

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
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")
    )
```

Define function


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

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

```
50! = 30414093201713378043612608166064768844377641568
      960512000000000000
```

Mutable state

cphbusiness
COPENHAGEN BUSINESS ACADEMY

```
import Data.IORef

data Tree d = Empty | Node d (Tree d) (Tree d)

main :: IO ()
main = do
  treeRef <- newIORef Empty
  writeIORef treeRef (Node 7 Empty Empty)
  tree <- readIORef treeRef
  putStrLn (show tree)
```

More information on Haskell IO

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

Assignment

Create a Haskell application that:

- Implements a balanced search tree[1] 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.

[1] or at least a binary search tree