List
Vector
Lazy sequence

Vee Satayamas

20170408

Clojurian’s meetup in Bangkok

Agenda

- Persistent data structure
- Cons cell
- Singly linked list
- Search tree
- Persistent vector
- Lazy sequence

Persistent data structure

Persistent data structure

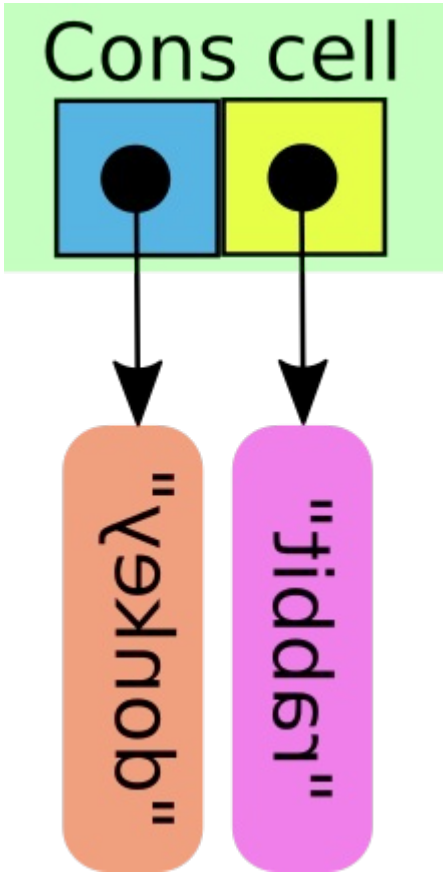
“In computing, a persistent data structure is a data structure that always preserves the previous version of itself when it is modified.”

https://en.wikipedia.org/wiki/Persistent_data_structure

How about array?

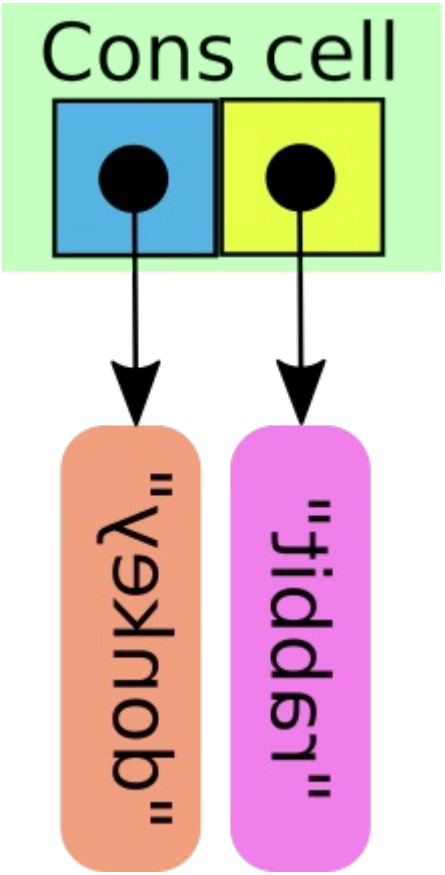
Cons cell

Cons cell



array of size 2

Cons cell (2)



(cons “donkey” “rabbit”)

Cons cell (3)

(car (cons “donkey” “rabbit”))

donkey

Cons cell (4)

(cdr (cons “donkey” “rabbit”))

“rabbit”

Clojure or Common Lisp?

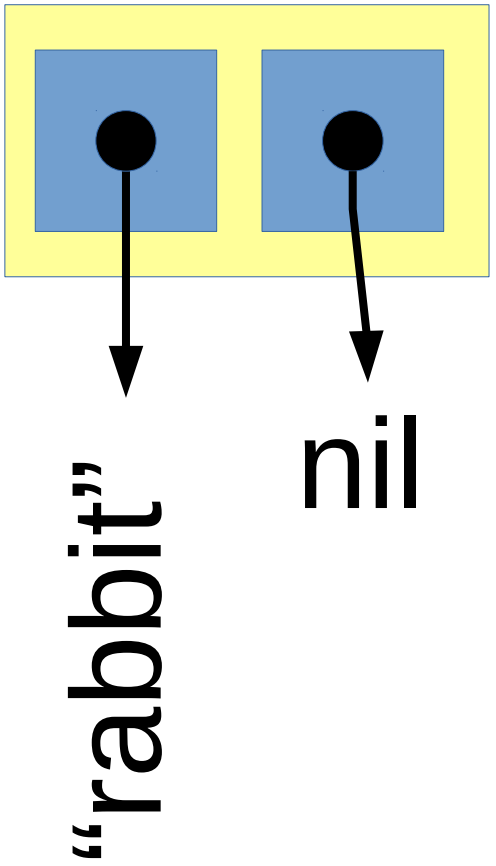
Singly linked list

LISP – LISt Processor

Singly-linked list

(list “rabbit”)

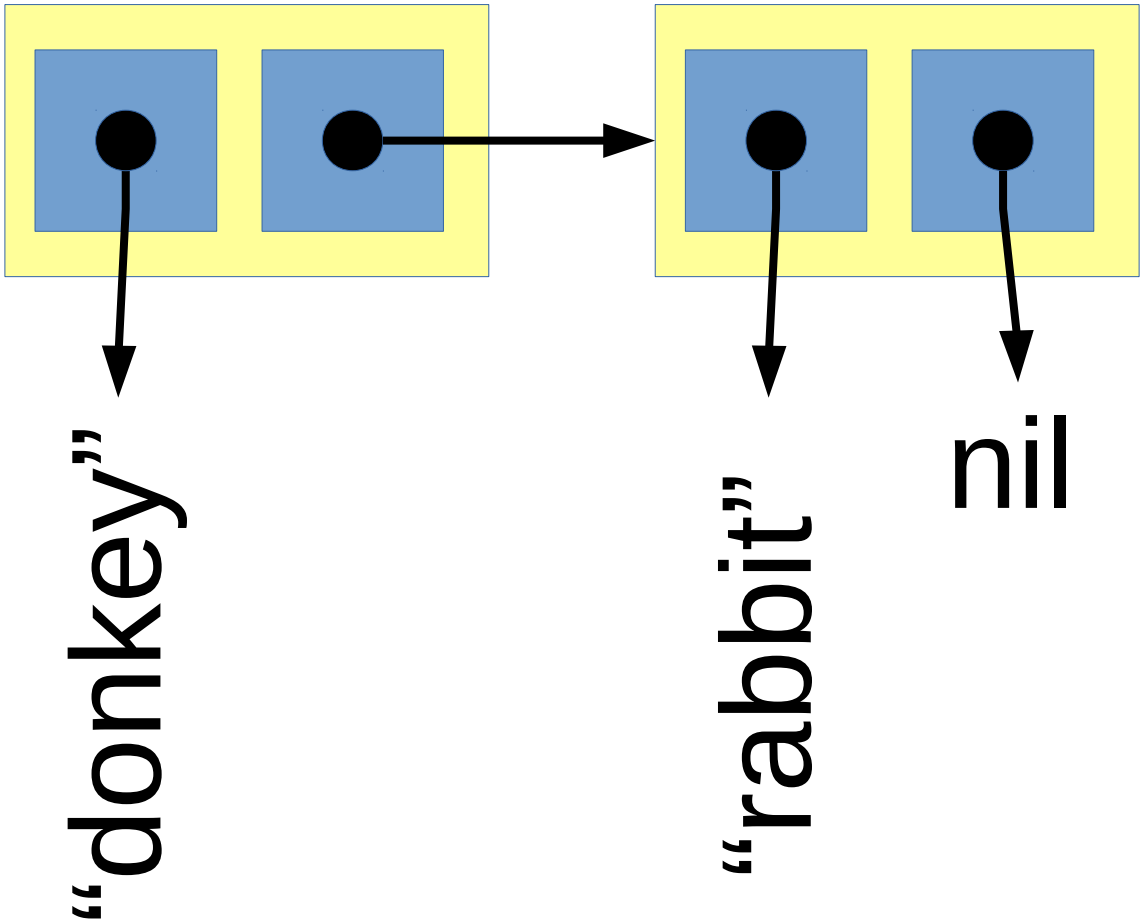
(cons “rabbit” nil)



Singly-linked list (2)

(list “donkey” “rabbit”)

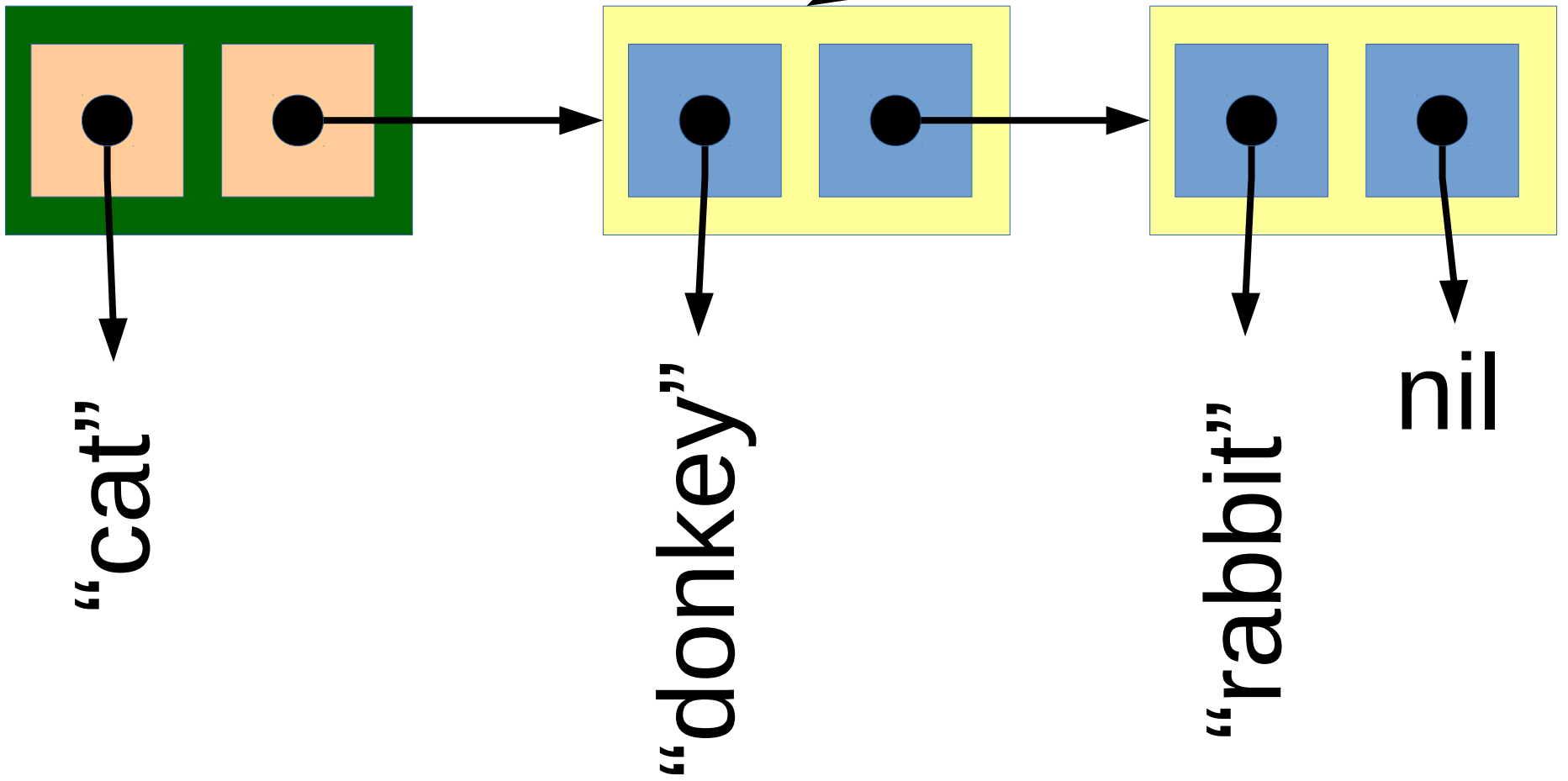
(cons “donkey”
(cons “rabbit”
nil))



Singly linked list

(def a (list “donkey” “rabbit”))

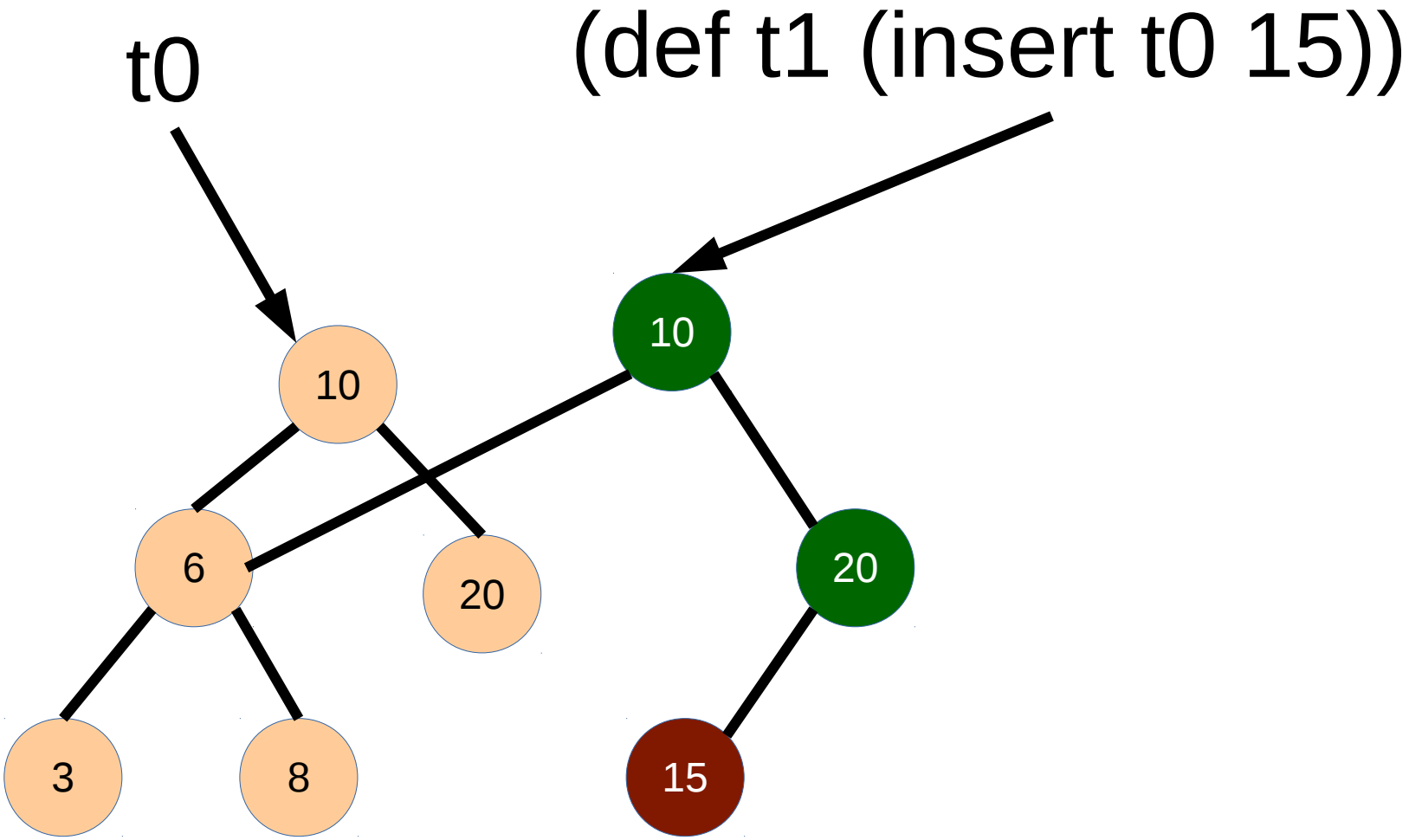
(def b (cons “cat” a))



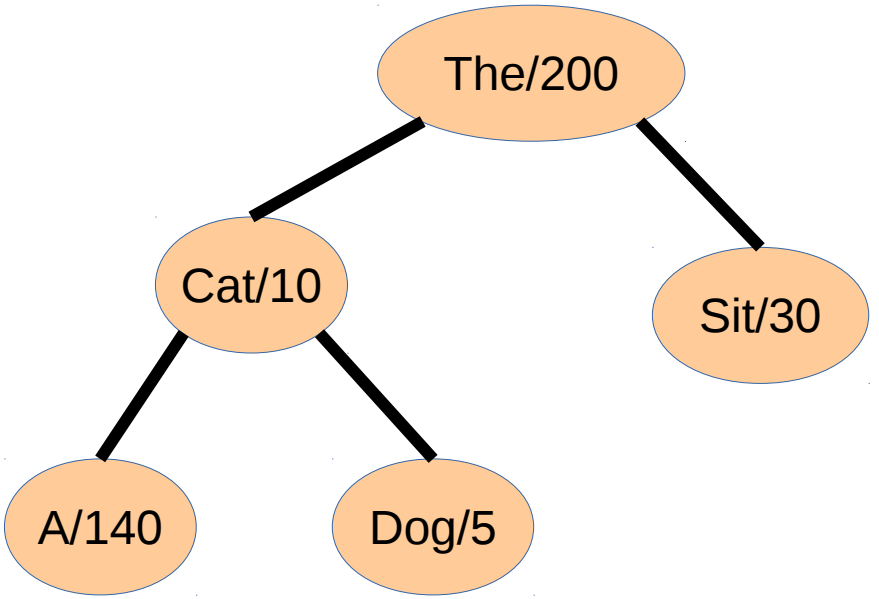
How about time complexity?

Search tree

Search tree

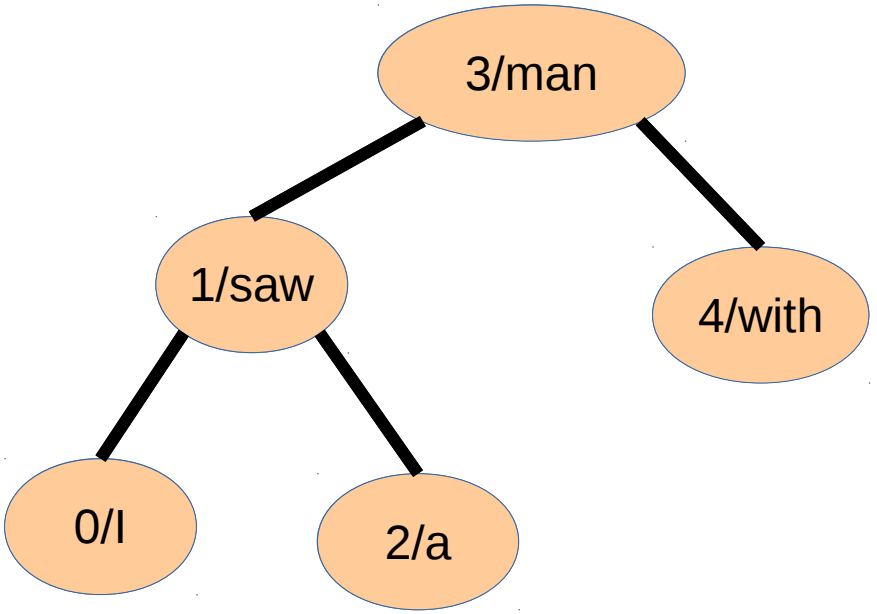


Search tree (2) key-value map?



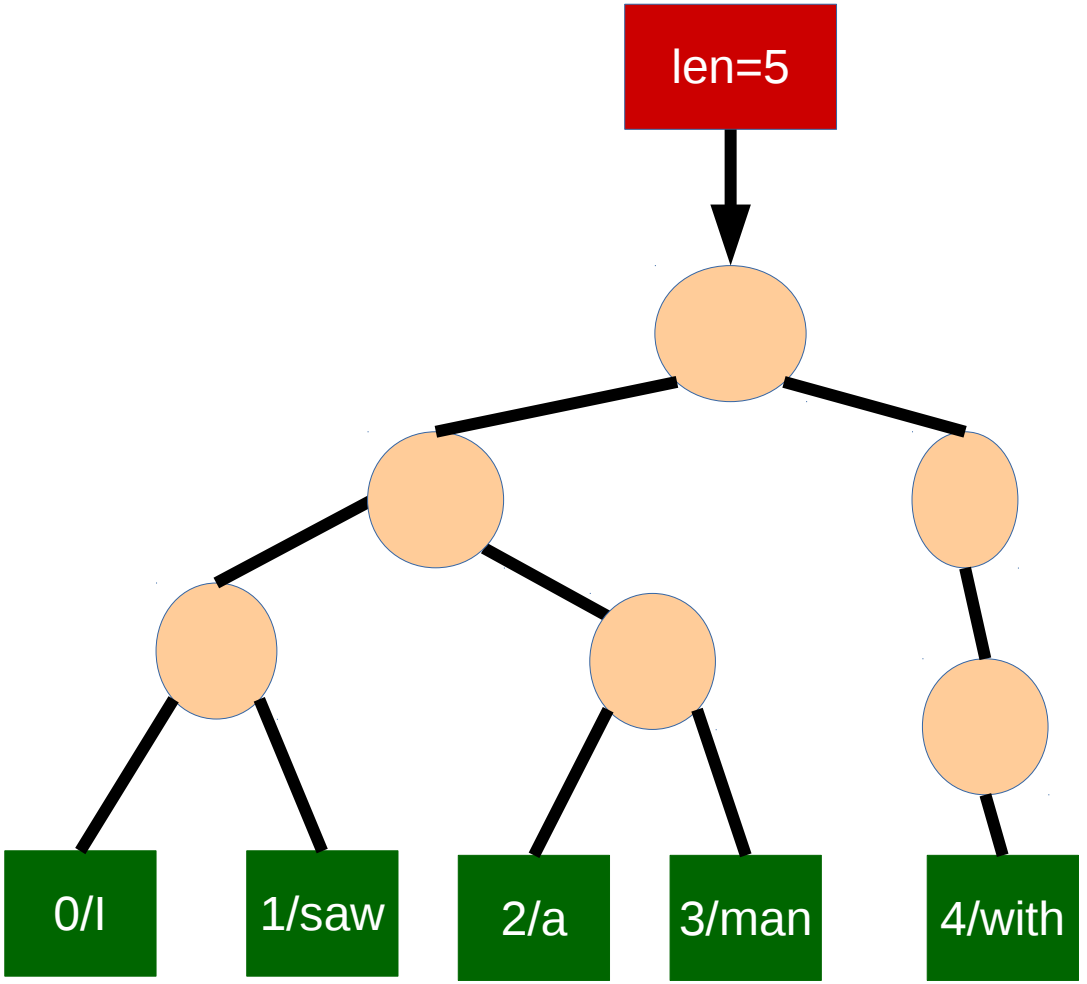
How about time complexity?

Search tree (3) vector?



Persistent vector

Persistent vector (simplified-)



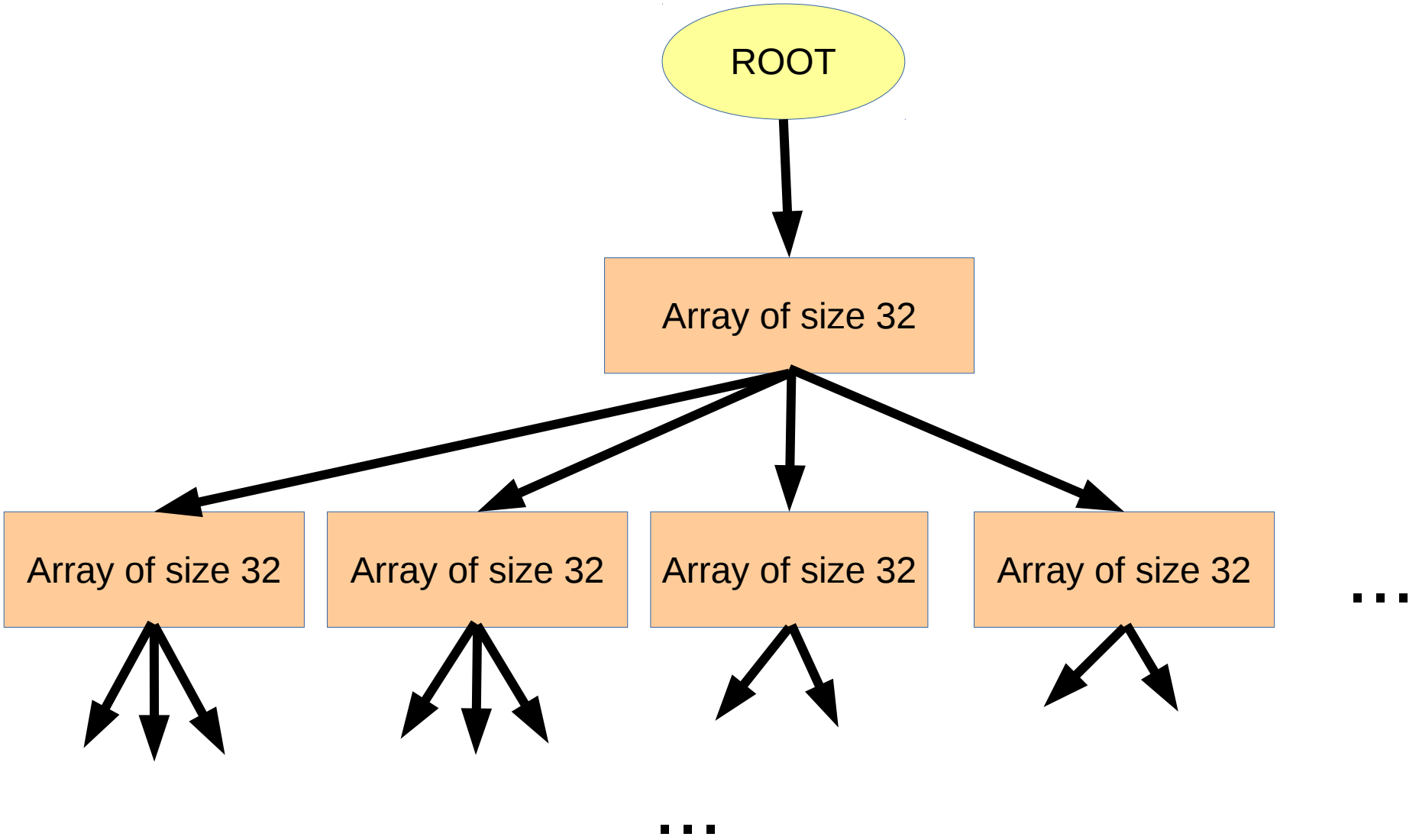
Persistent vector (2)

$Log_2 n$

vs

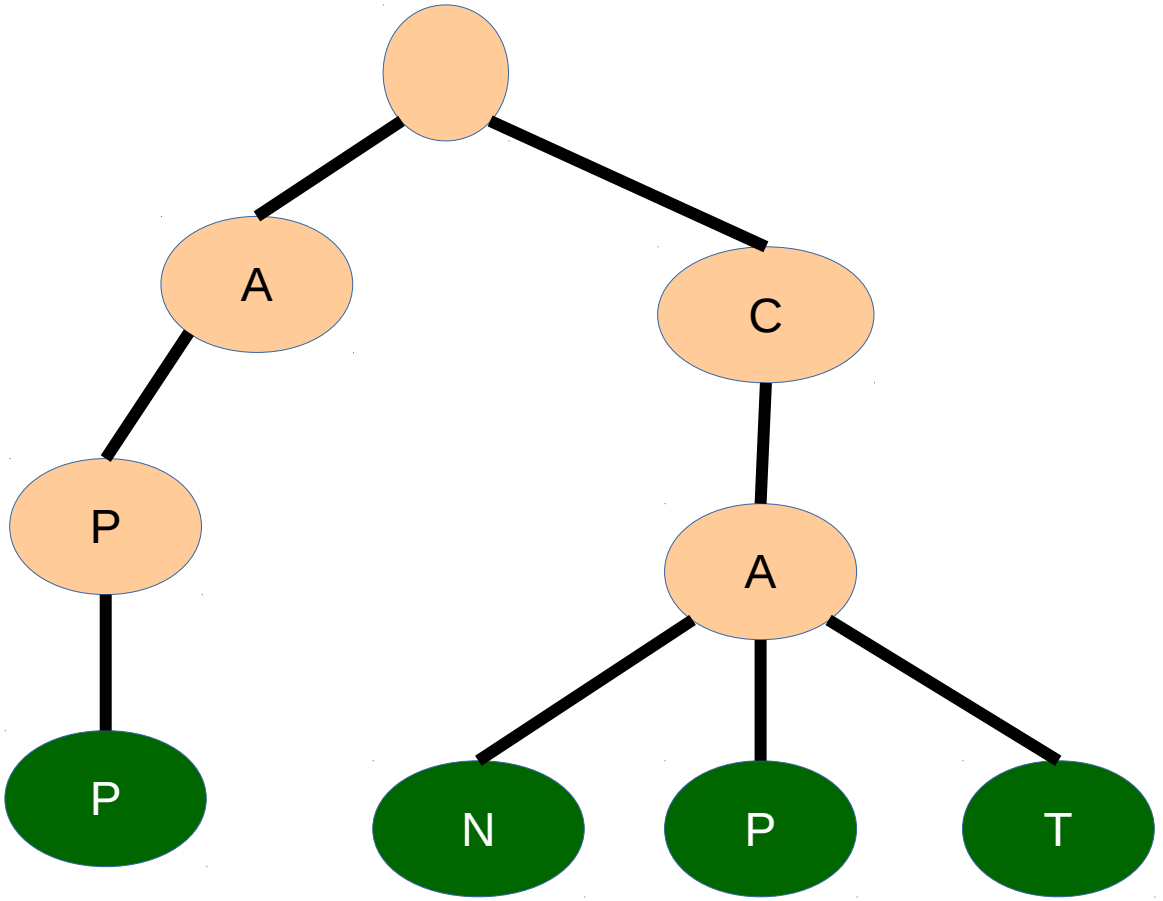
$Log_{32} n$

Persistent vector (3)



Persistent vector (4)

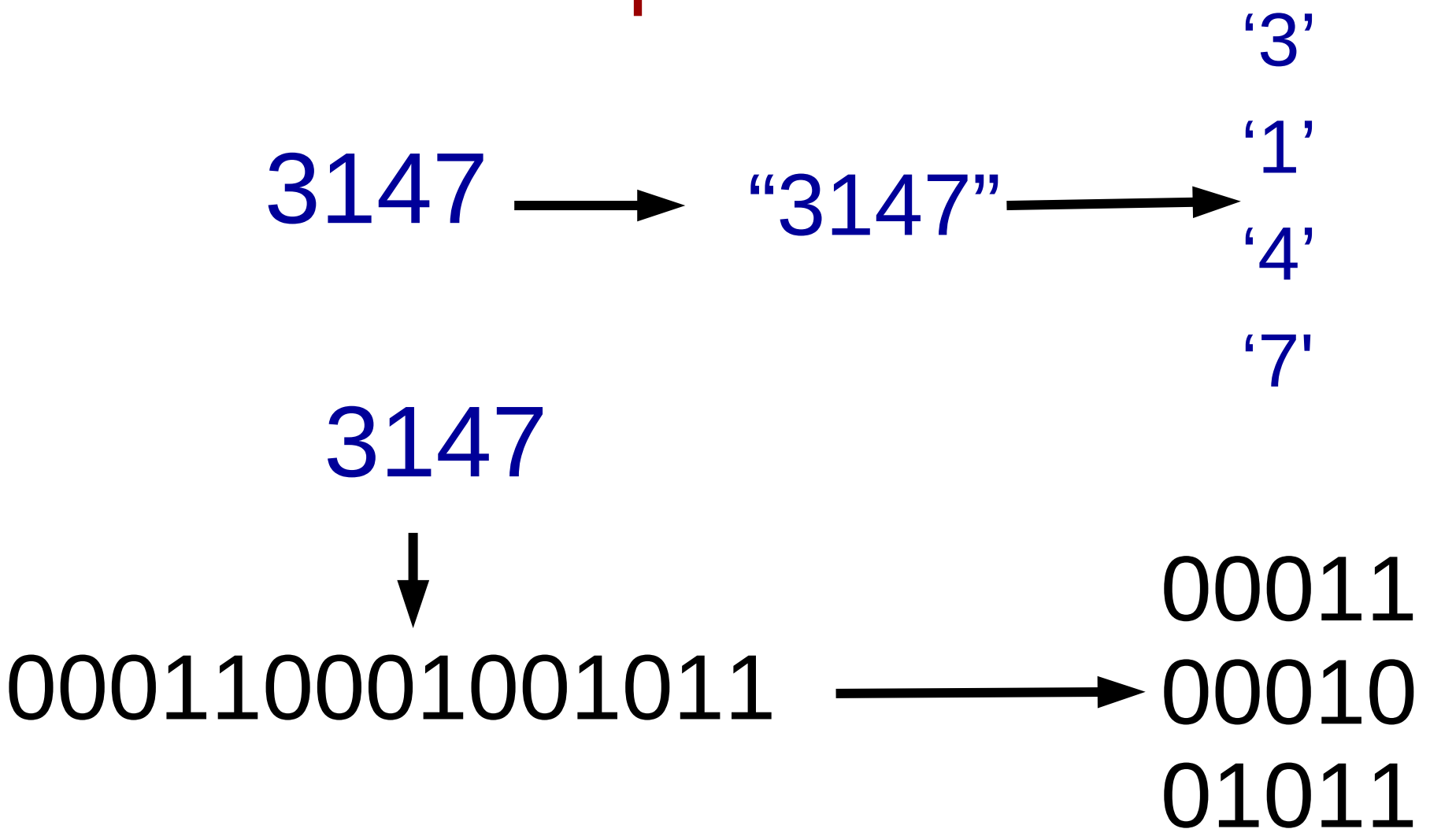
- Prefix tree:
  - CAT
  - CAN
  - CAP
  - APP



Persistent vector (5)

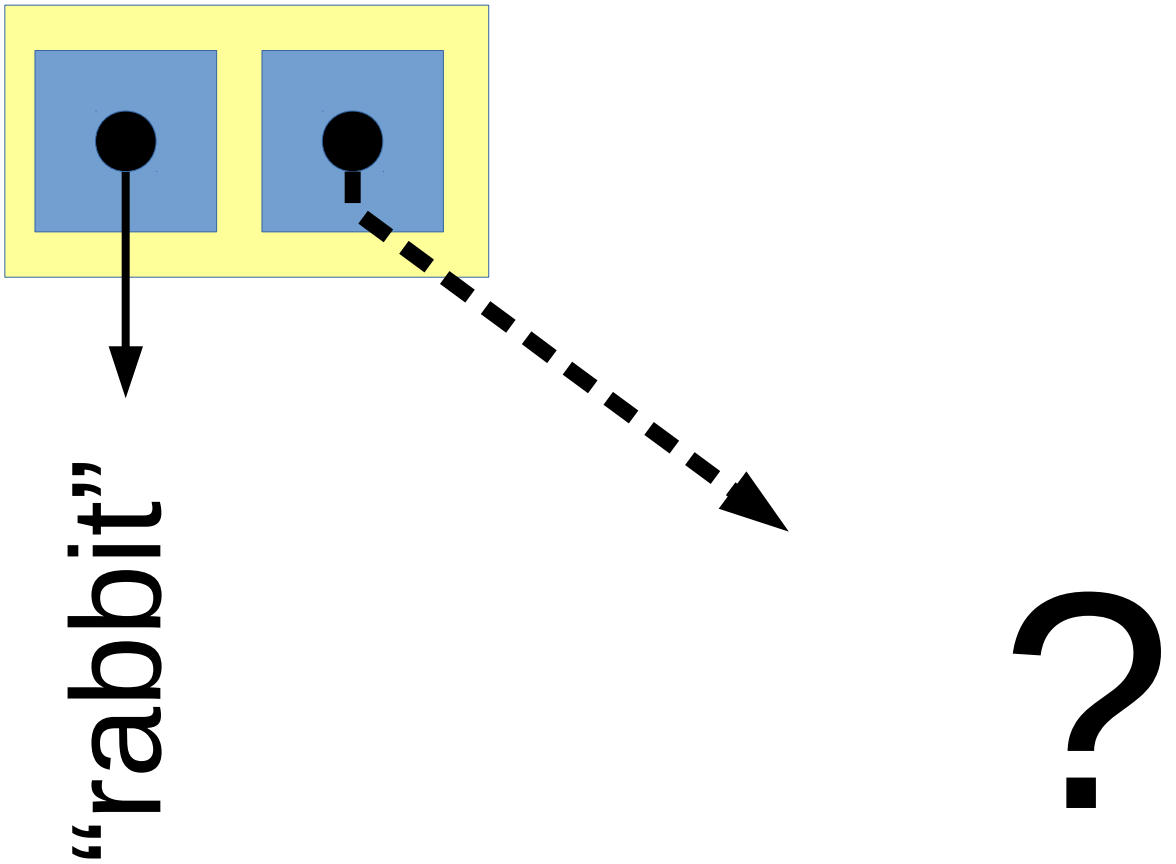
Prefix tree of indexes

Persistent vector (6)
prefix



Lazy sequence

Lazy sequence



Lazy sequence (2)

Time complexity?

(nth lazy-seq-0 180000)

Lazy sequence (3)

```
(map #(+ % 1) [1 2 3 5])
```

Lazy sequence (4)

Iterator?

Thank you

@vsatayamas