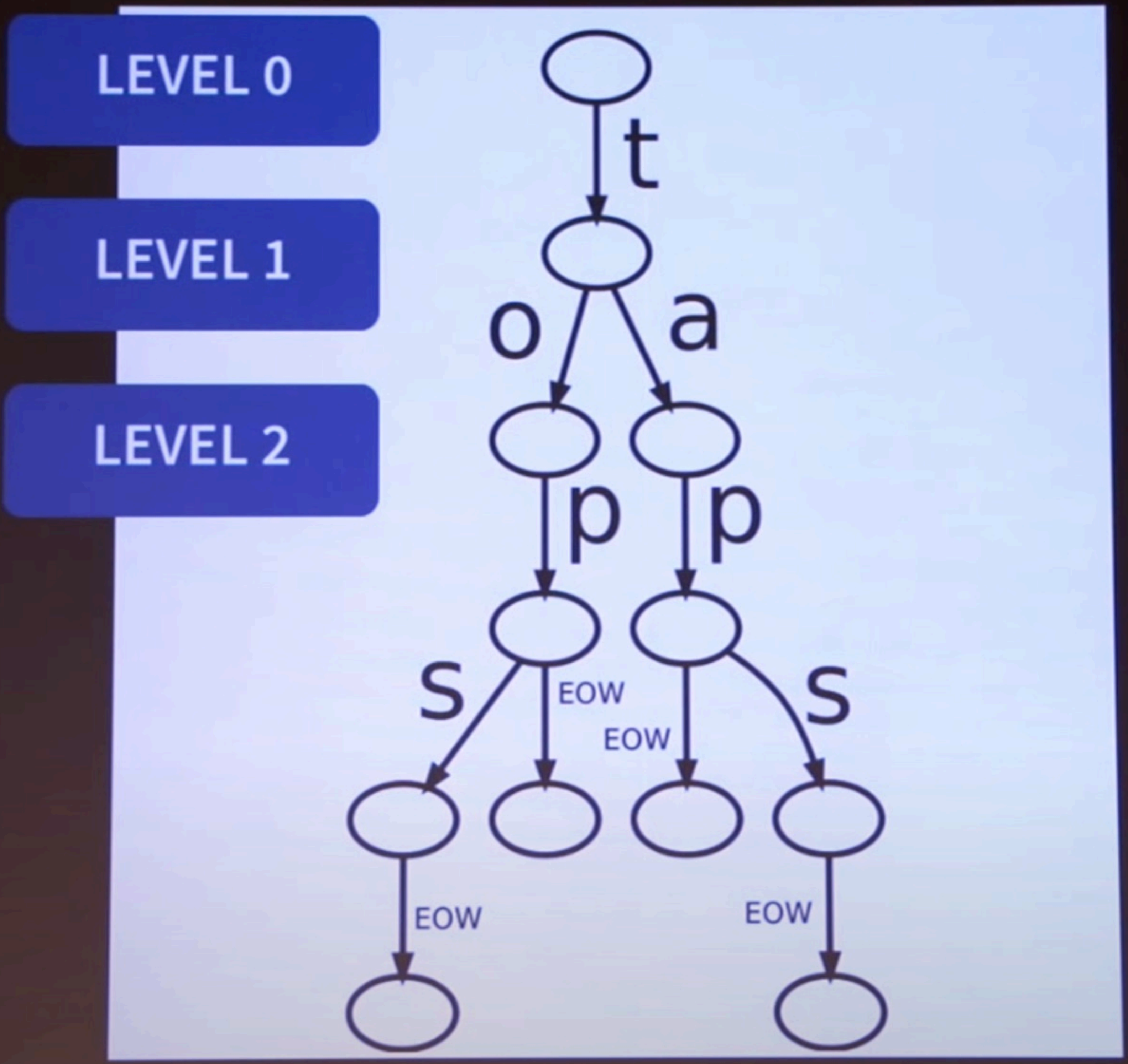
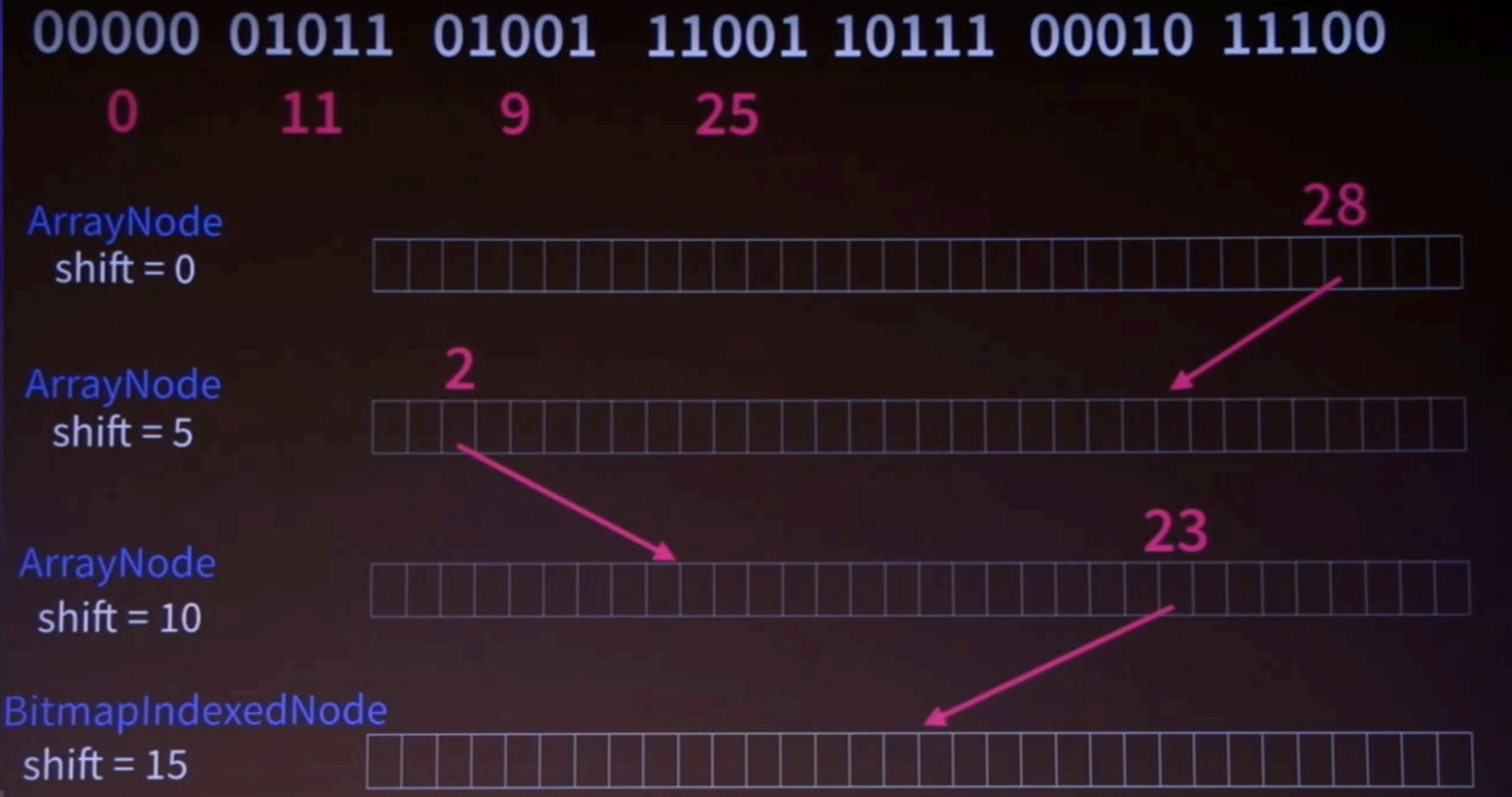
Persistent Data Structures

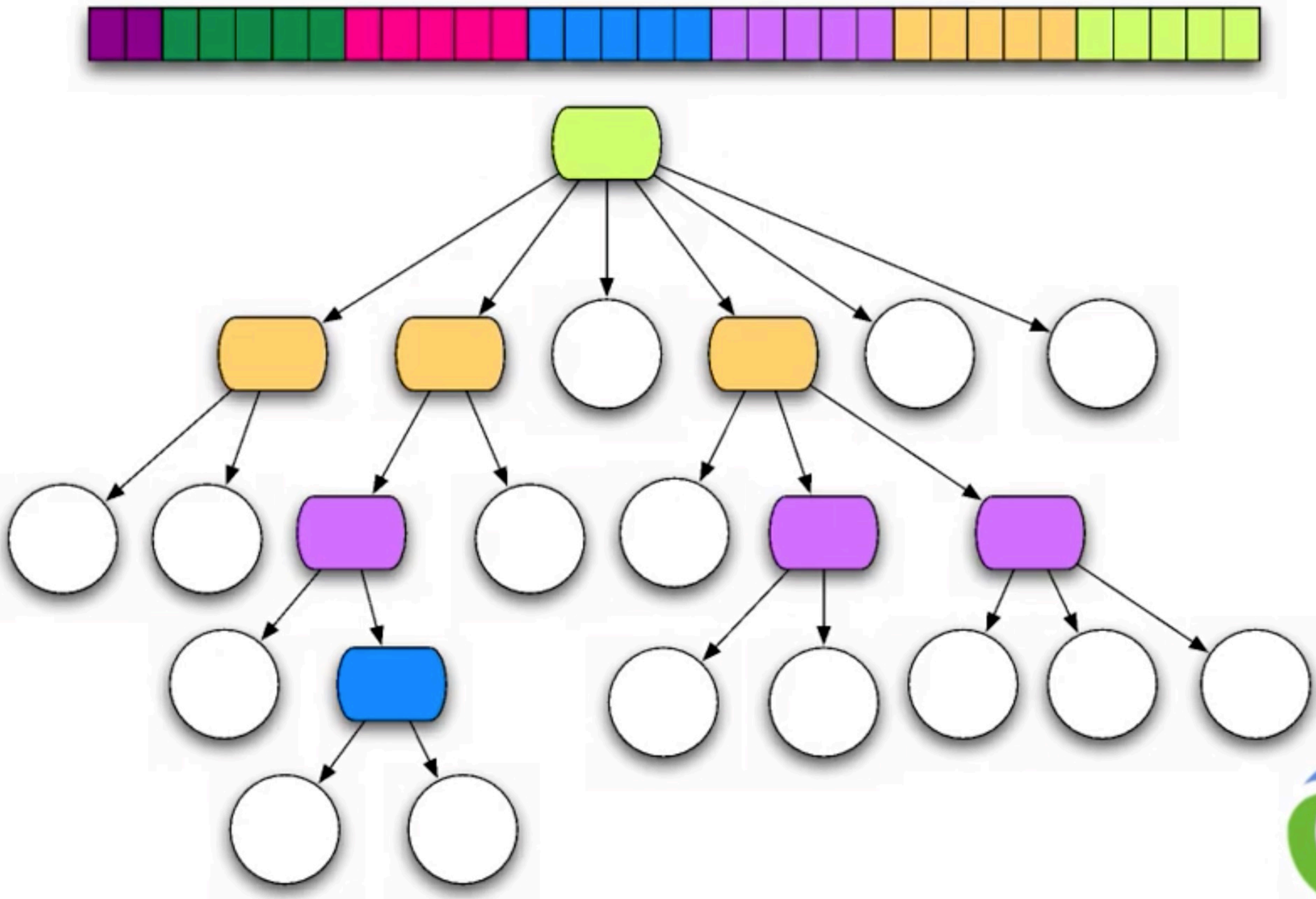
- Composite values - immutable
- 'Change' is merely a function, takes one value and returns another, 'changed' value
- Collection maintains its performance guarantees
  - Therefore new versions are not full copies
- Old version of the collection is still available after 'changes', with same performance
- Example - hash map/set and vector based upon array mapped hash tries (Bagwell)


LEVEL 0
LEVEL 1
LEVEL 2
t
o
a
p
p
s
EOW
EOW
s
EOW
EOW



… and then follow the AMT down

Bit-partitioned hash tries

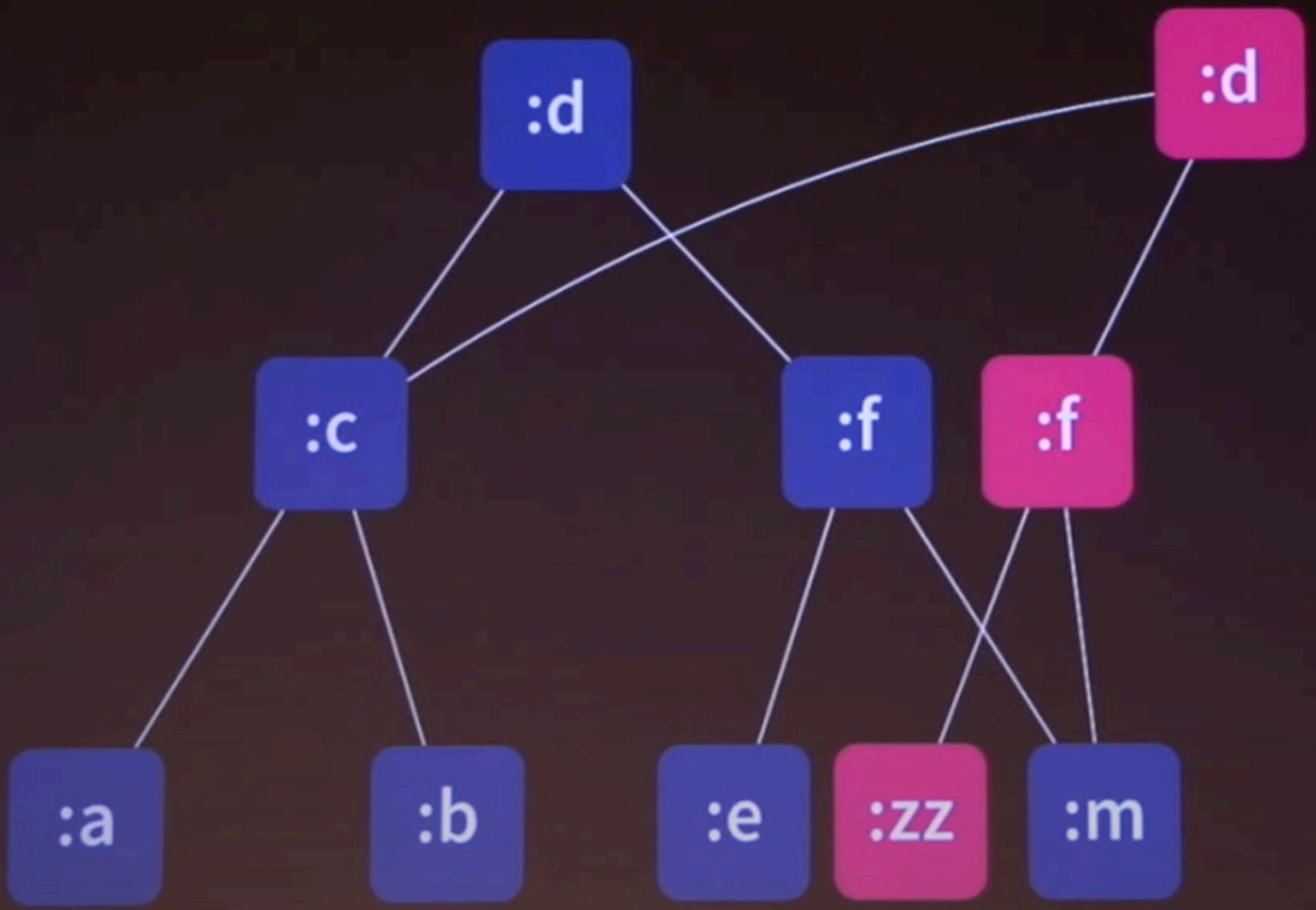


Structural Sharing

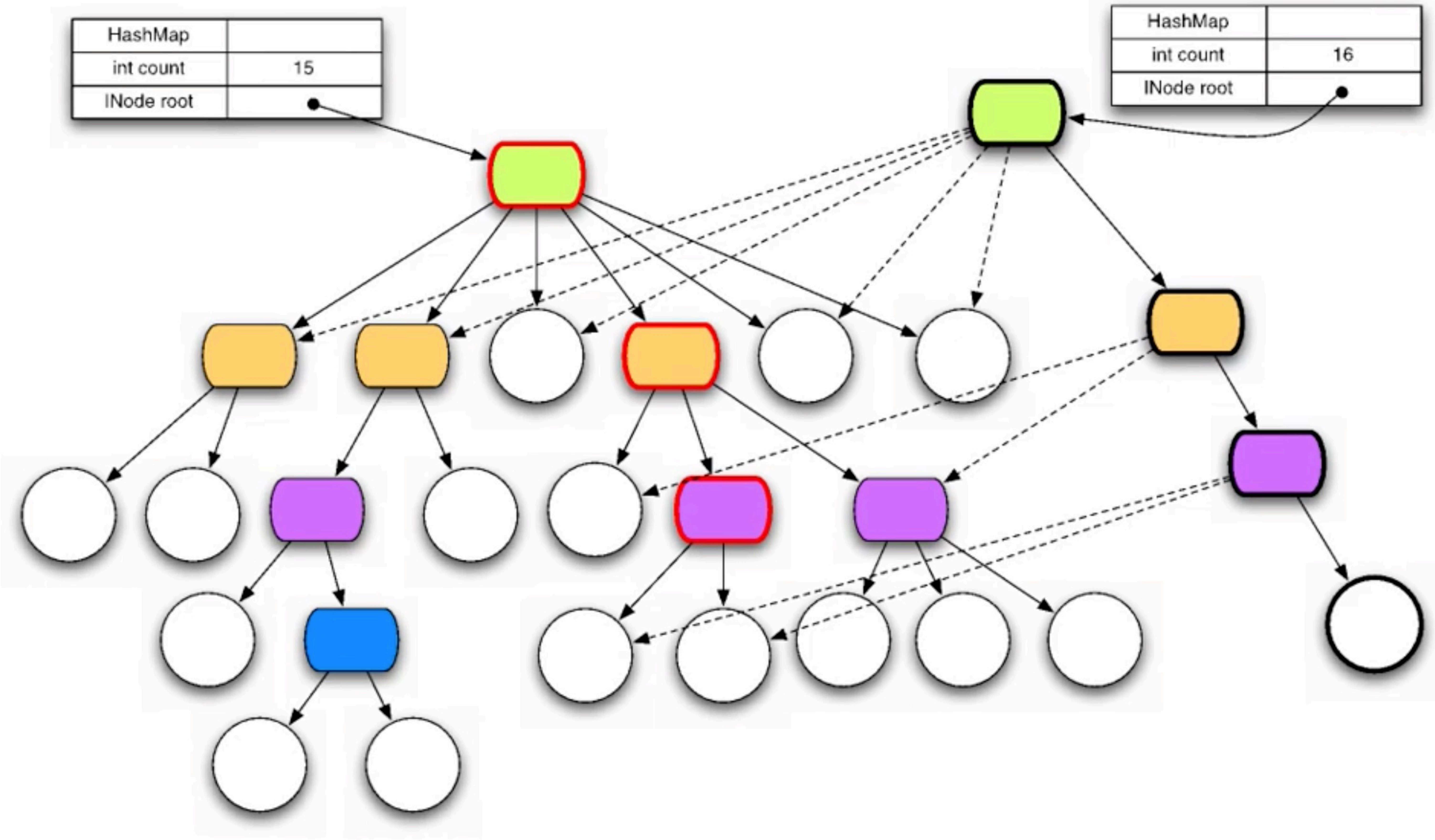
- Key to efficient 'copies' and therefore persistence
- Everything is immutable so no chance of interference
- Thread safe
- Iteration safe

STRUCTURAL SHARING

(assoc v 4 :zz)



Path Copying

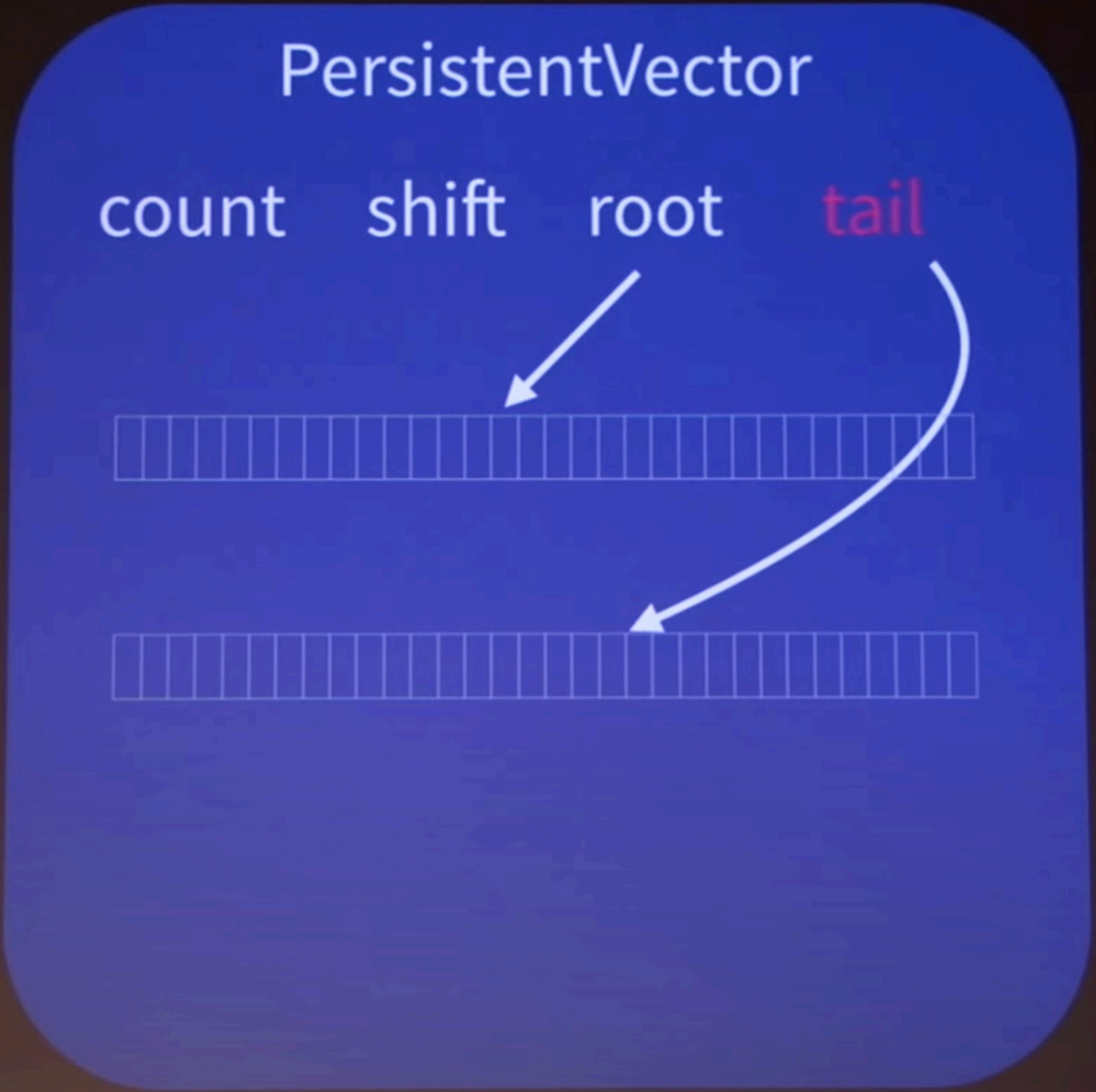


REGULAR HASH TABLE?

NEED ROOT RESIZING

NOT AMENABLE TO STRUCTURAL SHARING

THE TAIL OPTIMIZATION



Slides are taken from, the seminars:

1. What Lies Beneath - A Deep Dive Into Clojure's Data Structures - Mohit Thatte
2. Persistent Data Structures and Managed References - Rich Hickey