G54 AAD

L08 Persistence & Binary Search Trees

This lecture
PFDS ch 1 & 2

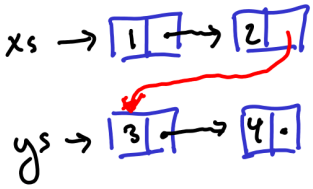
Functional Data Structures are Persistent

Persistent Data Structure – Operations do not modify data structure

Ephemeral Data Structure – Operations modify data structure

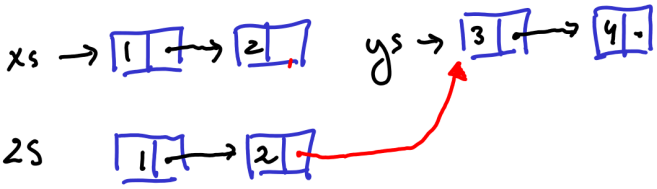
Ex Append of Linked List xs and ys: zs = append xs ys

Ephemeral



xs modified

Persistent



xs and ys not modified
(zs shares data with ys)

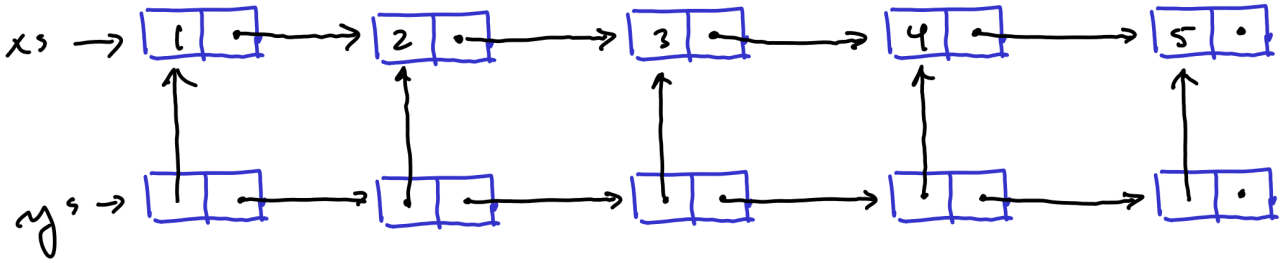
```
myAppend :: [a] -> [a] -> [a]
myAppend []     ys = ys
myAppend (x:xs) ys =  x : (myAppend xs ys)
```

↑ New cons!

Example of Sharing

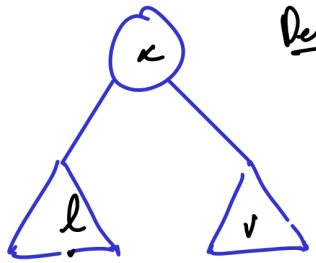
Sharing can lead to huge savings!

```
suffixes :: [a] -> [[a]]
suffixes []     = [[]]
suffixes ys@(x:xs) = ys : (suffixes xs)
```




```
xs = [1..5]
ys = suffixes xs
```

Binary Search Trees

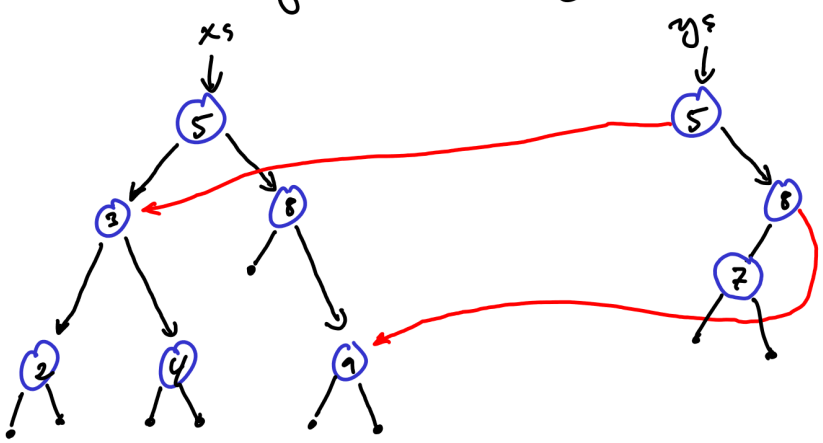


Def: A binary search tree is either
- a leaf node or
- a branch node containing an element x and two binary search trees l and r such that
  1) no element in l is larger than x
  2) ——"—— r is smaller than x.

```
data BSTree a = L | B a (BSTree a) (BSTree a)

treeMember :: (Ord a) => a -> BSTree a -> Bool
treeMember x L = False
treeMember x (B y l r)
    | x == y = True
    | x < y  = treeMember x l
    | x > y  = treeMember x r
```

Sharing in Binary Search Trees




```
xs = B 5 (B 3 (B 2 LL) (B 4 LL))
     (B 8 L (B 9 LL))

ys = insert 7 xs
```

```
insert :: (Ord a) => a -> BSTree a -> BSTree a
insert x L = B x LL
insert x (B y l v)
    | x < y = B y (insert x l) v
    | x > y = B y l (insert x v)
```

Sorting with Binary Search Trees

```
flatten :: BSTree a -> [a]
flatten L = []
flatten (B x l v) = (flatten l) ++ [x] ++ (flatten v)

treeSort = flatten o (foldr insert L)
```

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f x [y] = f y x
foldr f x (y:ys) = f y (foldr f x ys)
```