50001 - Algorithm Analysis and Design - Lecture 13

Oliver Killane

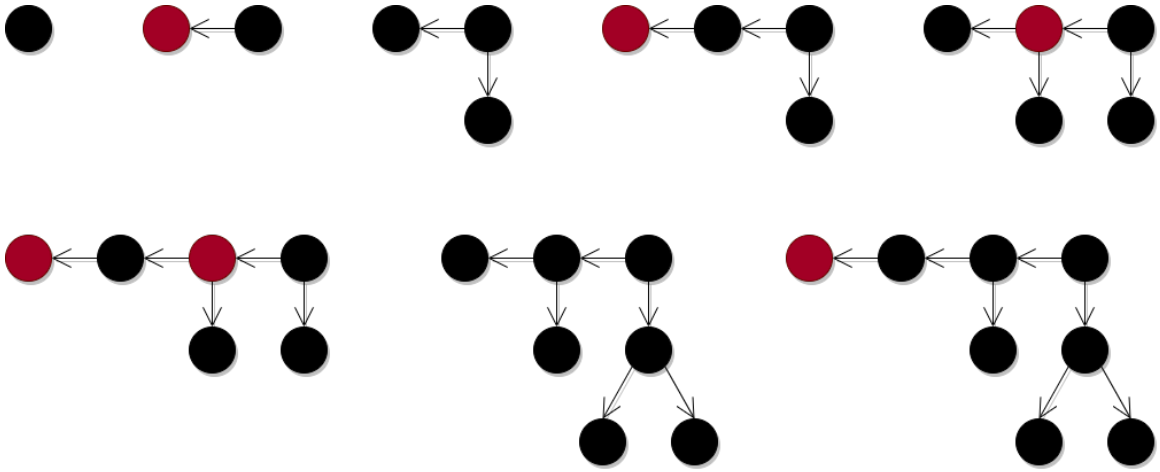
22/11/21

Lecture Recording

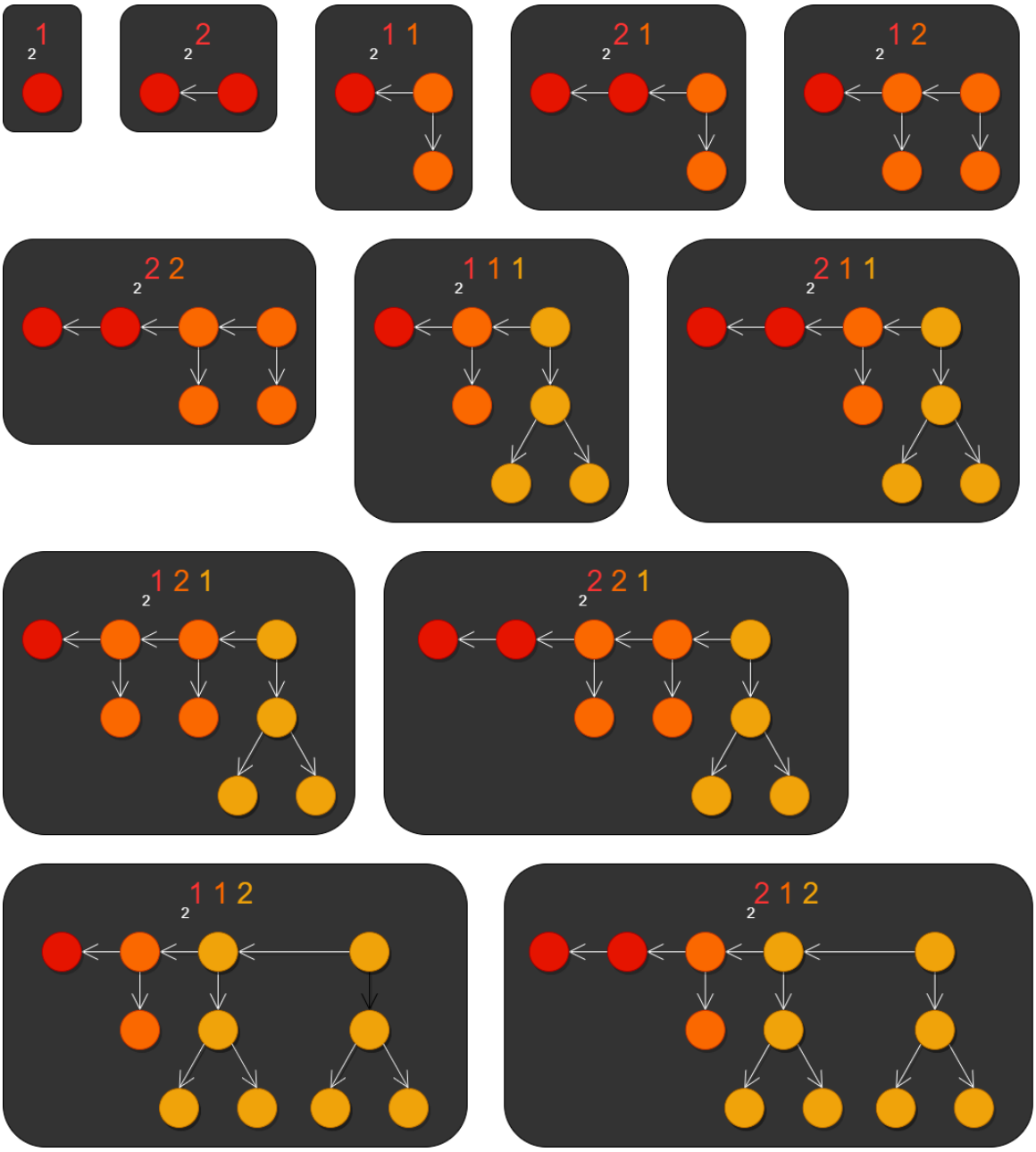
Lecture recording is available here

Red Black Trees Continued

We have a pattern with inserting elements from an ordered list into the tree.



We can encode this as a special binary number system, using 1 and 2 such that the least significant bit is a the number of trees of 2^0 nodes, and the nth is 2^n.



We can increment this by:

```haskell
-- each element is a red black tree (+ an extra root element (a))
data Digit a = One a (RBTree a) | Two a (RBTree a) a (RBTree a)

incr :: a -> RBTree a -> [Digit a] -> [Digit a]
incr x t [] = [One x t]
incr x t ((One y u) : ds) = Two x t y u : ds
incr x t ((Two y u z v) : ds) = One x t : incr y (Node Black u z v) ds
```

We can convert a list of digits back to a red black tree by:

```
-- fold left to combine the digits together into a tree
fromList :: [a] -> RBTree a
fromList xs = foldl link Empty (foldr add xs)

link :: RBTree a -> Digit a -> RBTree a
link l (One x t) = Node Black l x t
link l (Two x t y u) = Node Black (Node Red l x t) y u
```