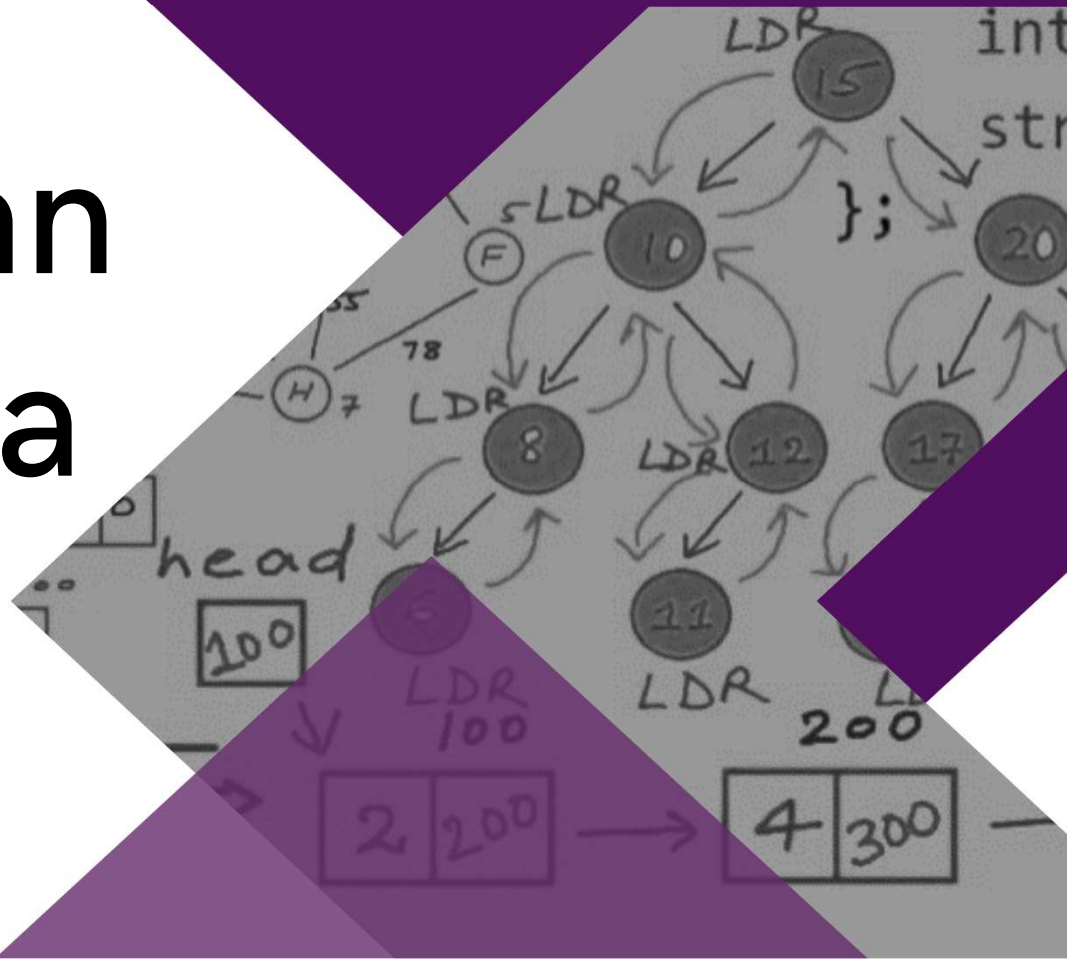
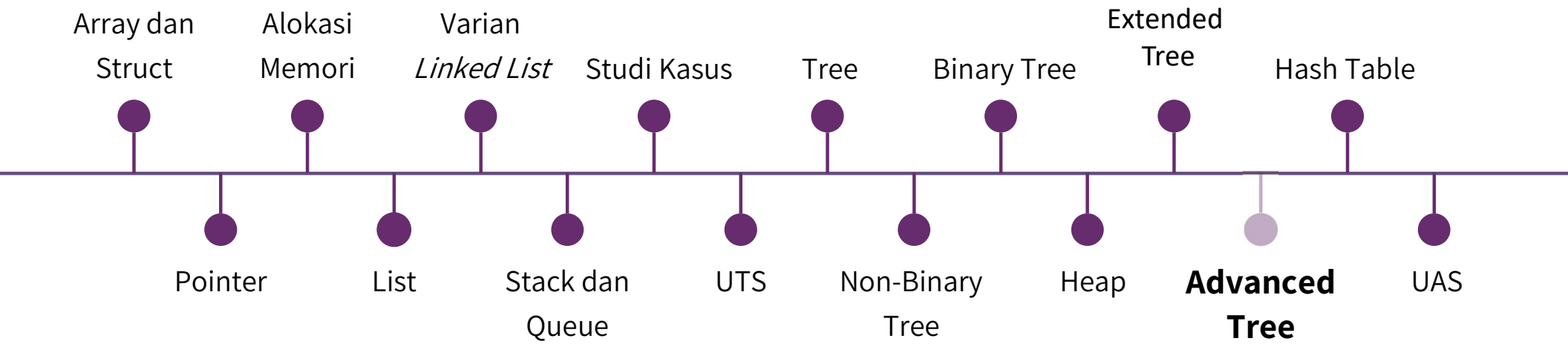
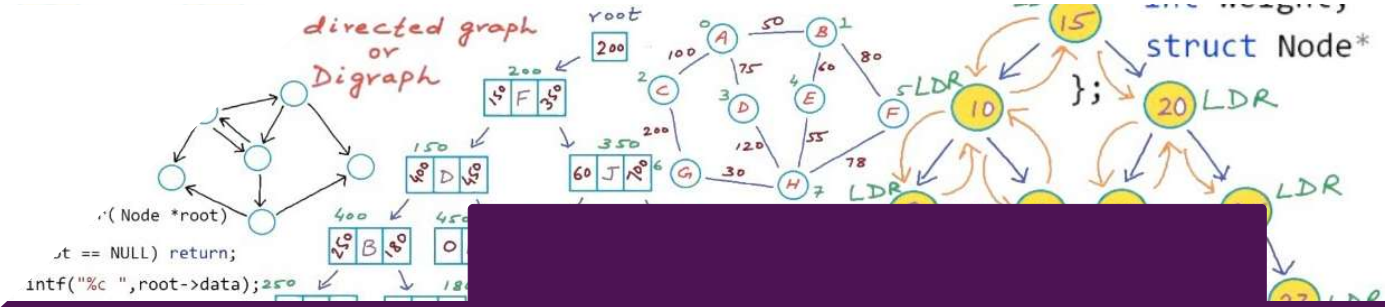
Algoritma dan Struktur Data



Pekan 14


Array dan Struct
Alokasi Memori
Varian *Linked List*
Studi Kasus
Tree
Binary Tree
Extended Tree
Hash Table
Pointer
List
Stack dan Queue
UTS
Non-Binary Tree
Heap
Advanced Tree
UAS

Tujuan



1	Mahasiswa memahami tree yang digunakan untuk aplikasi khusus.
2	Mahasiswa memahami cara kerja trie dan balanced tree
3	Mahasiswa mampu mengimplementasikan trie dan balanced tree.

Object Space Decomposition

- The decomposition of the key range is driven by the objects (i.e., the key values of the data records) stored in the tree.
- The shape of the tree will depend on the order of key insertion.
- Example: BST

Key Space Decomposition

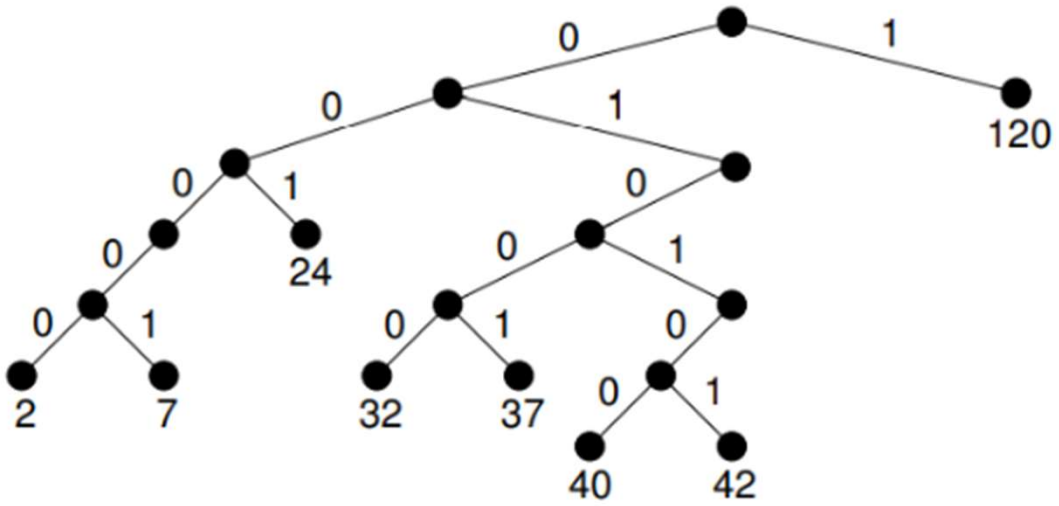
- Splitting based on predetermined subdivisions of the key range (the root could be predefined to split the key range into two equal halves, regardless of the particular values or order of insertion for the data records)
- The shape of the tree will not depend on the order of key insertion.
- The depth of the tree will be limited by the resolution of the key range.
- Example: Trie

Key Space Decomposition

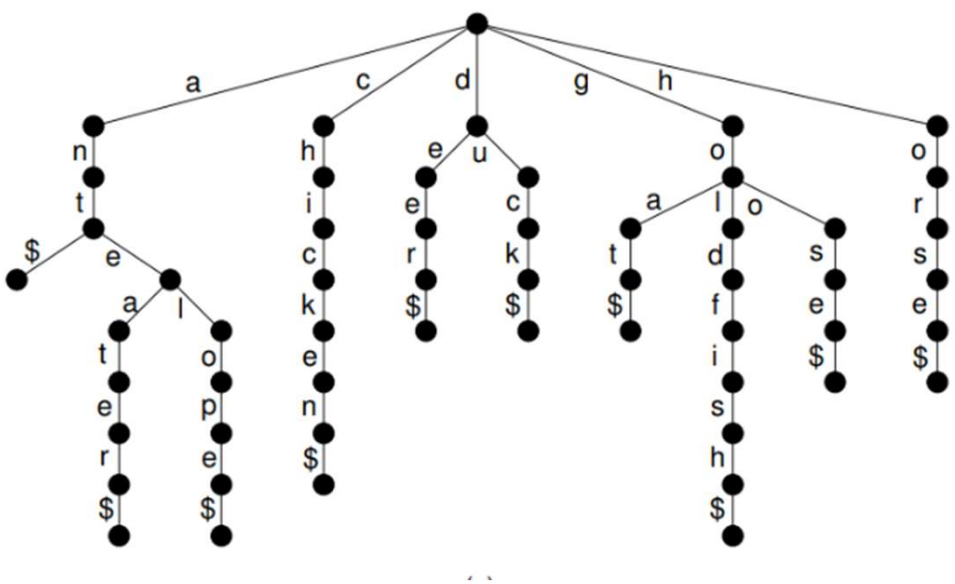
- If the keys are integers in the range 0 to 1023
  - The resolution for the key is log 1023 = 10 bits.
  - As a result, the tree will never be more than ten levels deep.
  - In contrast, a BST containing n records could be as much as n levels deep.

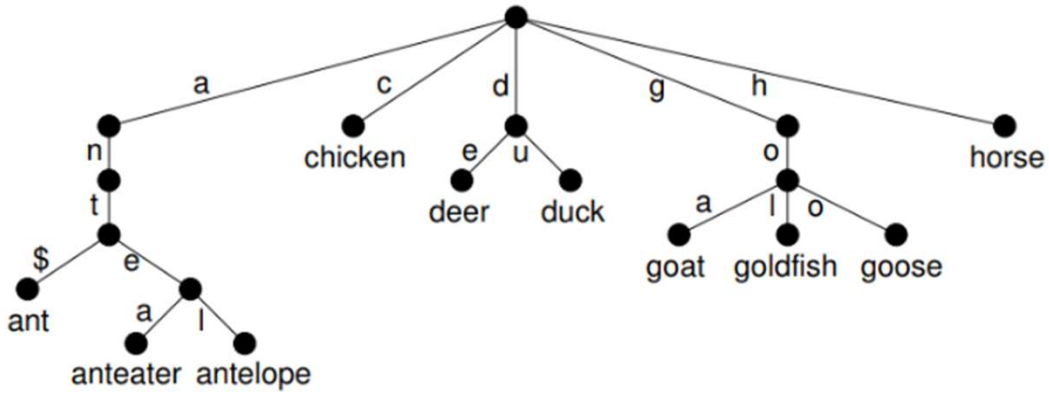
Trie

Binary Trie

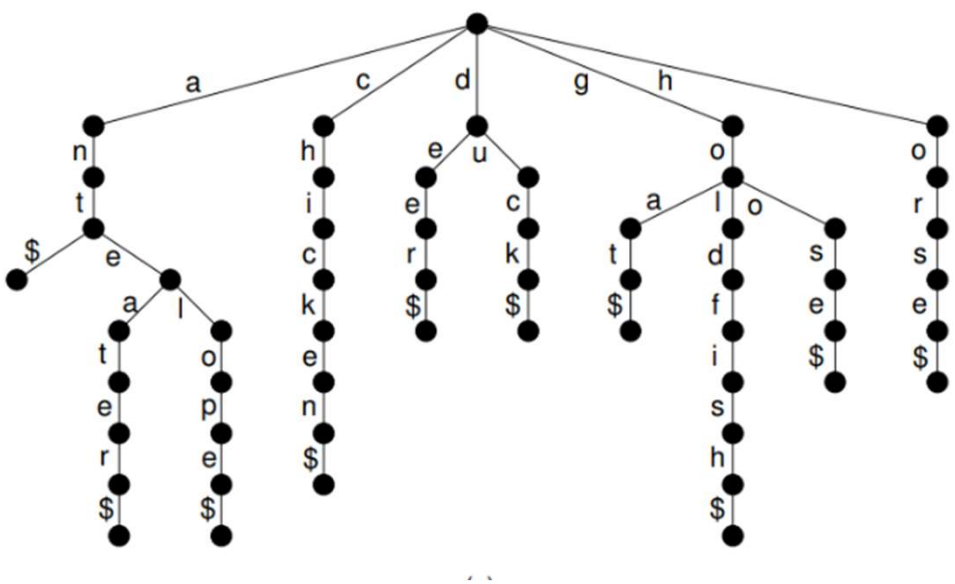


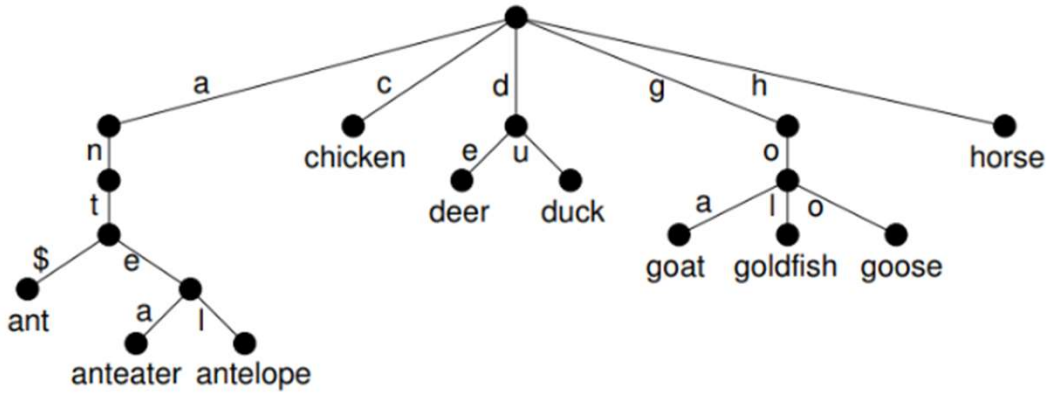
Alphabet Trie



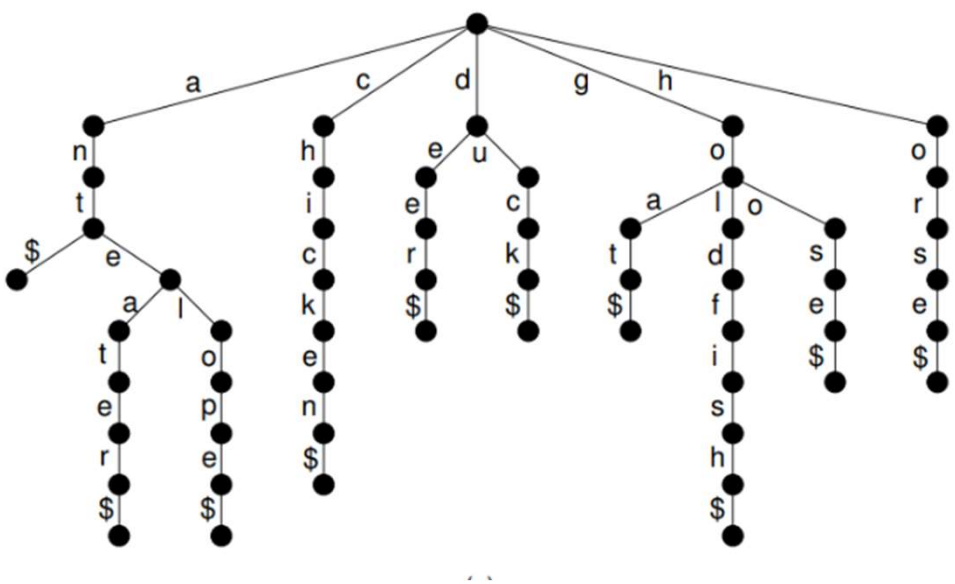


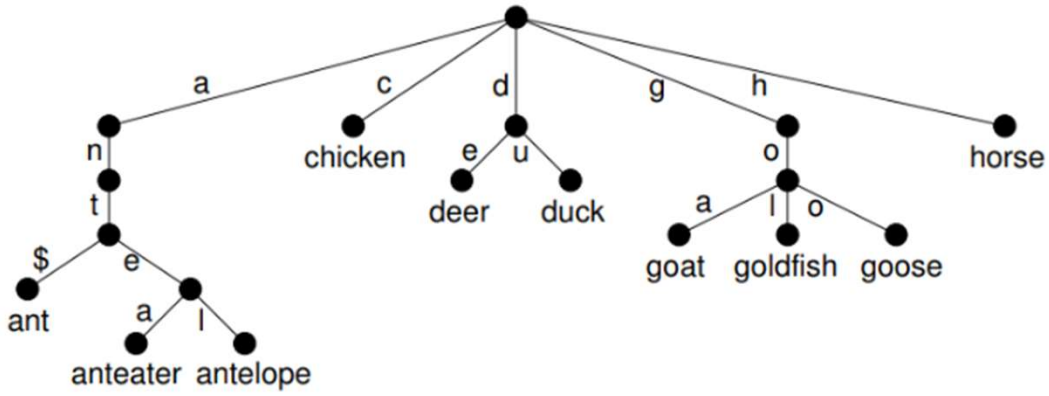
Alphabet Trie





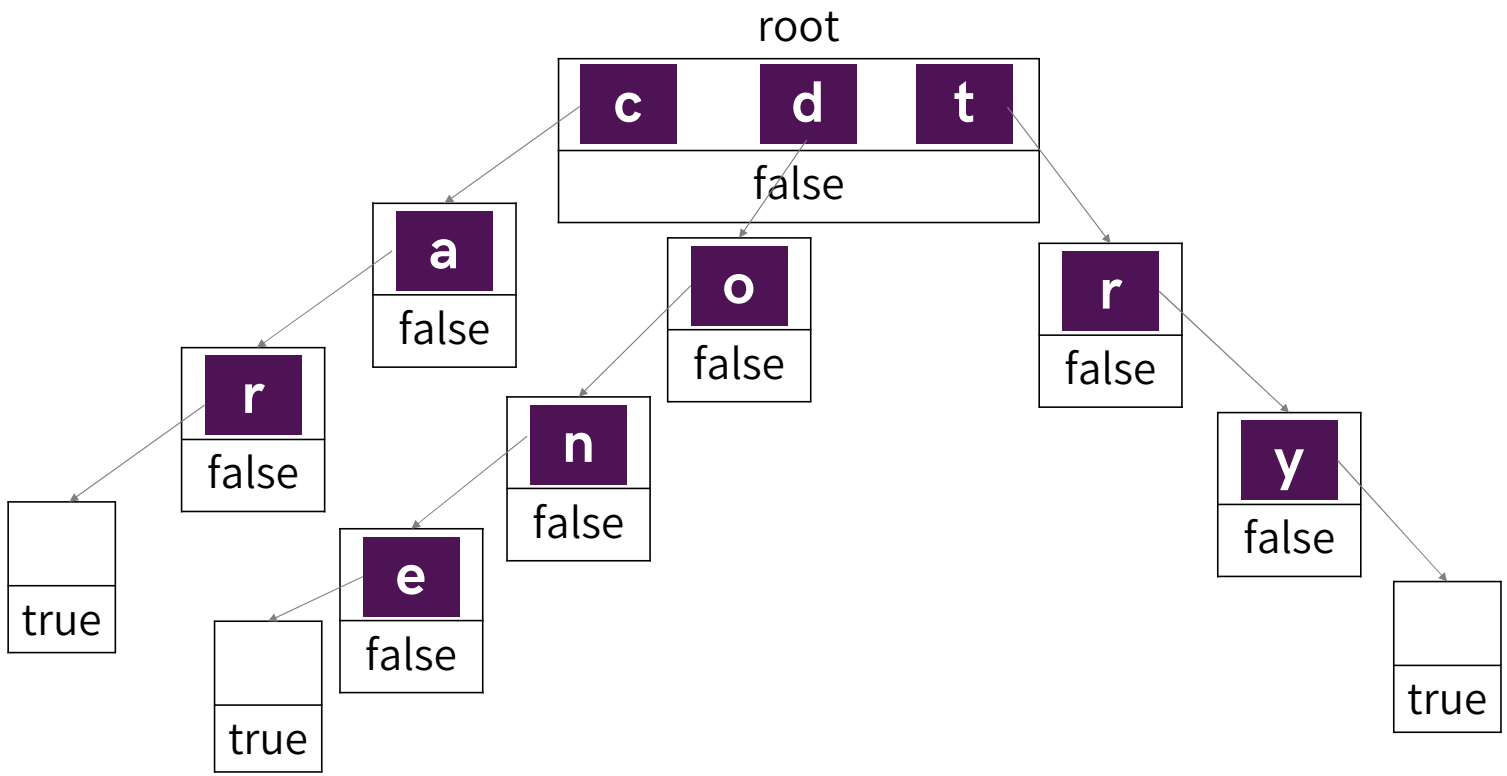
Alphabet Trie





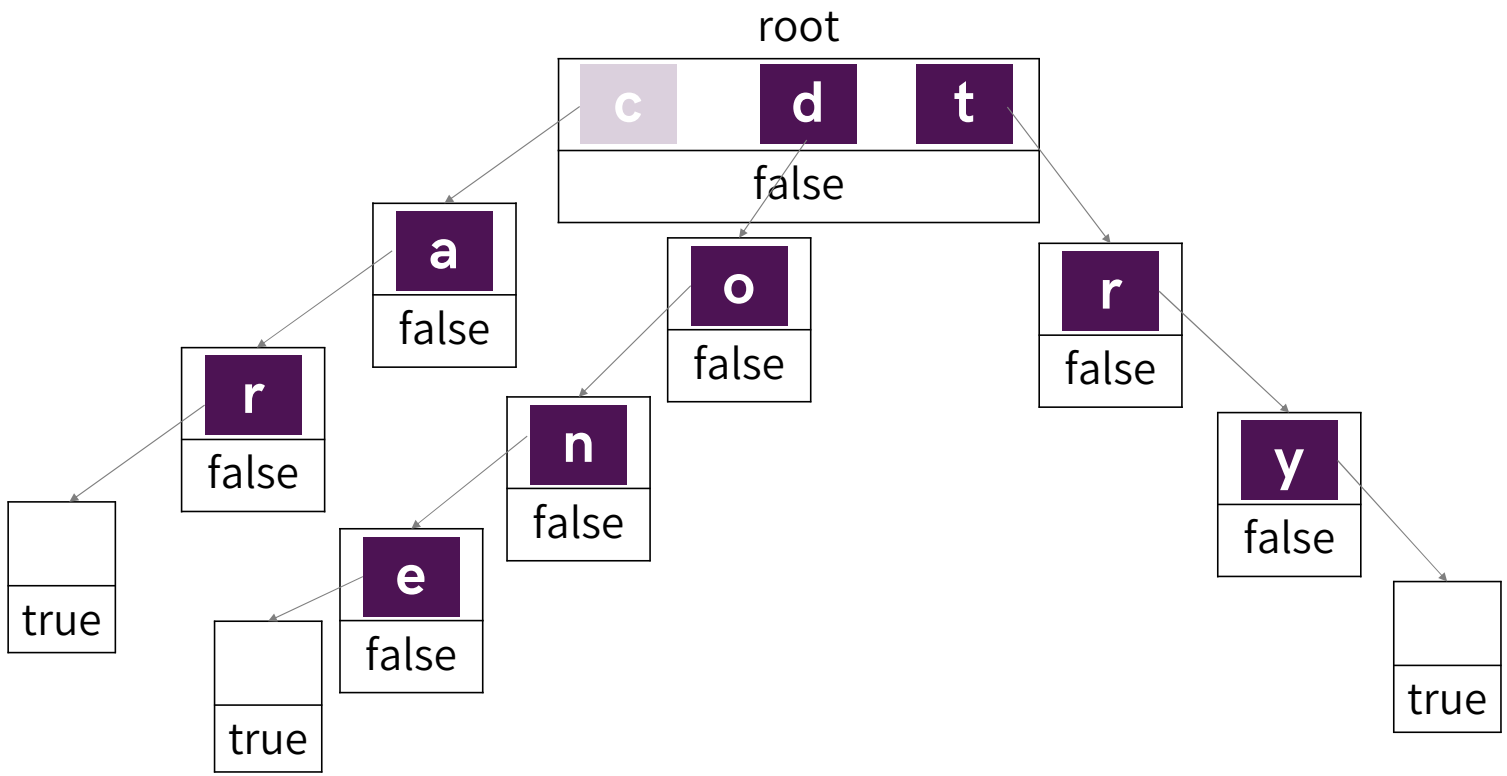
Trie Operations - Insert

car
done
try
cat
trie
do



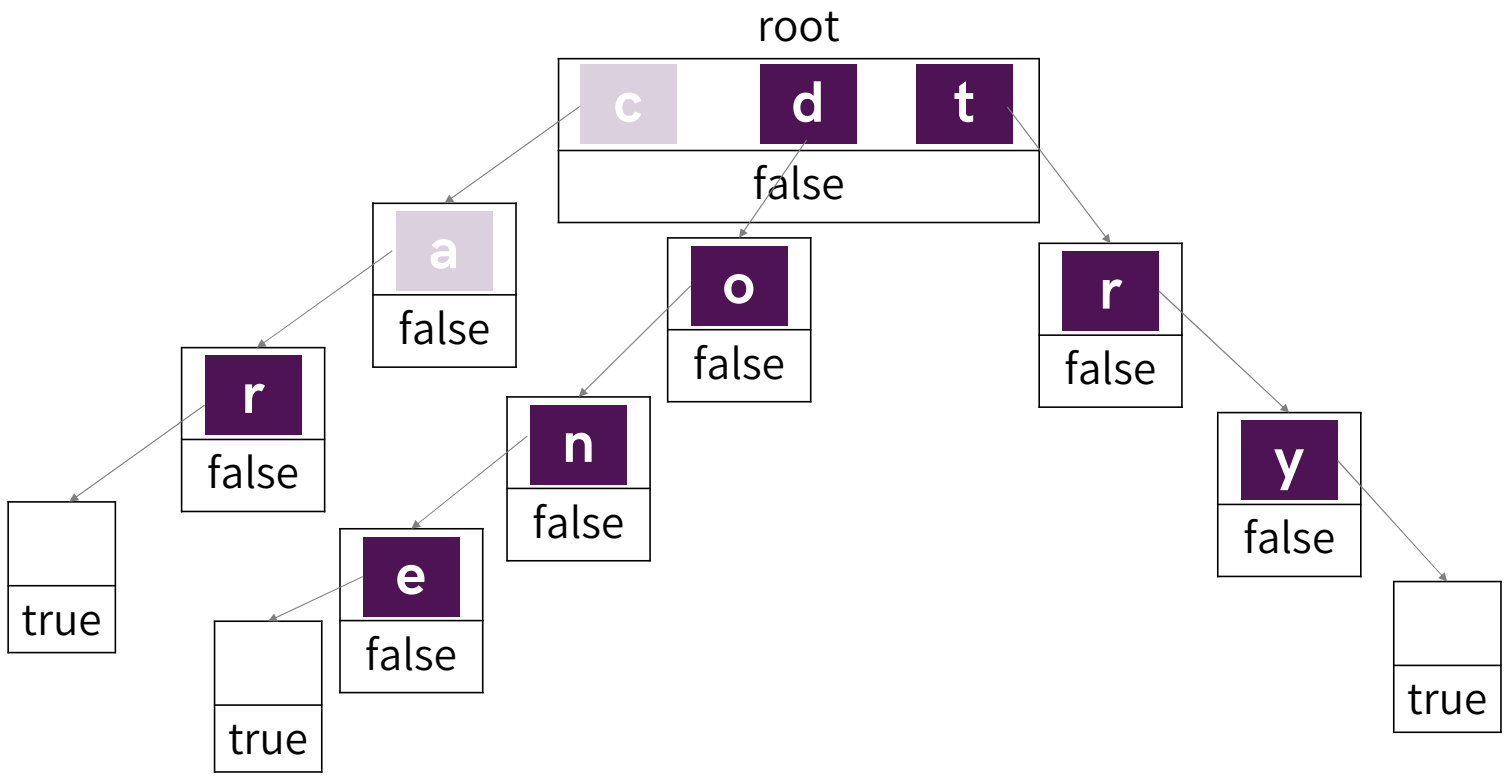
Trie Operations - Insert

car
done
try
cat
trie
do



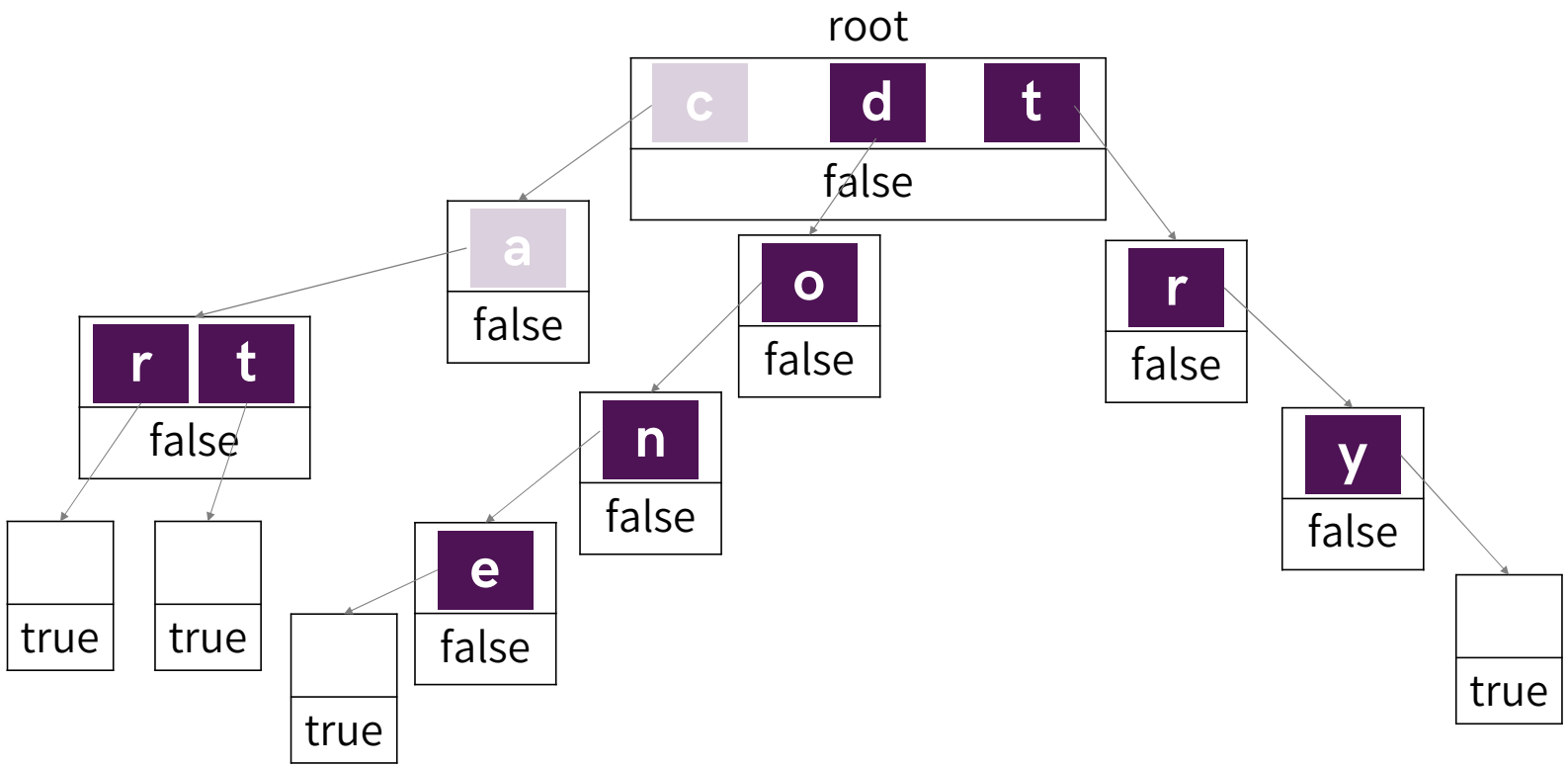
Trie Operations - Insert

car
done
try
cat
trie
do



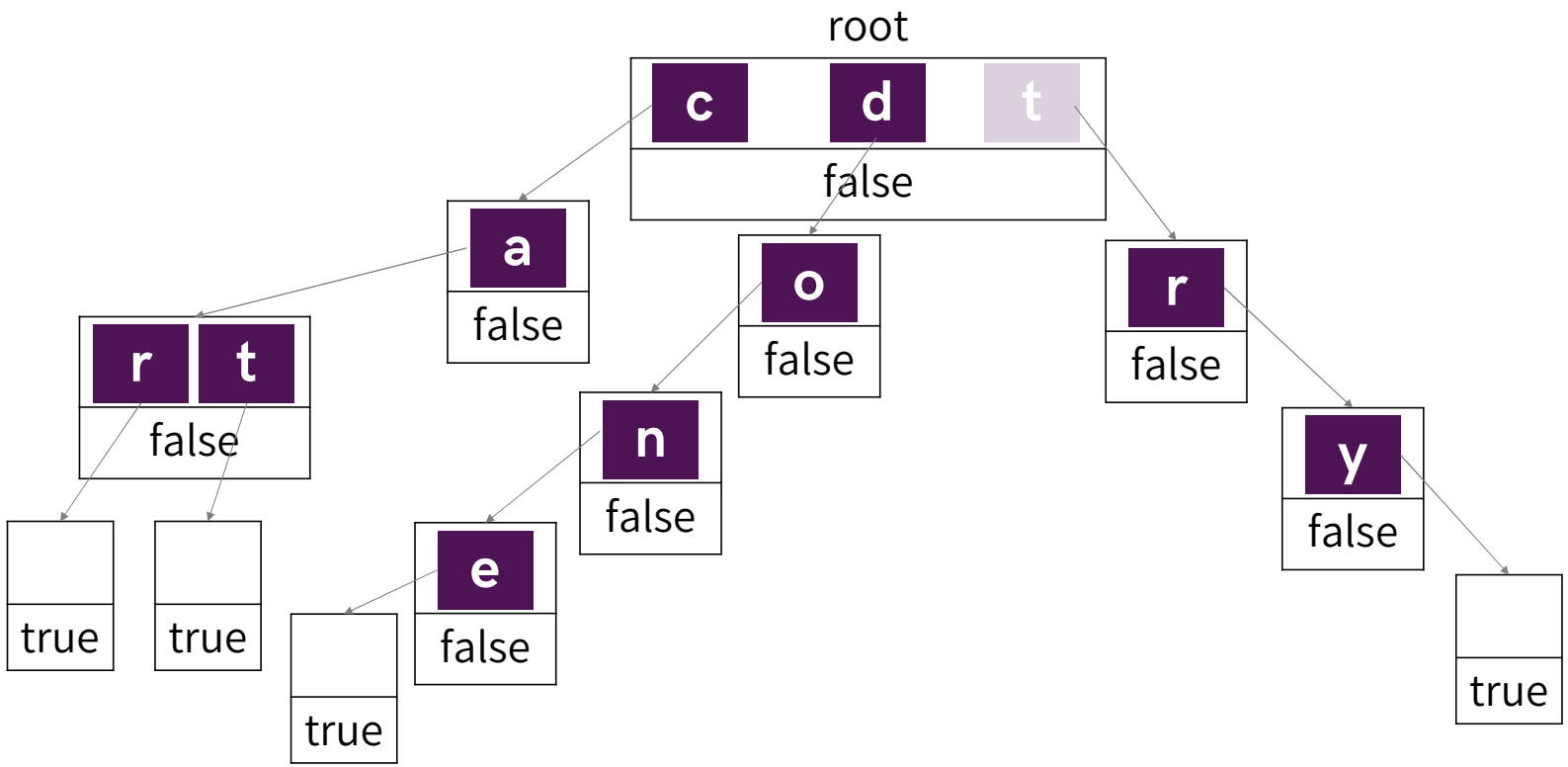
Trie Operations - Insert

car
done
try
cat
trie
do



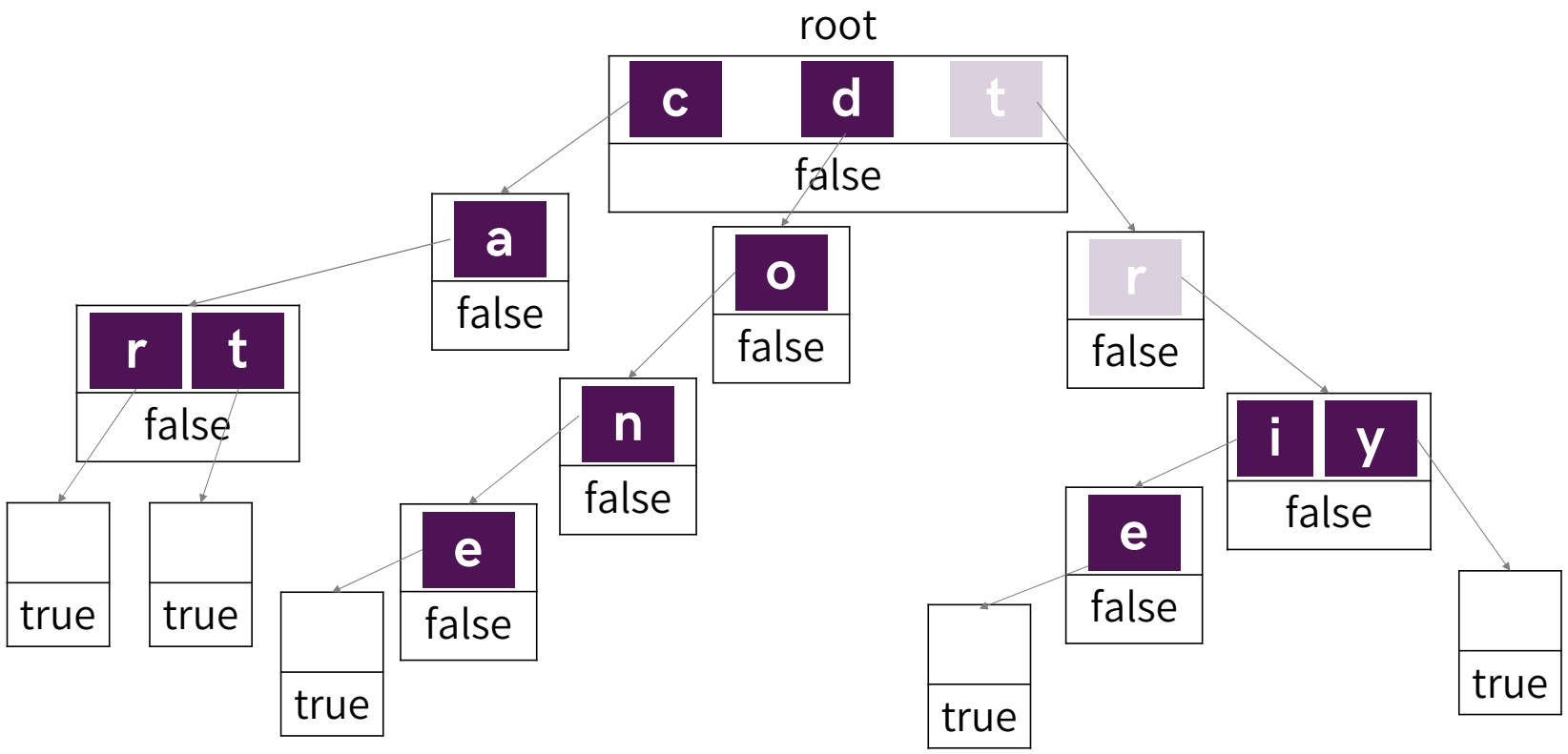
Trie Operations - Insert

car
done
try
cat
trie
do



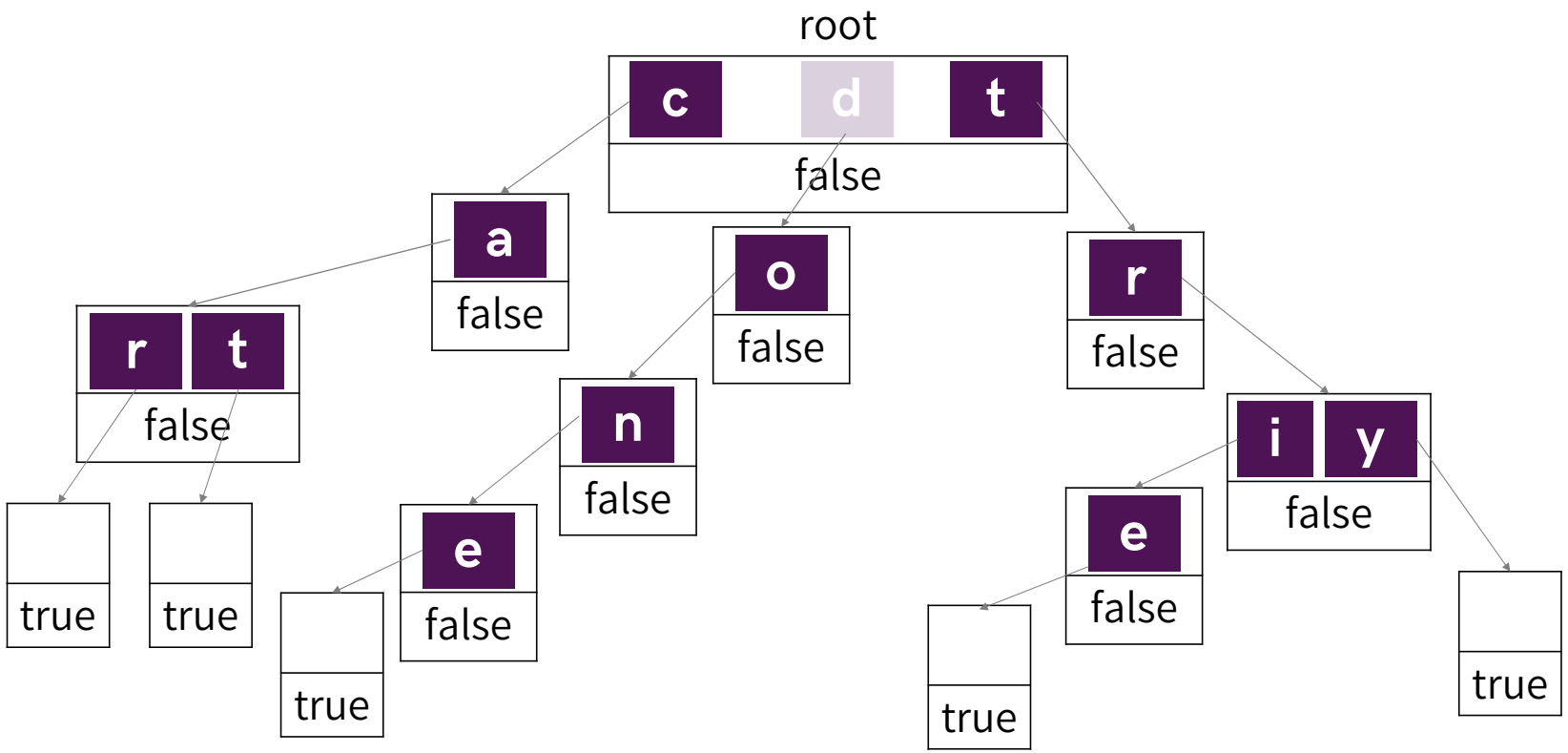
Trie Operations - Insert

car
done
try
cat
trie
do



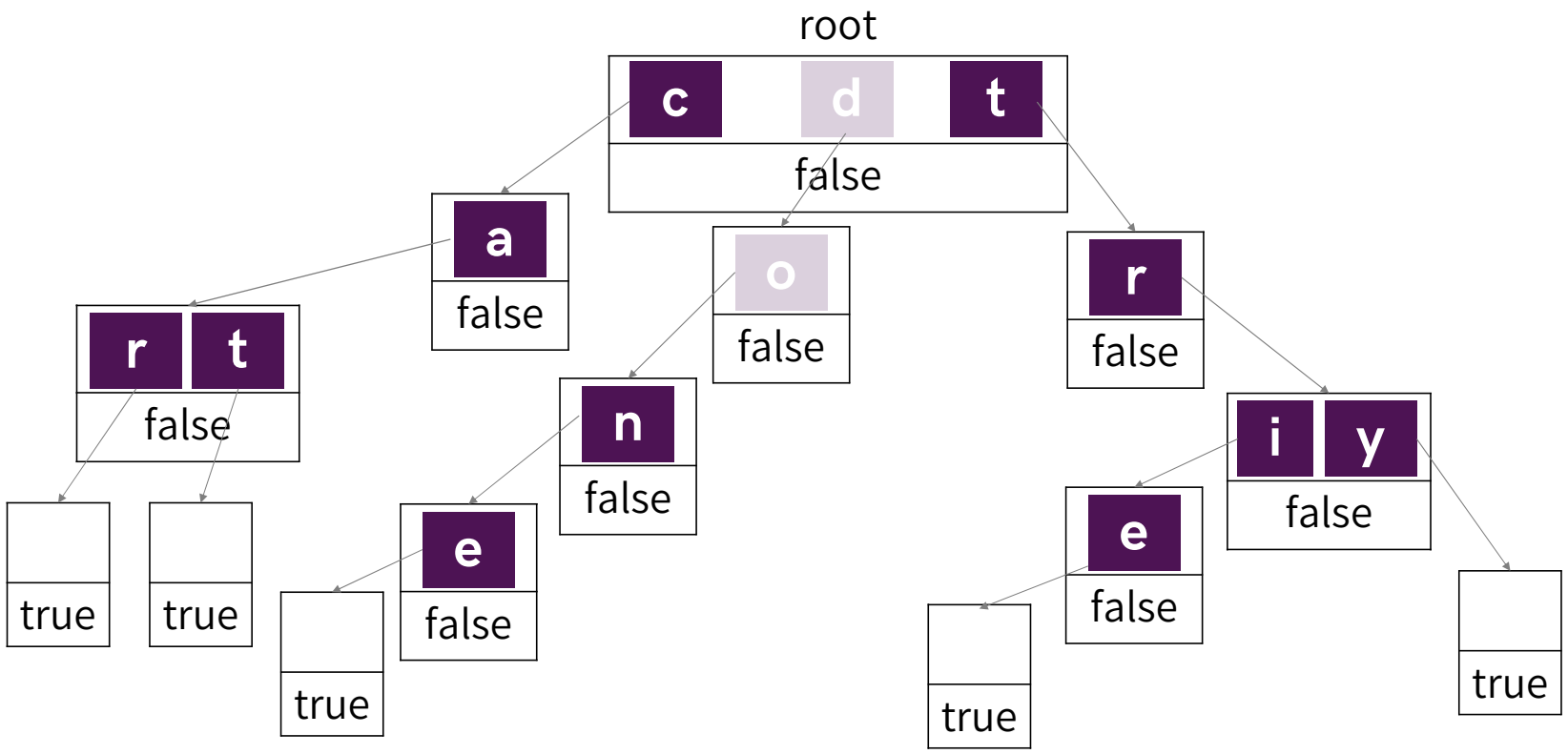
Trie Operations - Insert

car
done
try
cat
trie
do



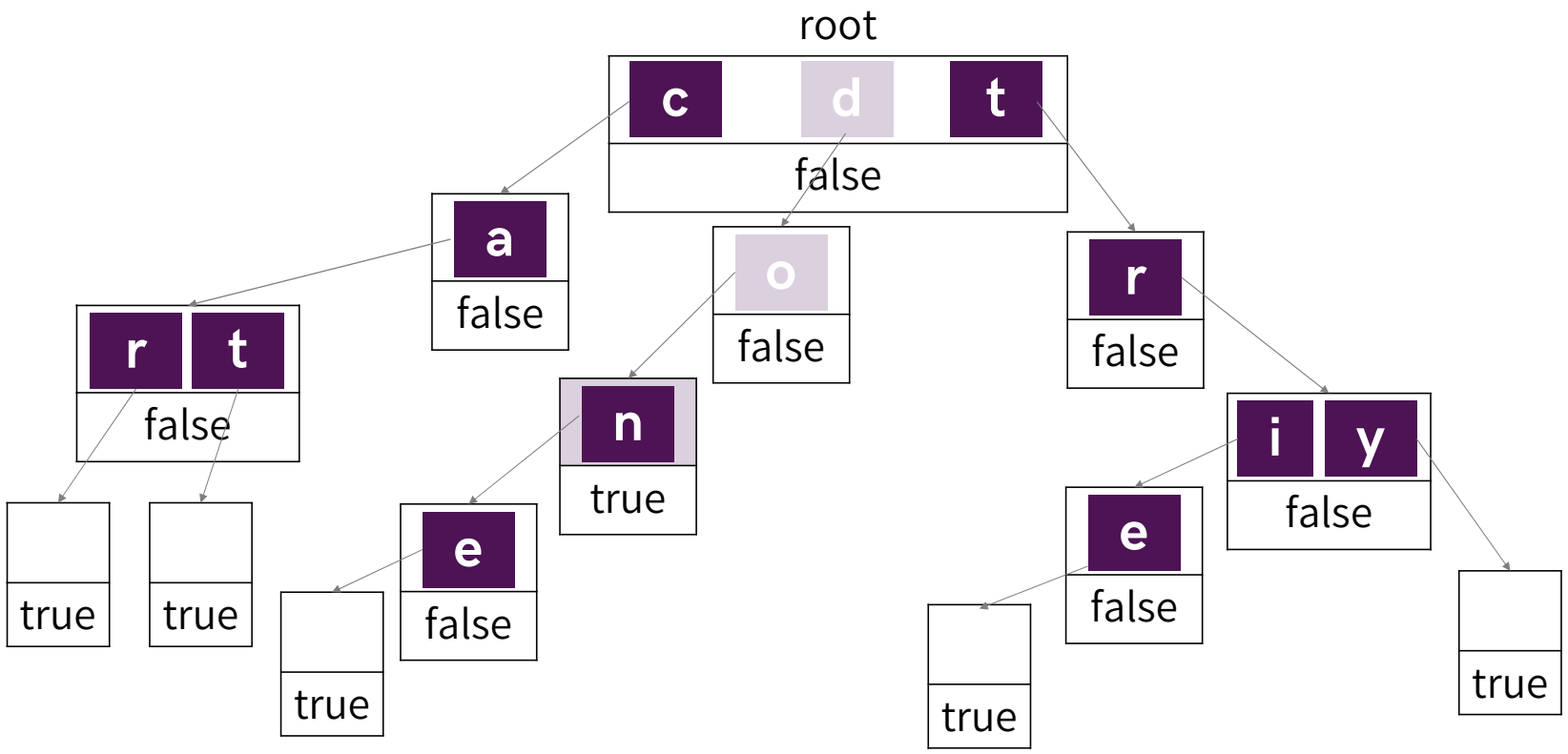
Trie Operations - Insert

car
done
try
cat
trie
do



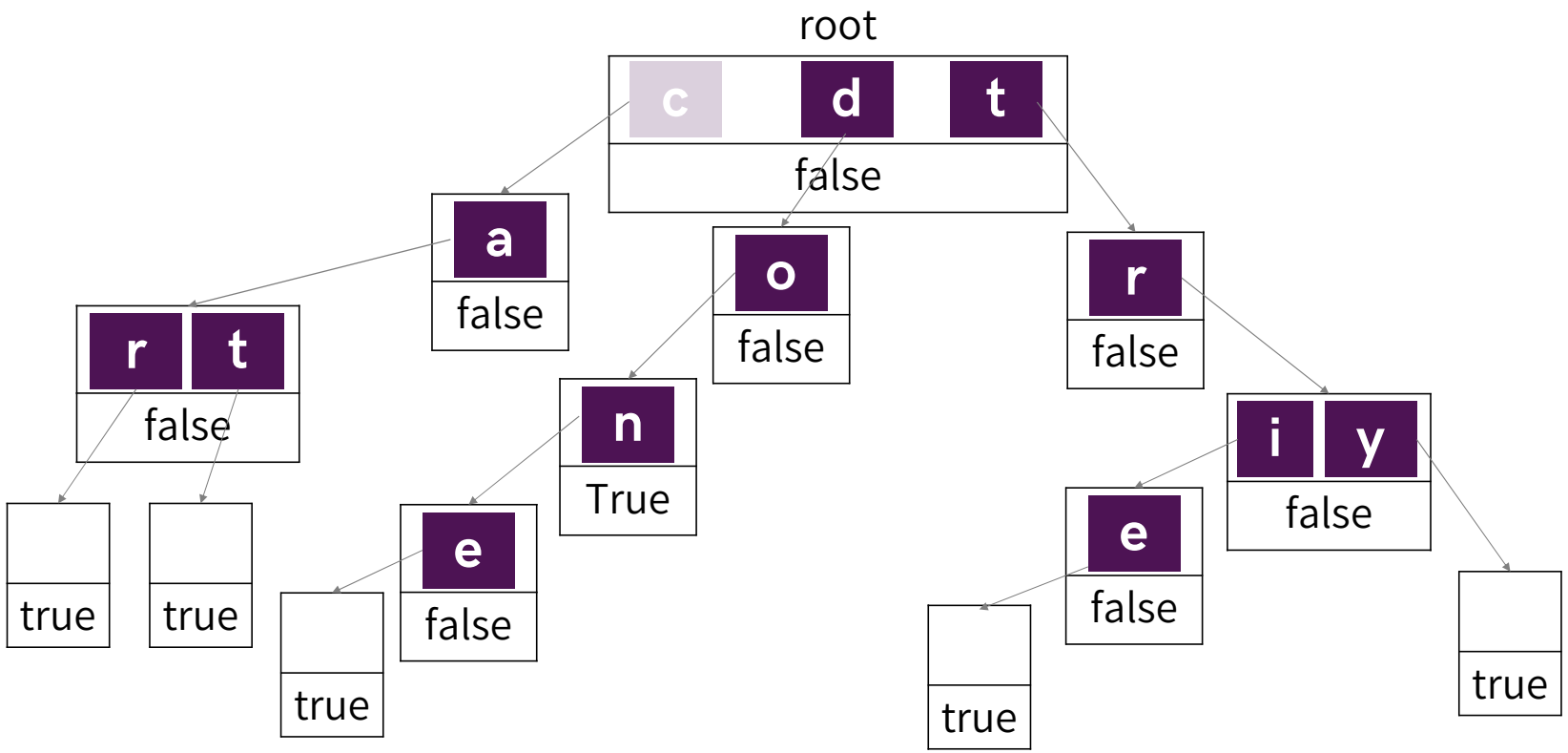
Trie Operations - Insert

car
done
try
cat
trie
do



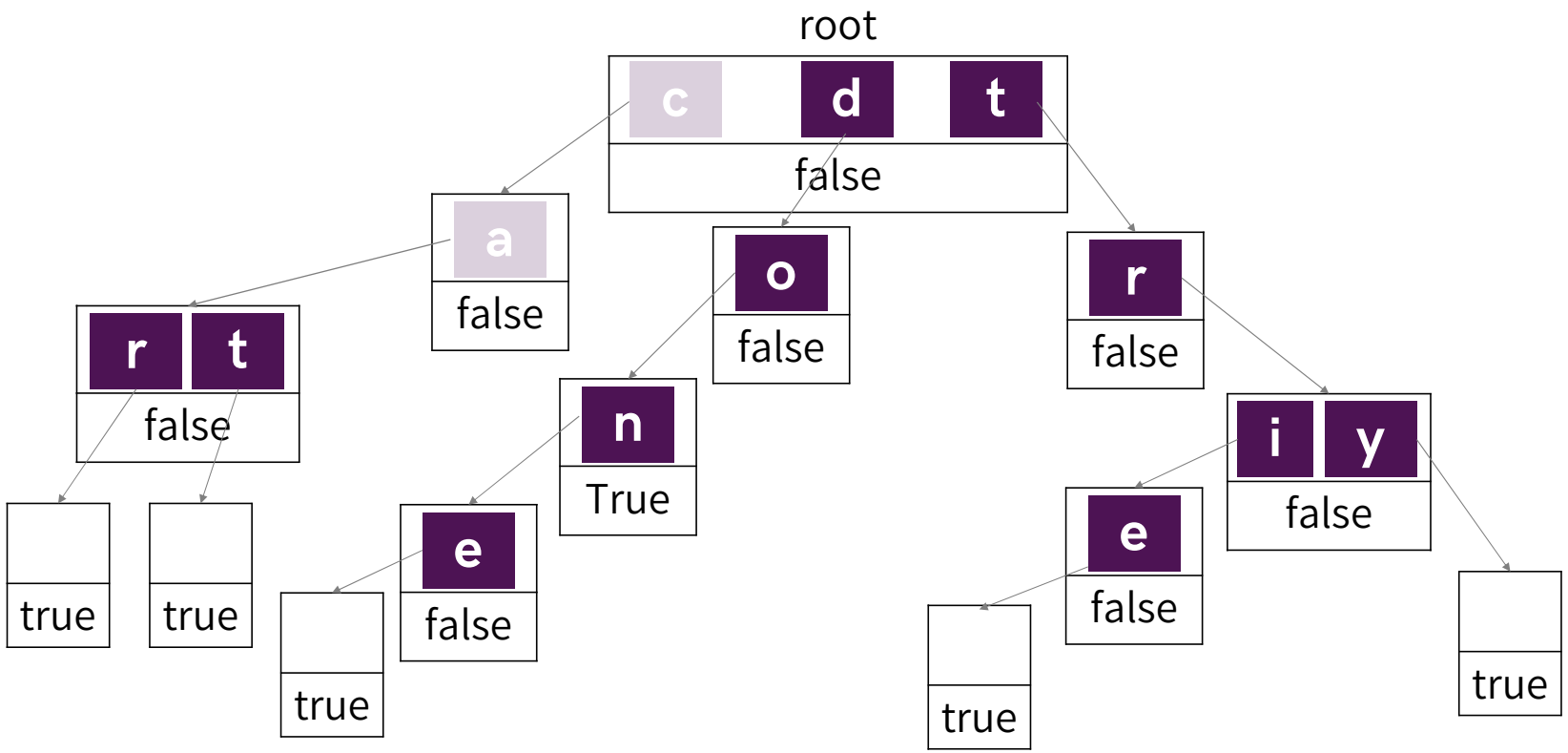
Trie Operations - Search

ca
zebra
try
card



Trie Operations - Search

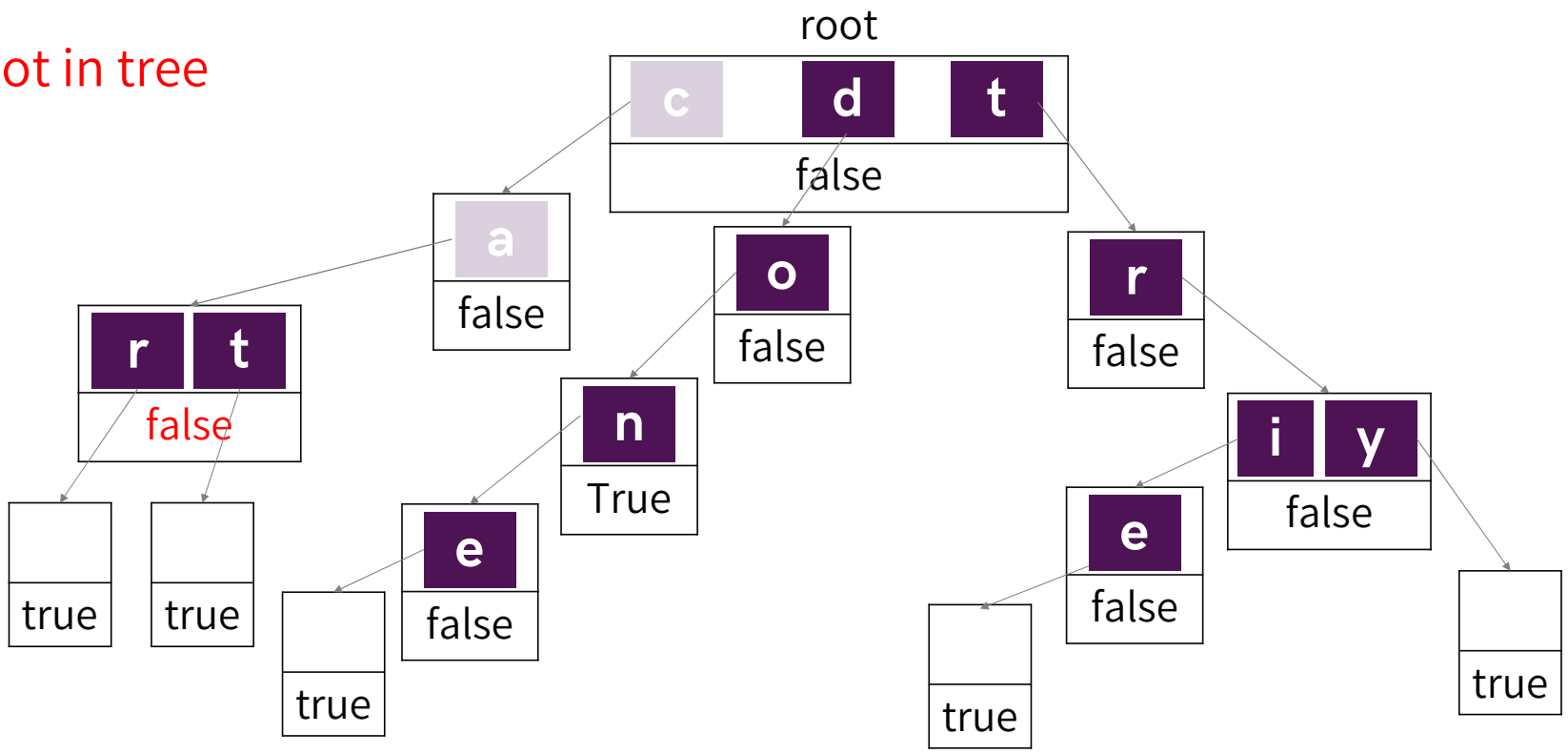
ca
zebra
try
card



Trie Operations - Search

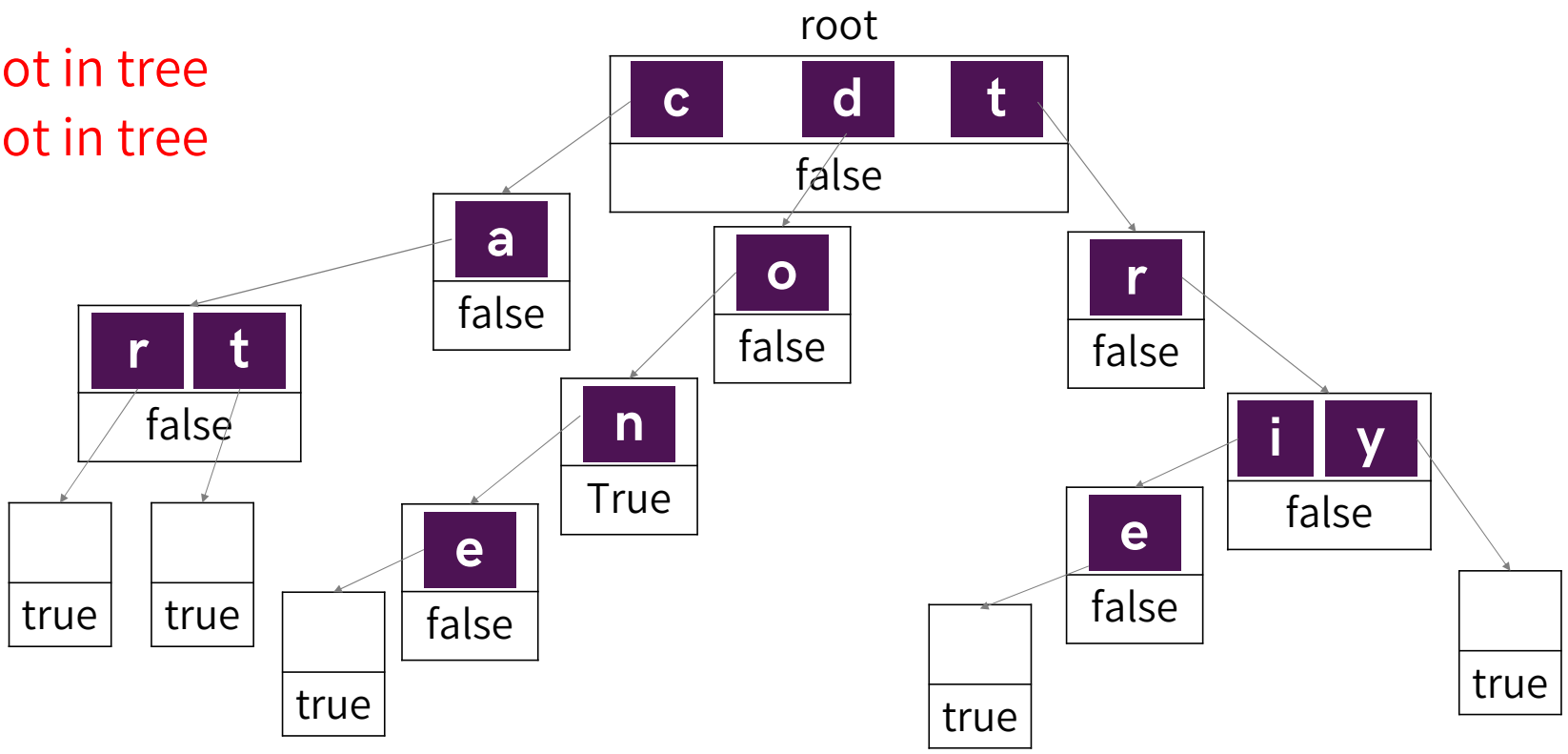
ca
zebra
try
card

Not in tree



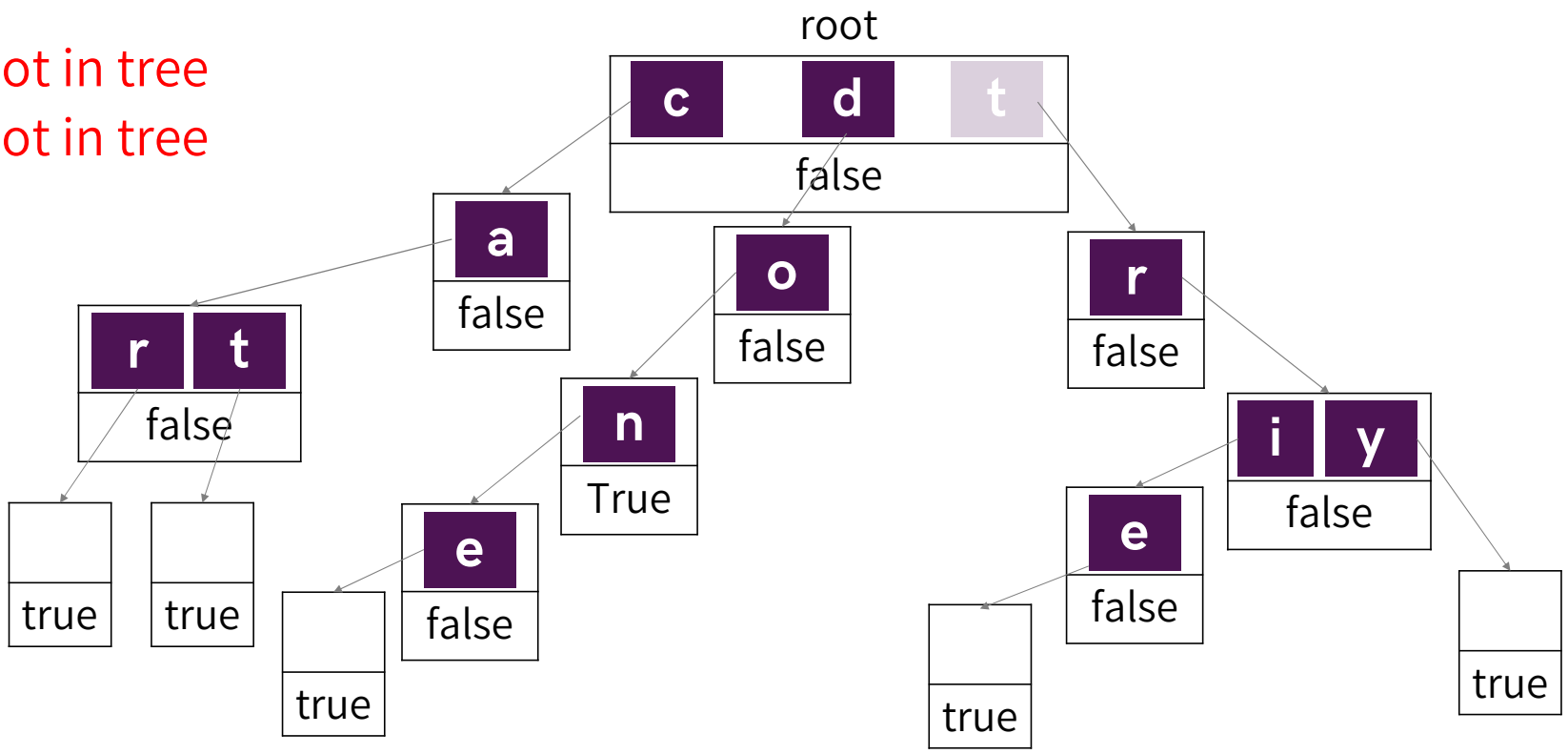
Trie Operations - Search

ca Not in tree
zebra Not in tree
try
card



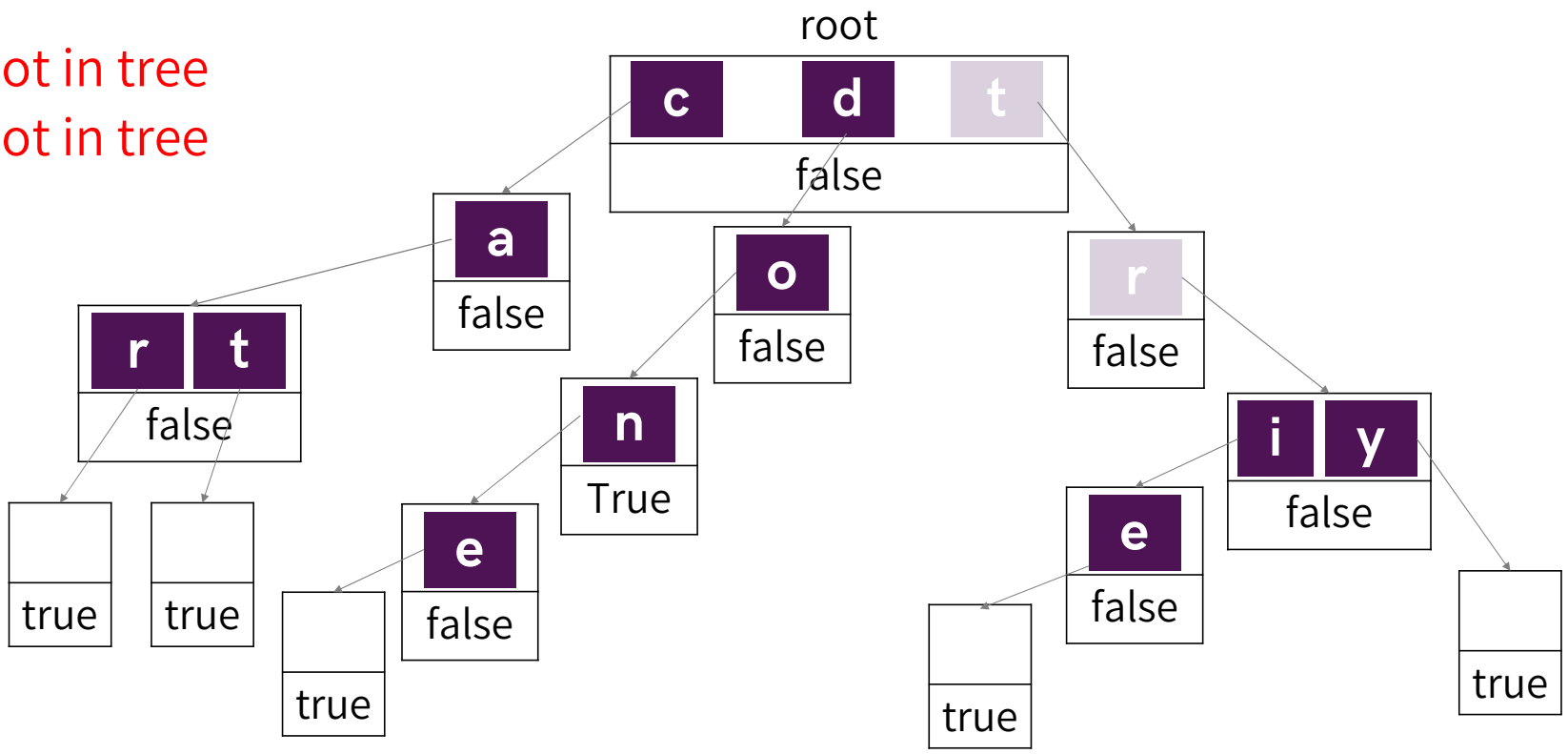
Trie Operations - Search

ca Not in tree
zebra Not in tree
try
card



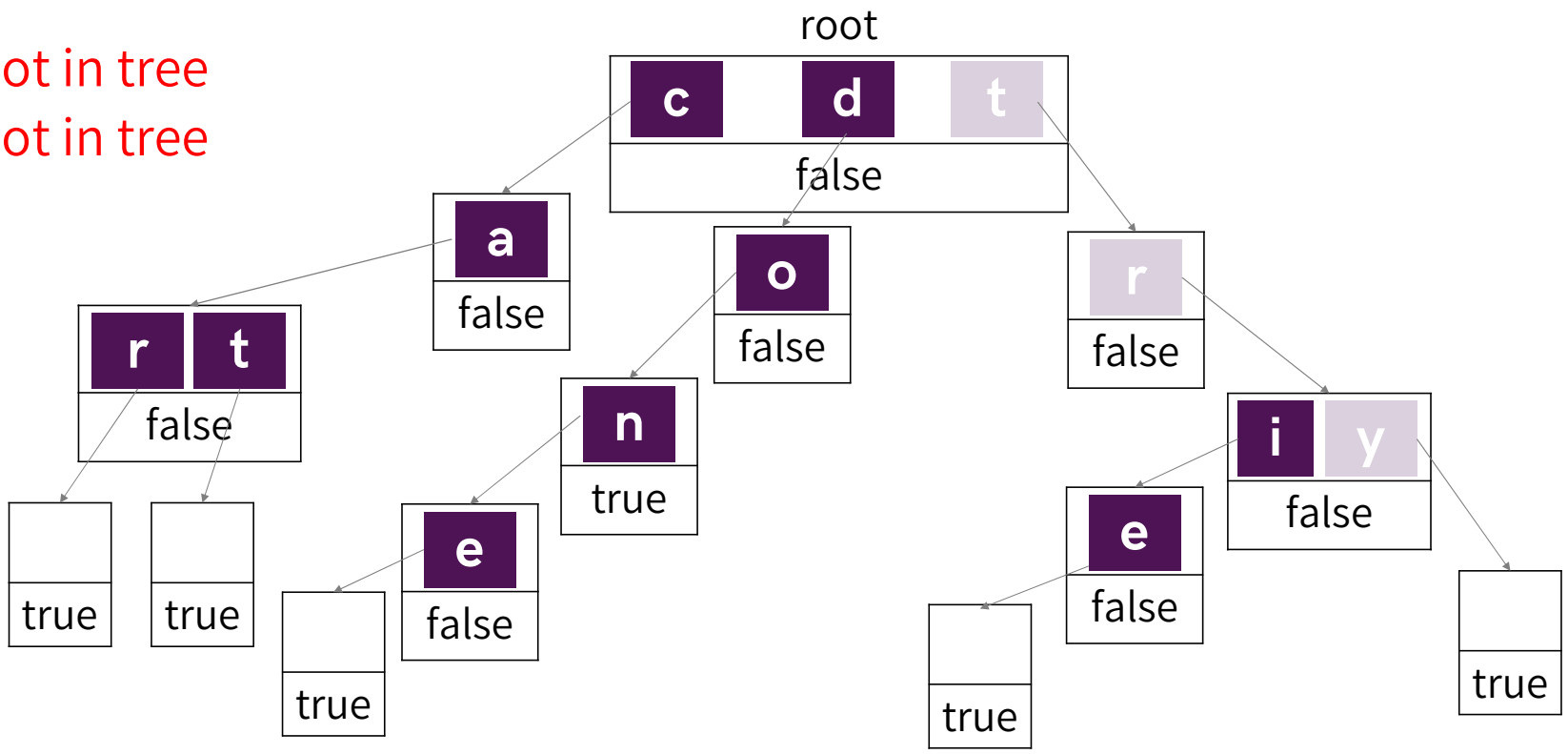
Trie Operations - Search

ca Not in tree
zebra Not in tree
try
card



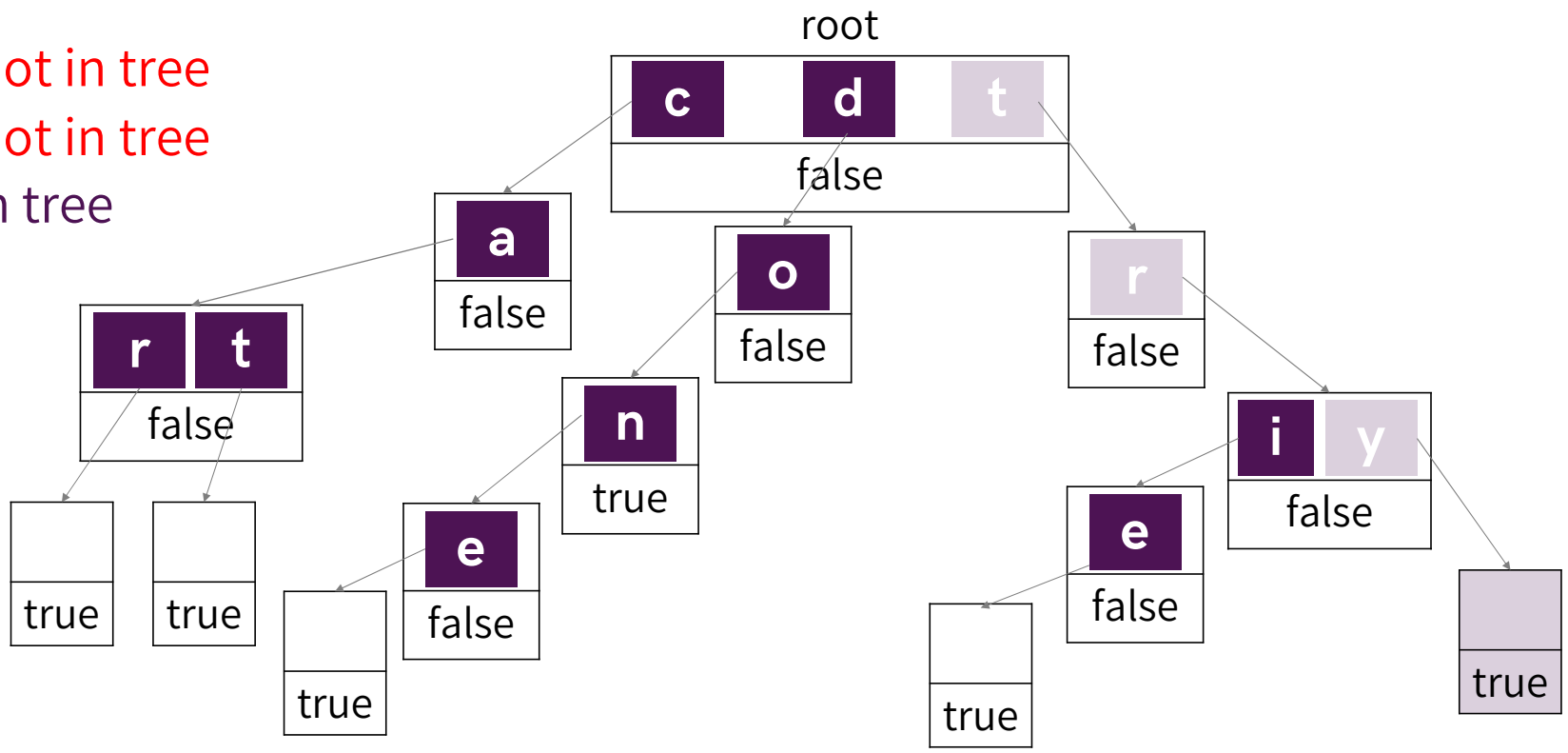
Trie Operations - Search

ca Not in tree
zebra Not in tree
try
card



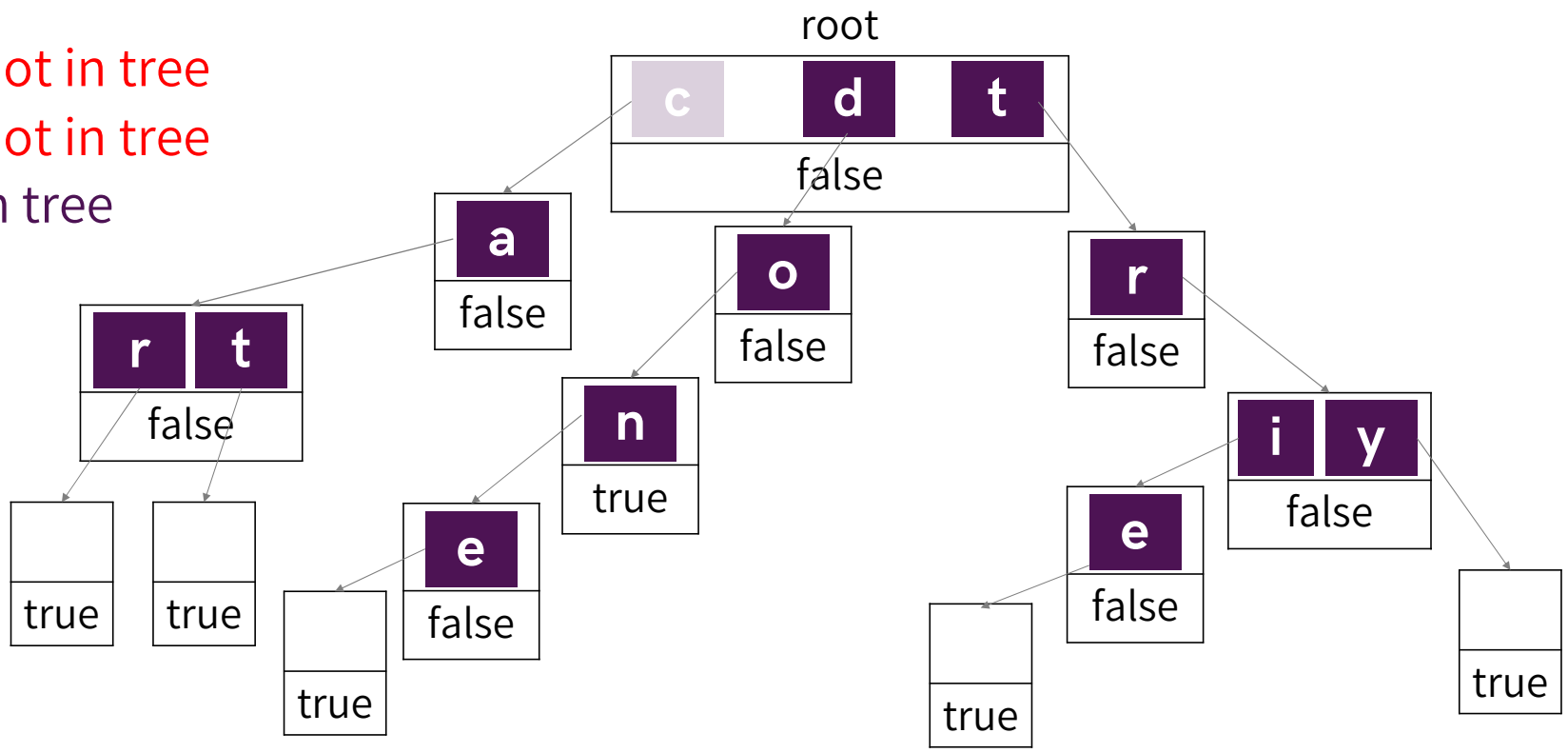
Trie Operations - Search

ca Not in tree
zebra Not in tree
try In tree
card



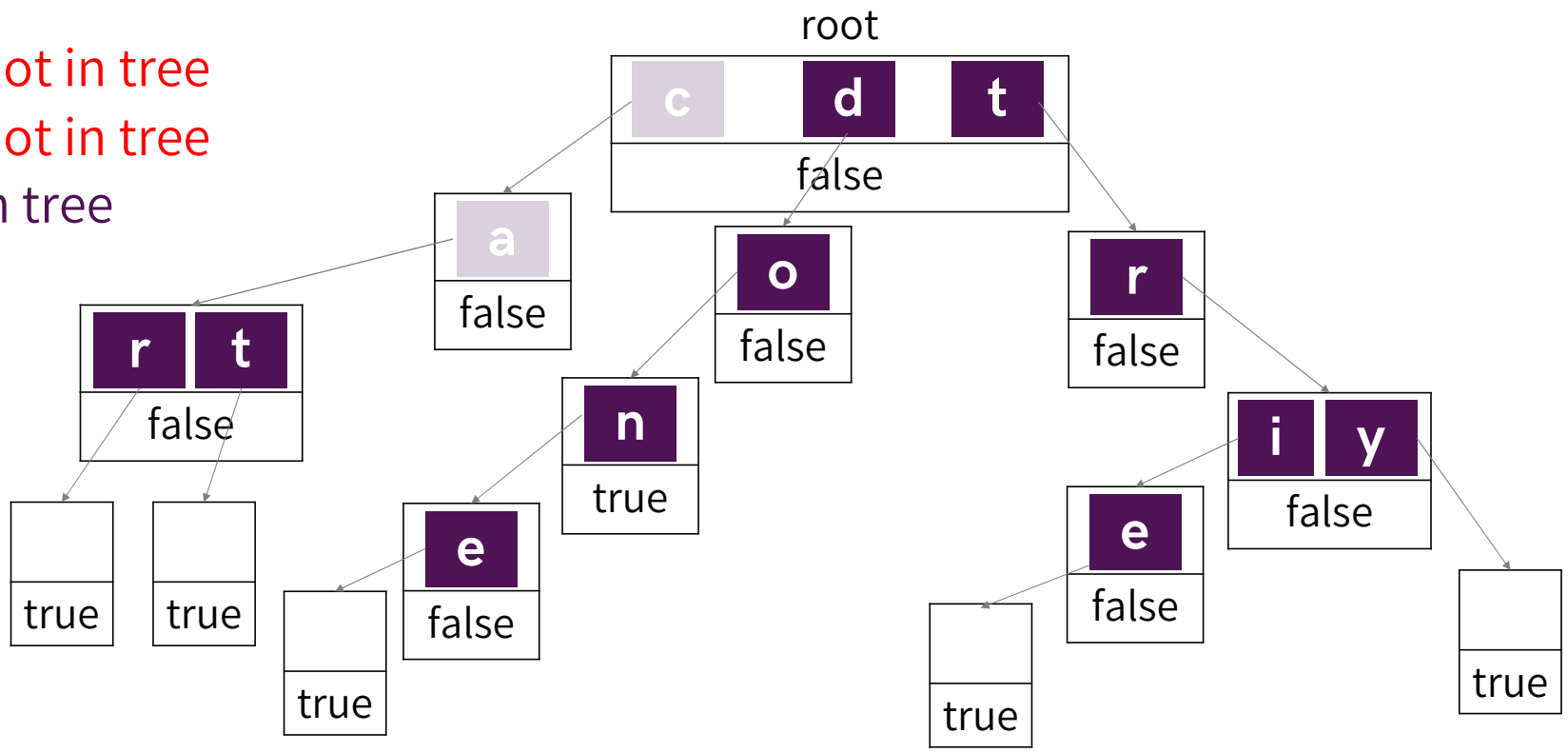
Trie Operations - Search

ca Not in tree
zebra Not in tree
try In tree
card



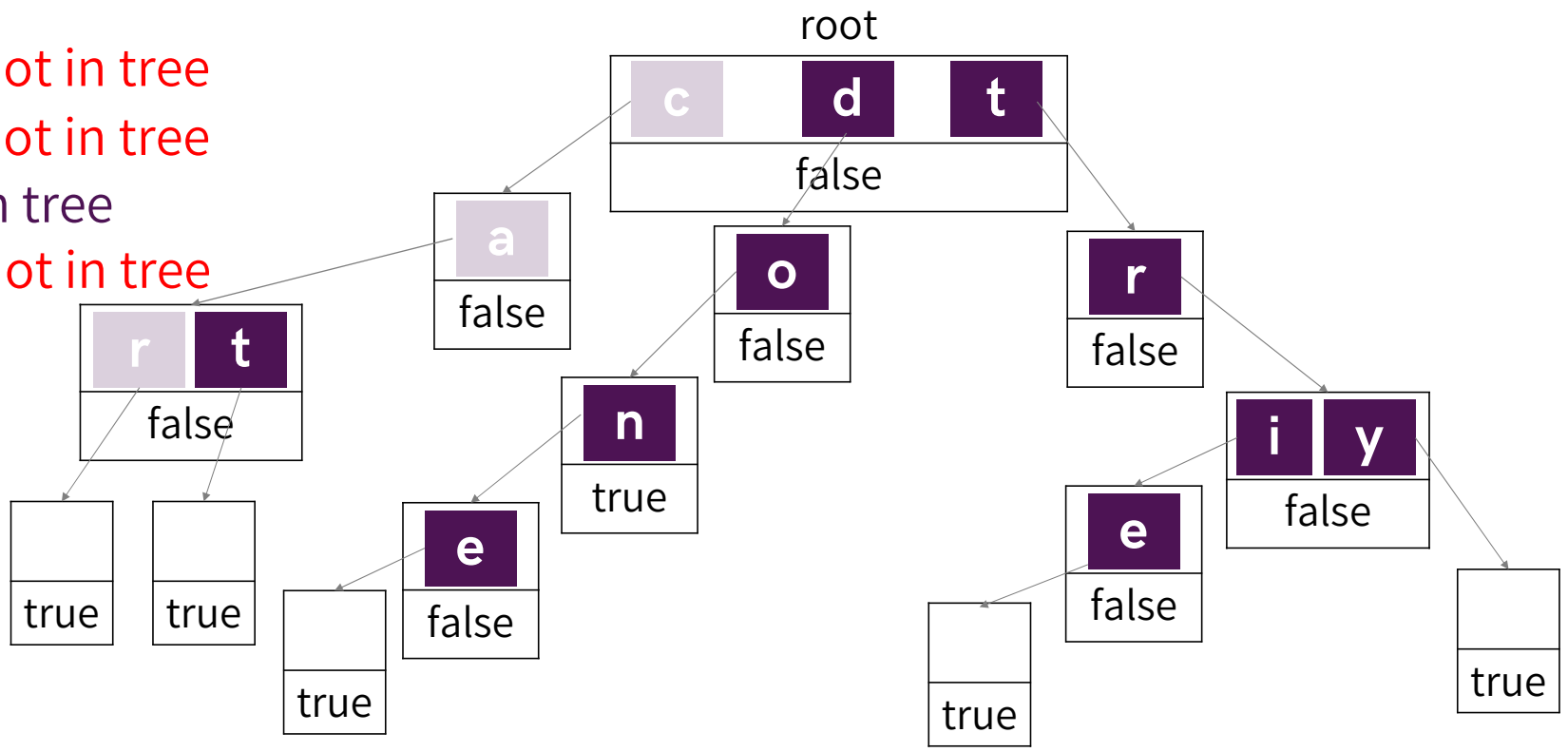
Trie Operations - Search

ca Not in tree
zebra Not in tree
try In tree
card



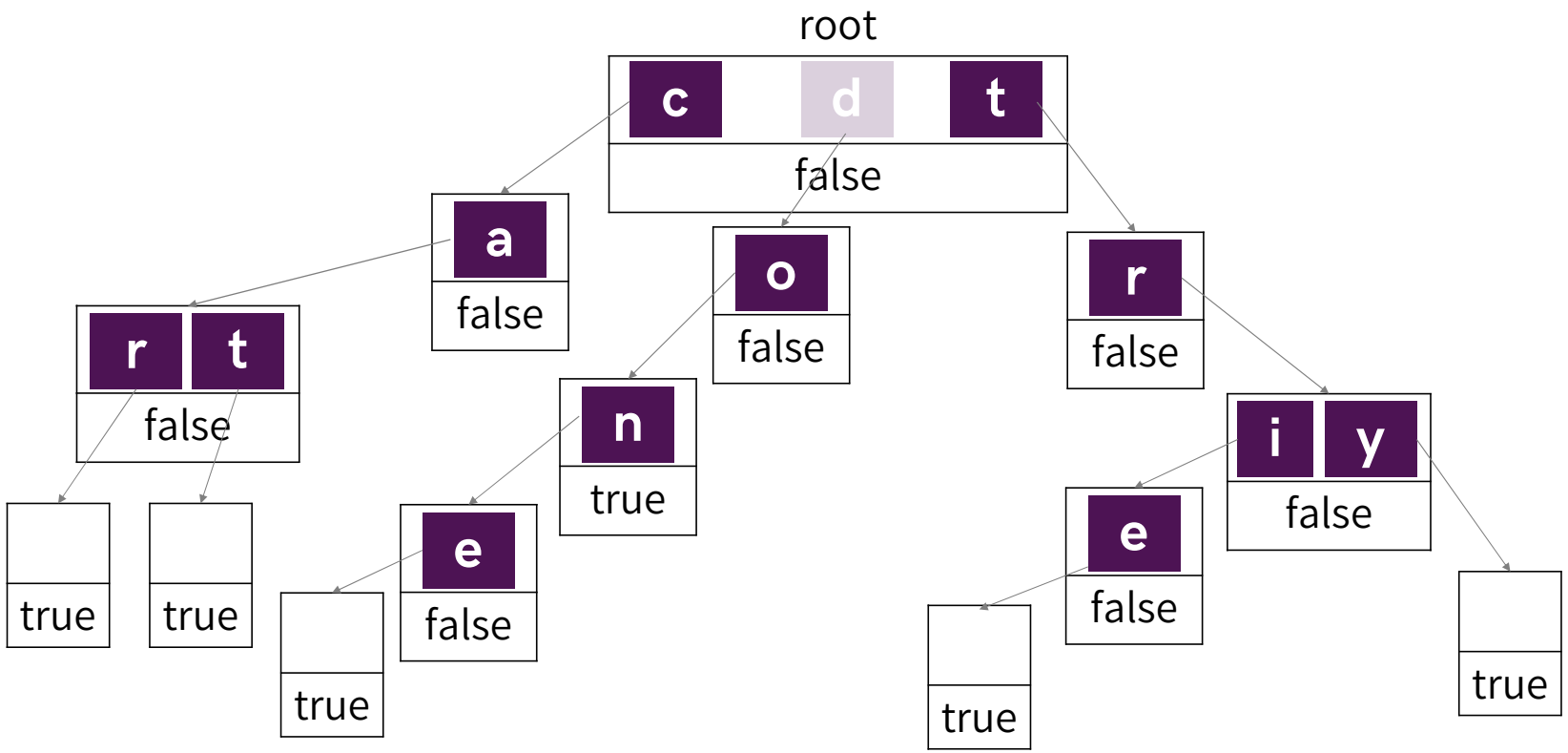
Trie Operations - Search

ca Not in tree
zebra Not in tree
try In tree
card Not in tree



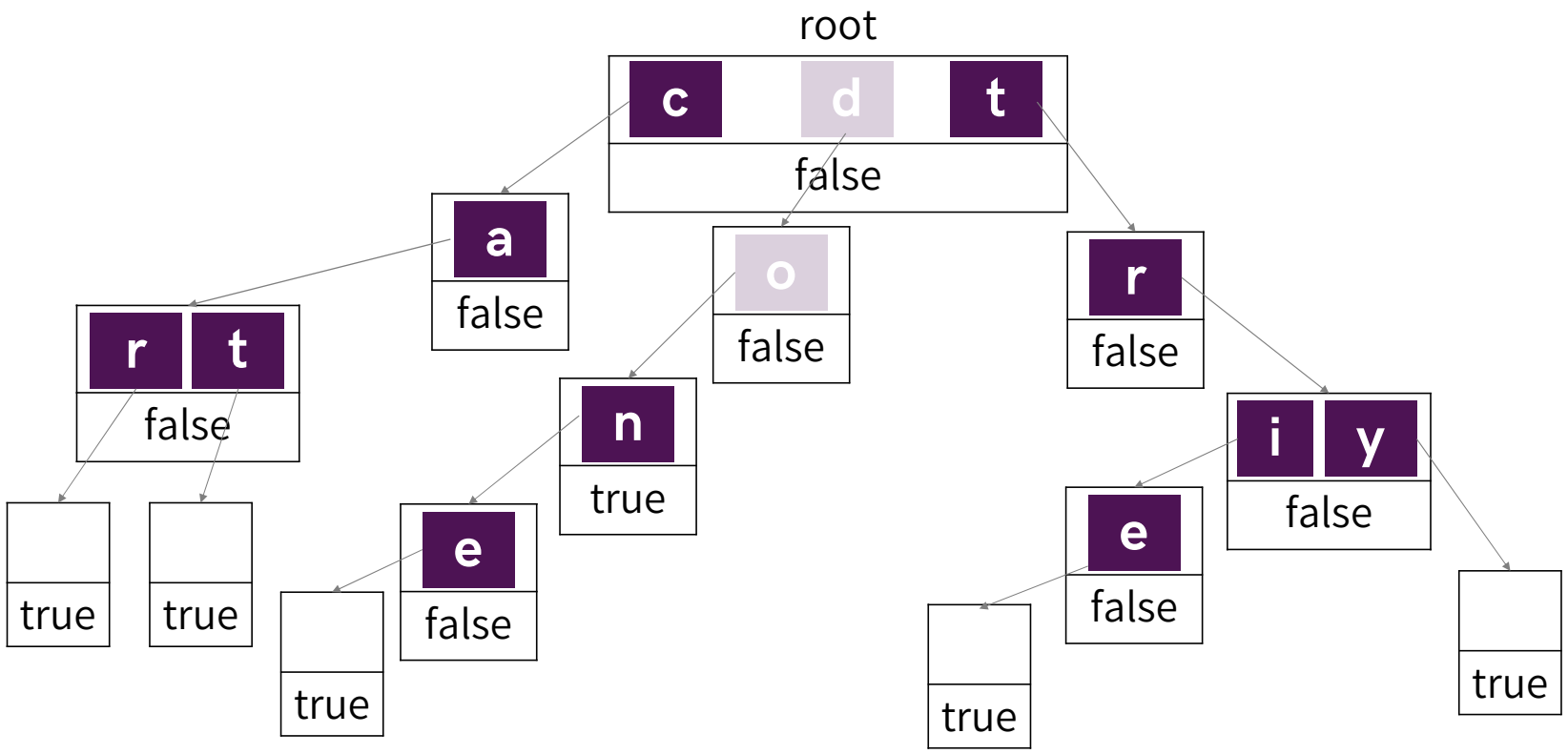
Trie Operations - Delete

do
trie
ca



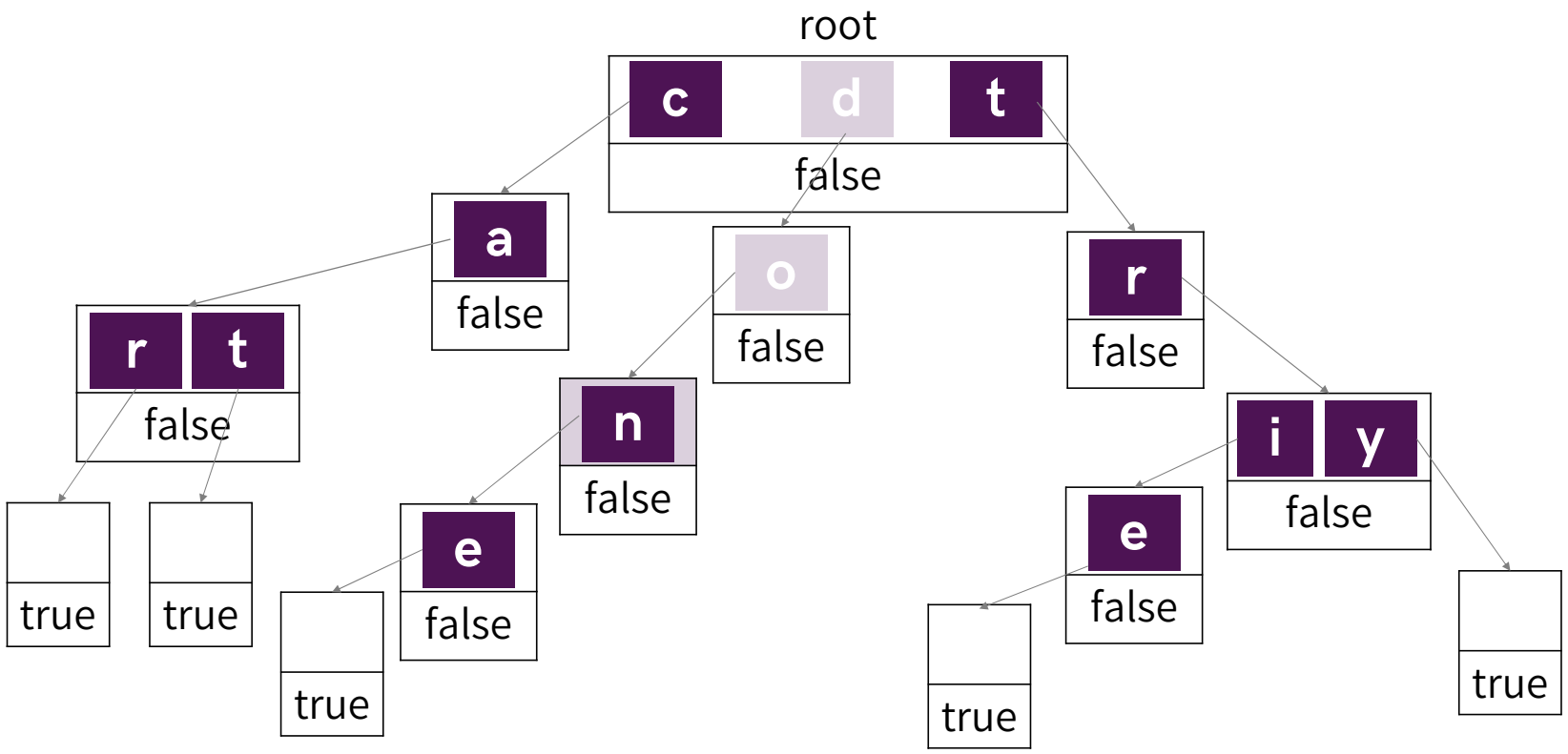
Trie Operations - Delete

do
trie
ca



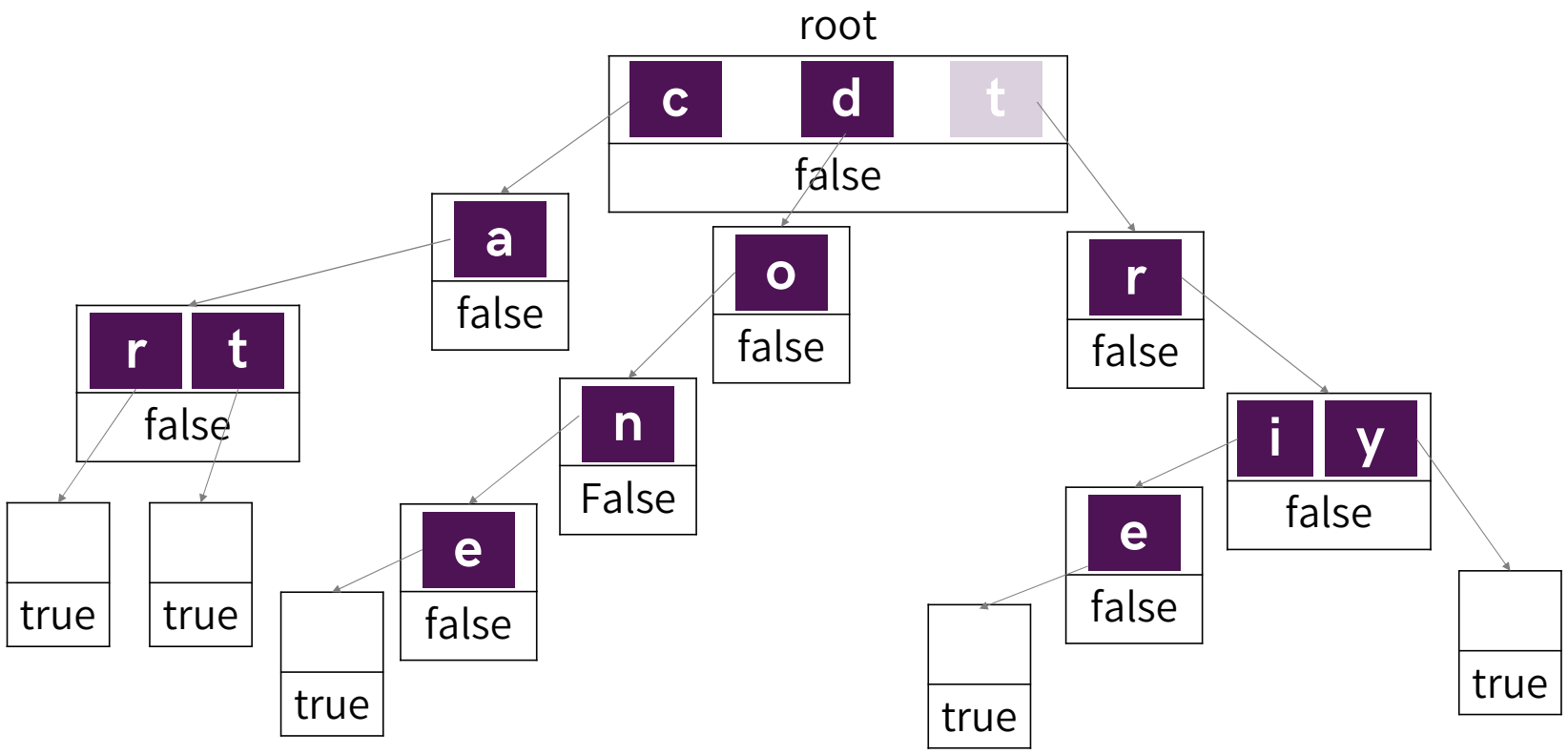
Trie Operations - Delete

do
trie
ca



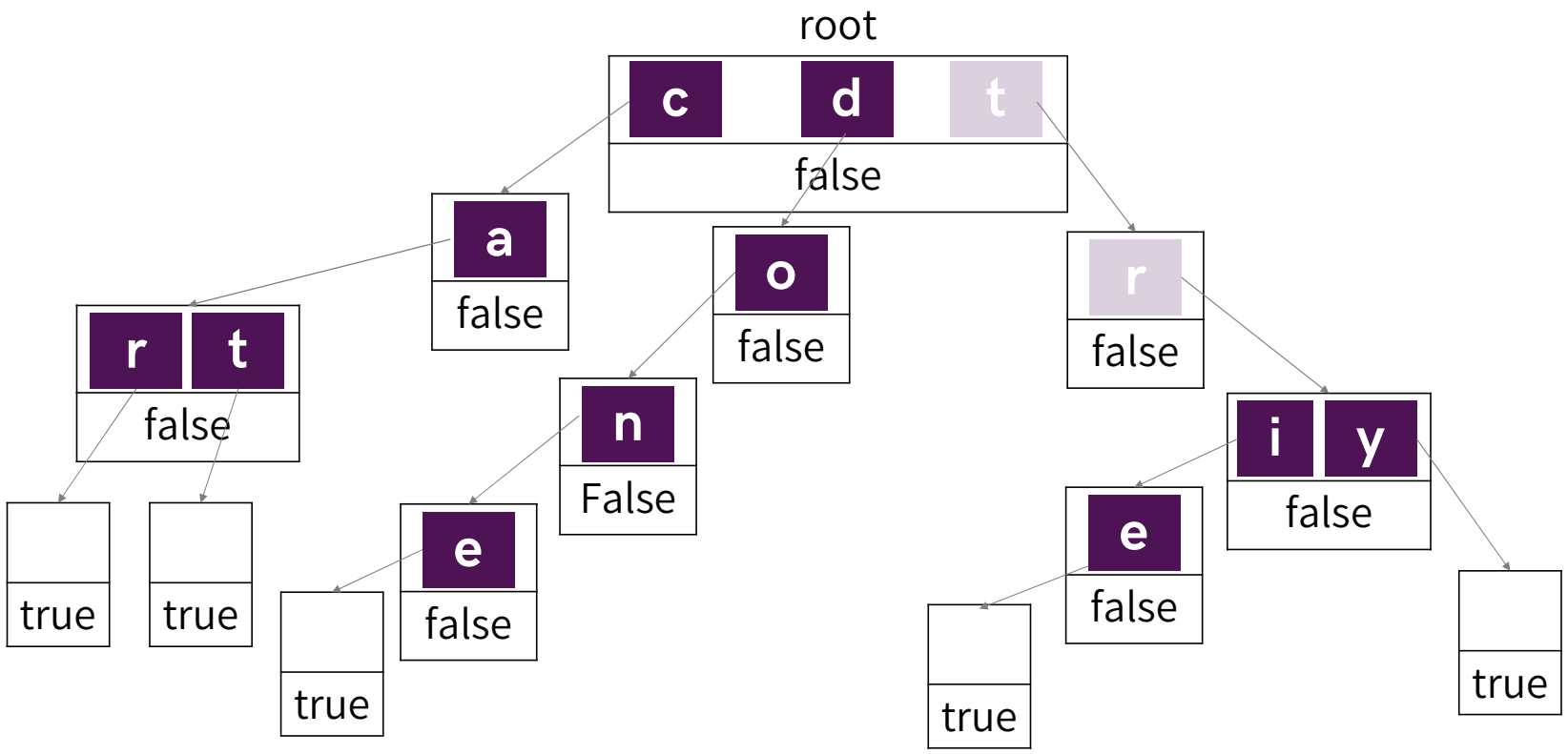
Trie Operations - Delete

trie
ca



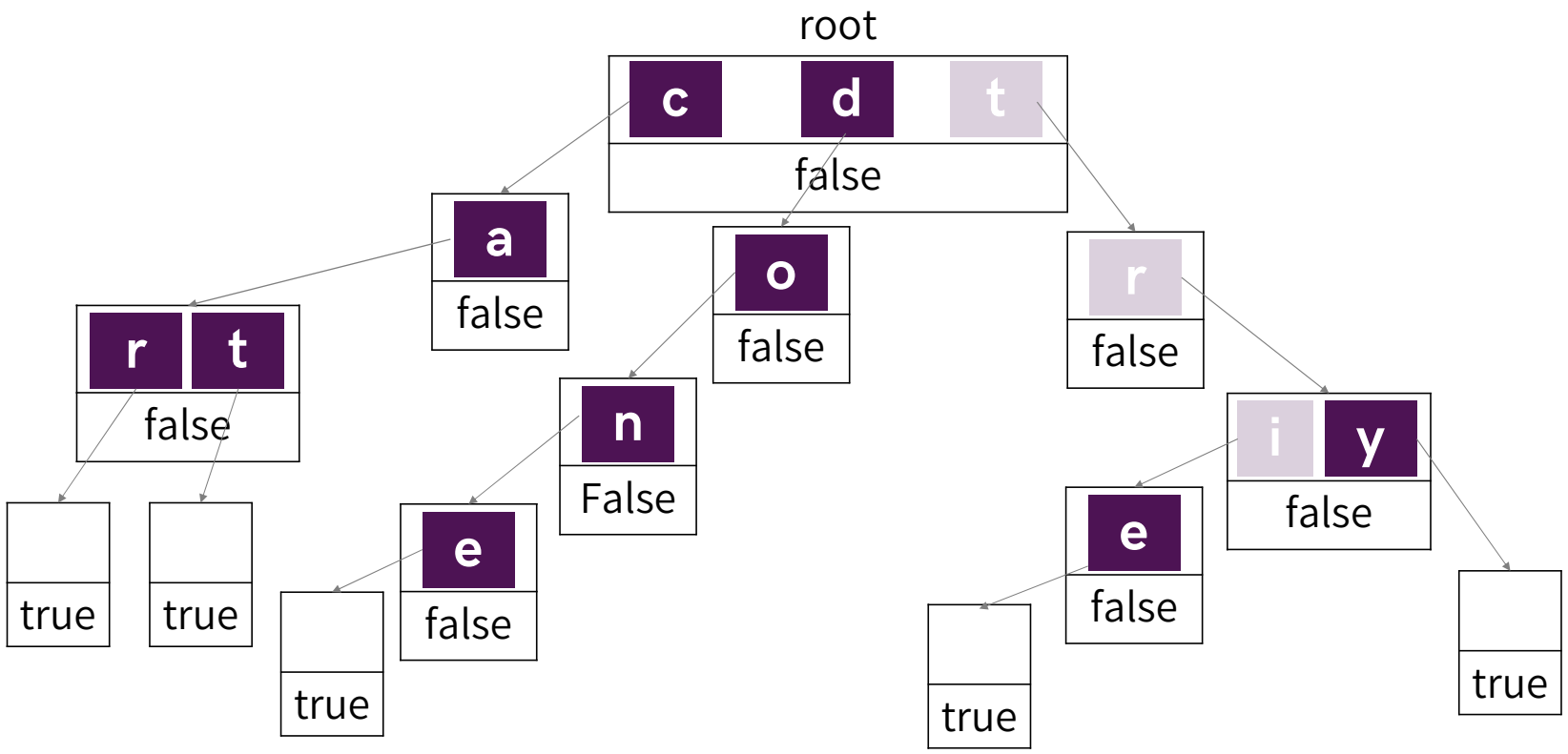
Trie Operations - Delete

trie
ca



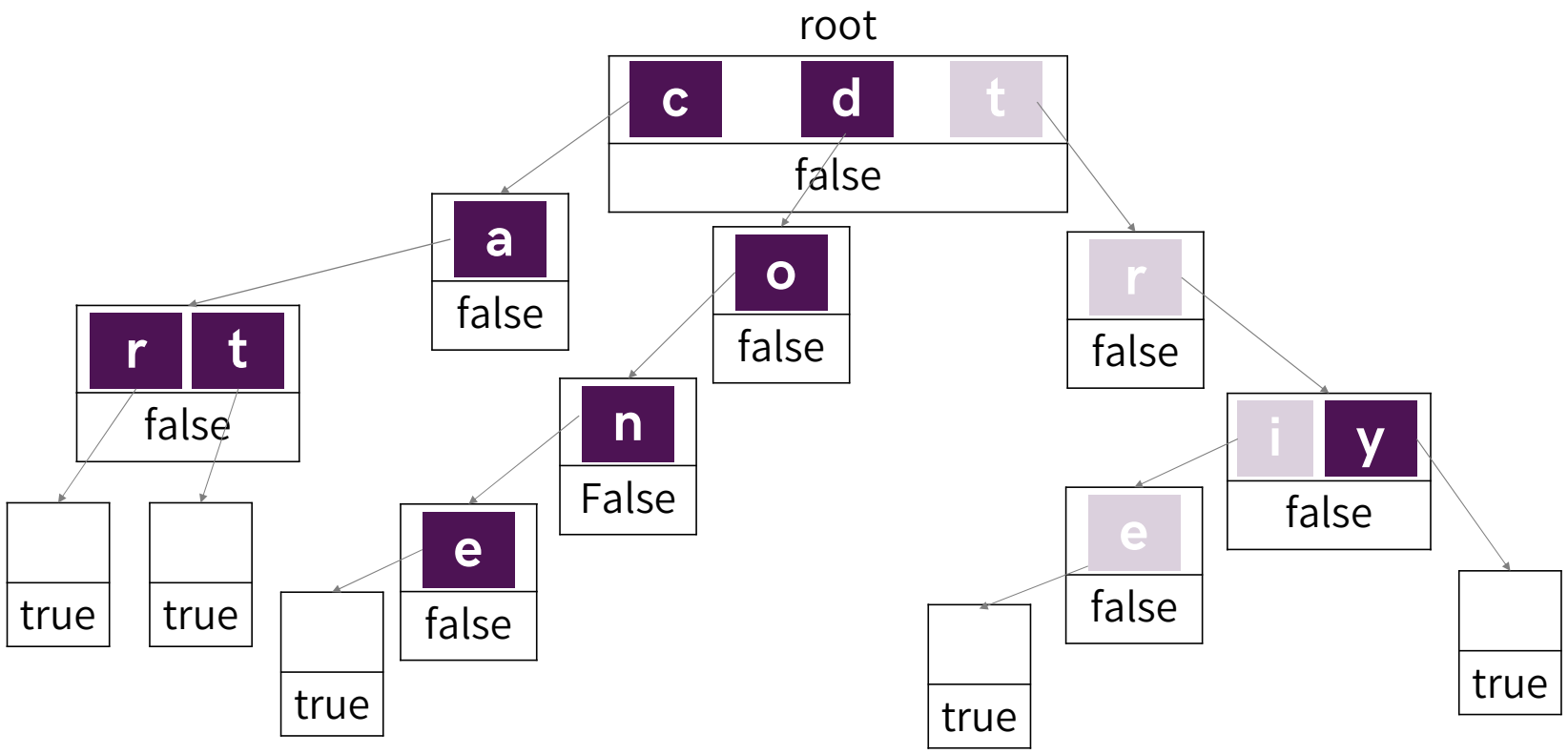
Trie Operations - Delete

trie
ca



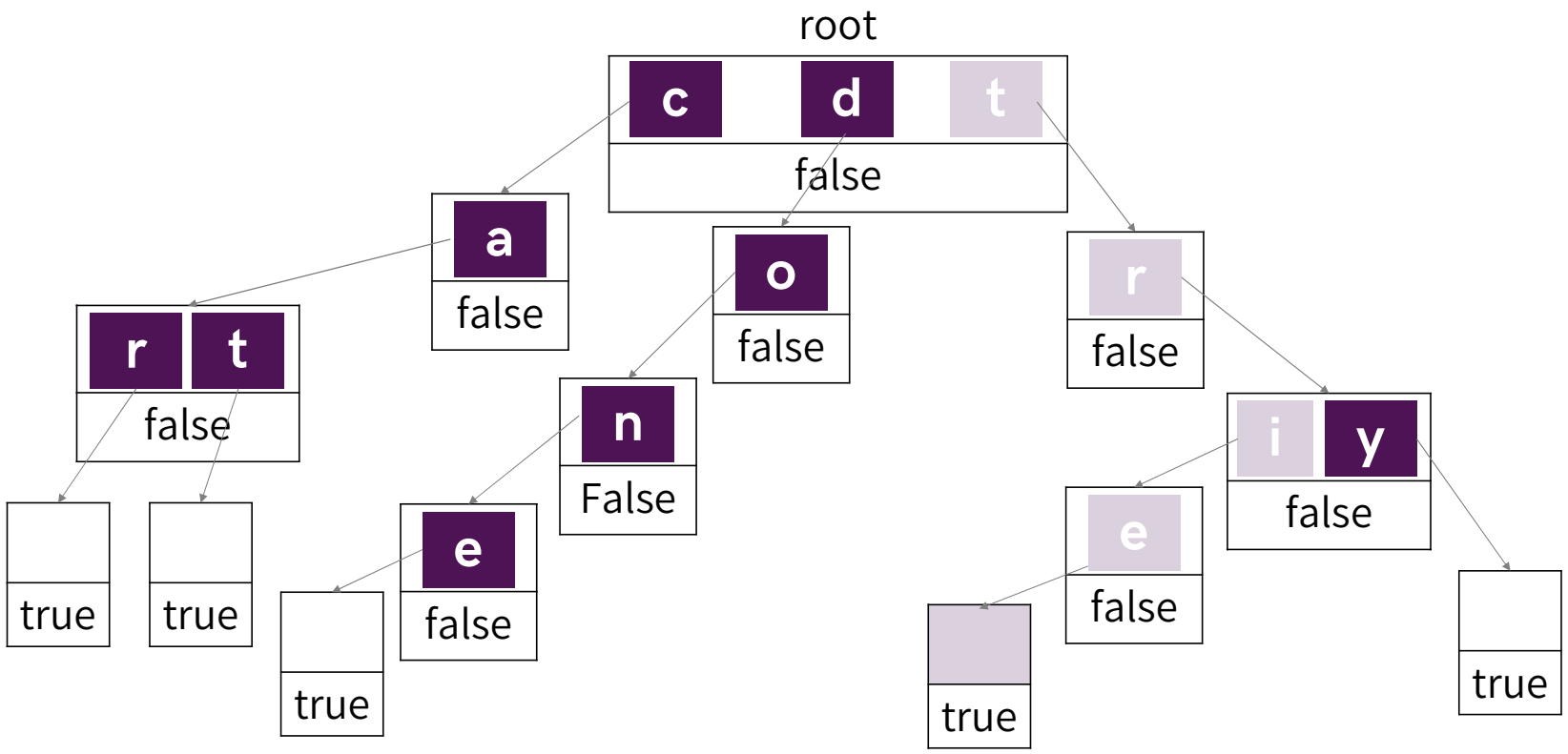
Trie Operations - Delete

trie
ca



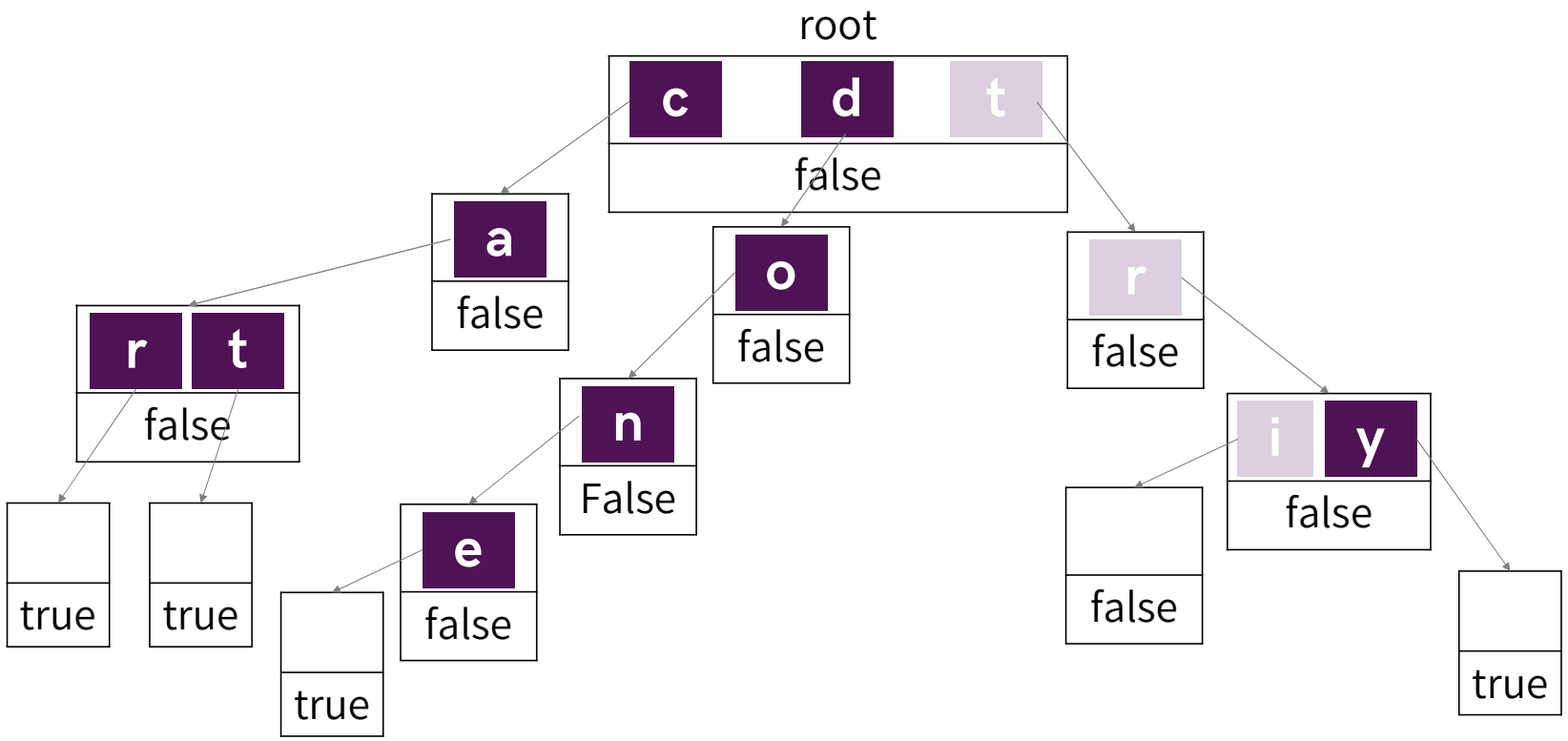
Trie Operations - Delete

trie
ca



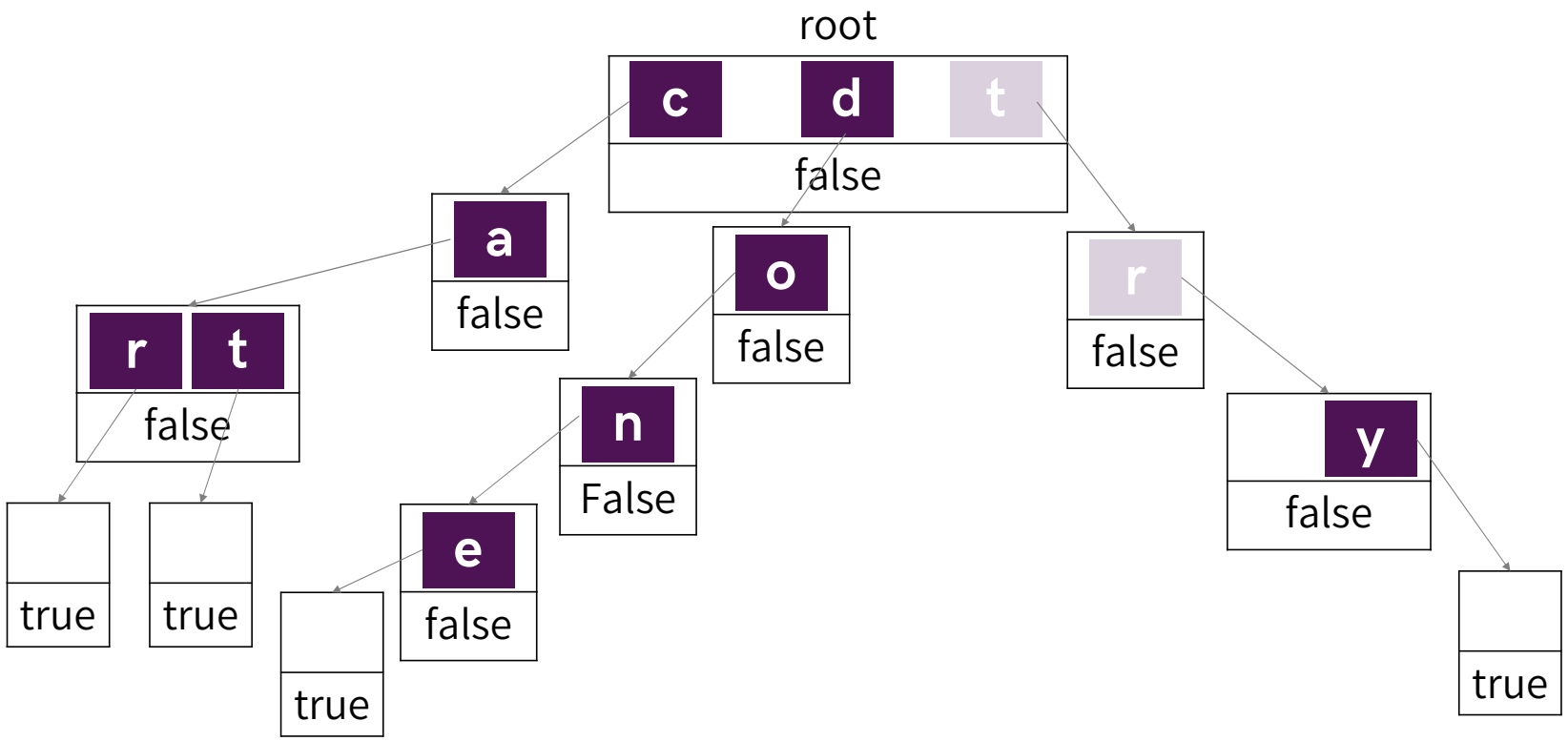
Trie Operations - Delete

trie
ca



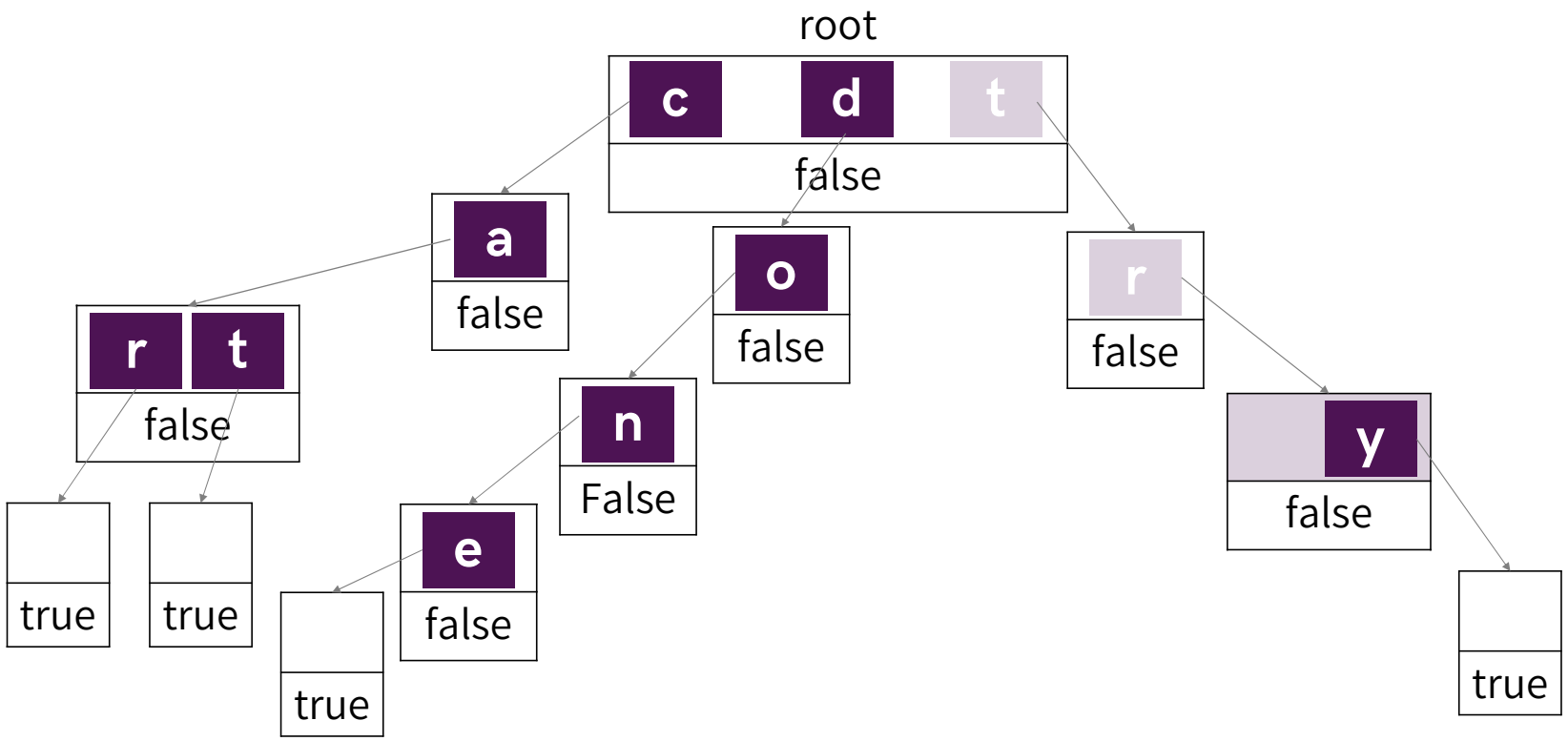
Trie Operations - Delete

trie
ca



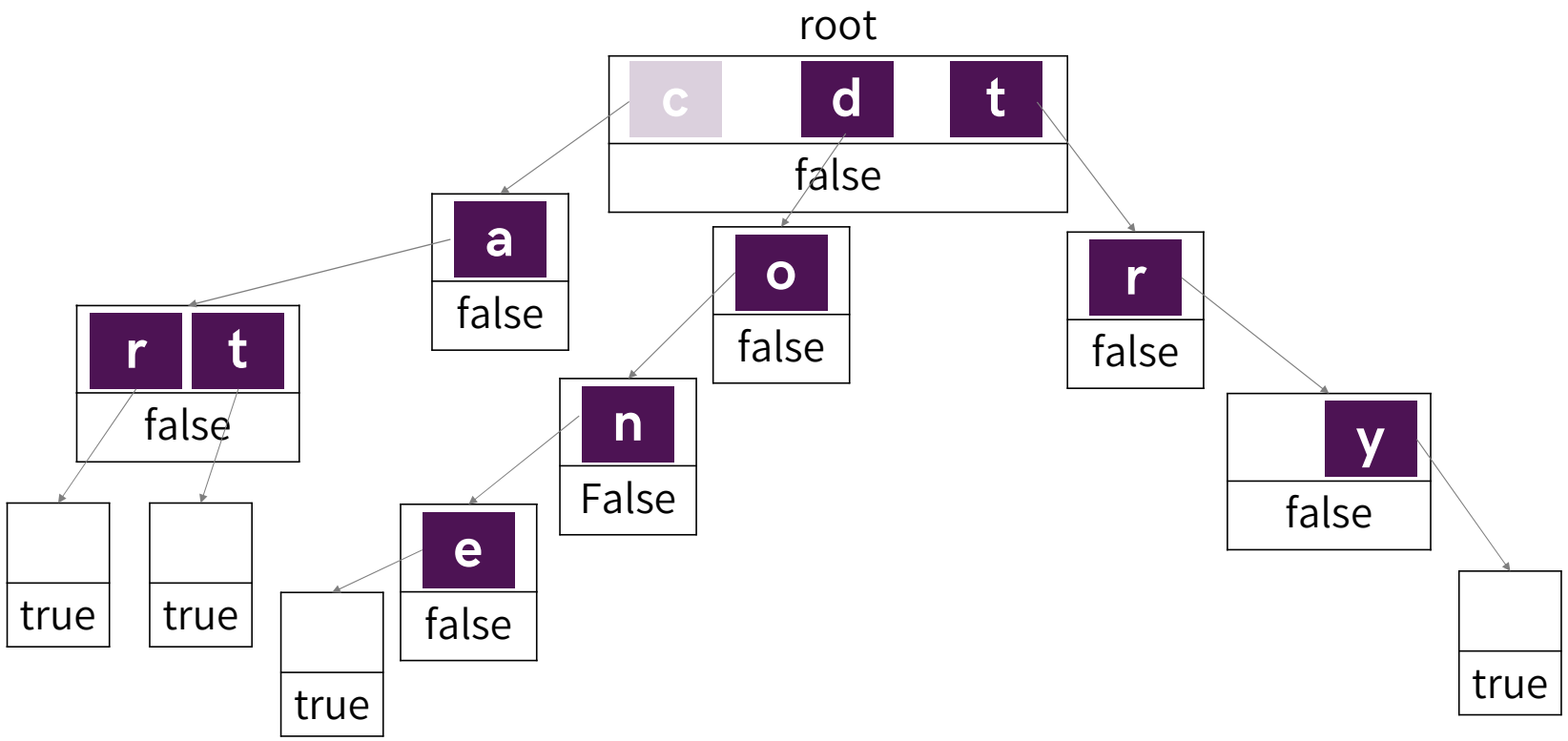
Trie Operations - Delete

trie
ca



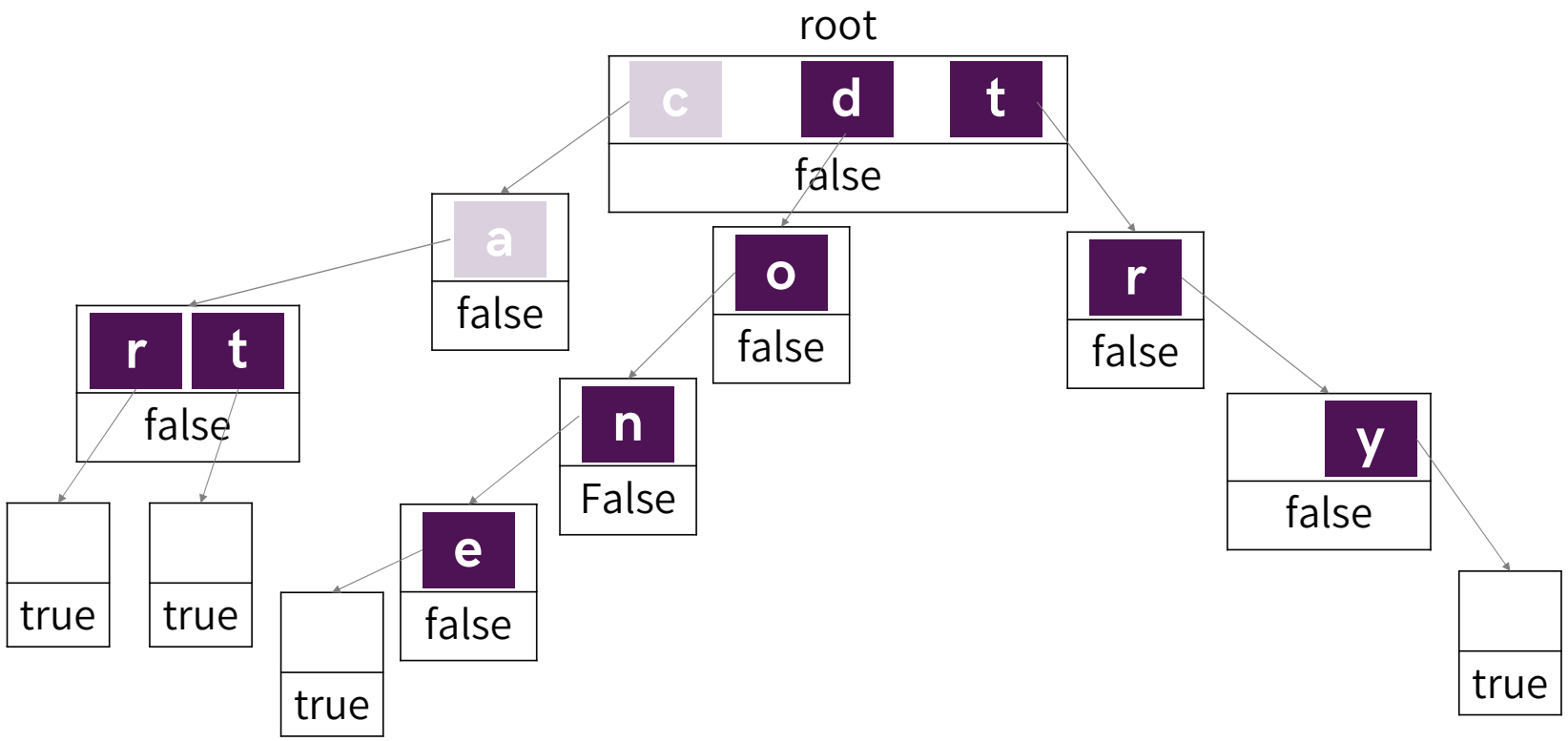
Trie Operations - Delete

ca



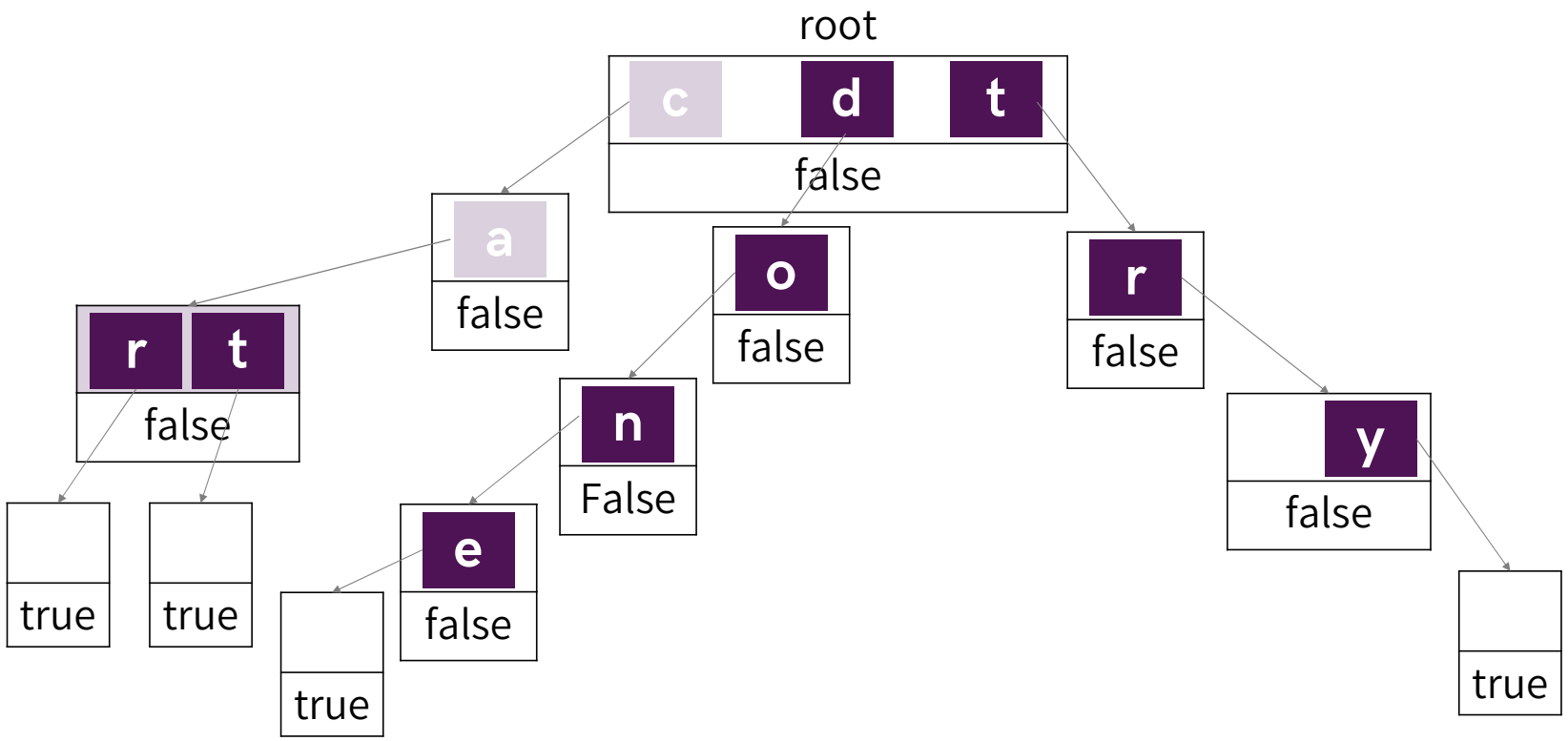
Trie Operations - Delete

ca



Trie Operations - Delete

ca



Alphabet Trie

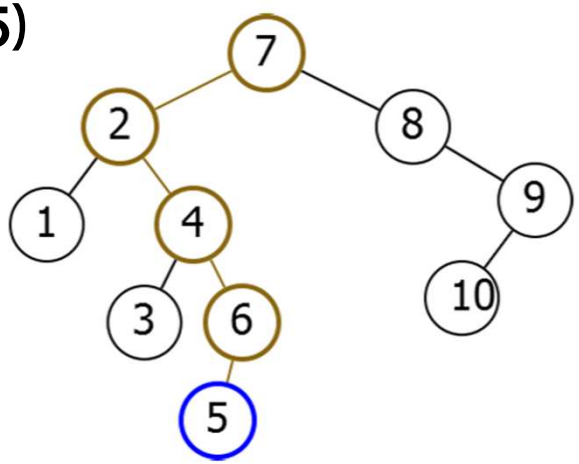
- To determine if a given word is in the dictionary
- Only the letter that leads to a word are shown as branches
- Node's depth depends on the number of character required to distinguish this node from any other
- Implementation:
  - Array of 27 pointers indexed by letter
  - A linked list of pointers to the child nodes

Balanced Tree

BST Runtime

How long do Binary Search Tree operations take?

Find(5)

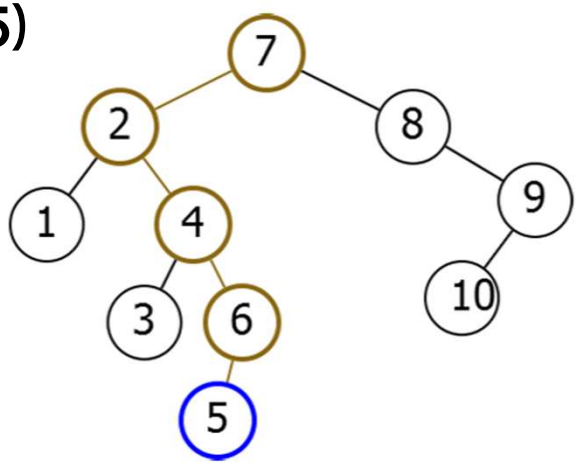


Number of operations=O(Depth) → Depth can be as bad as **n**

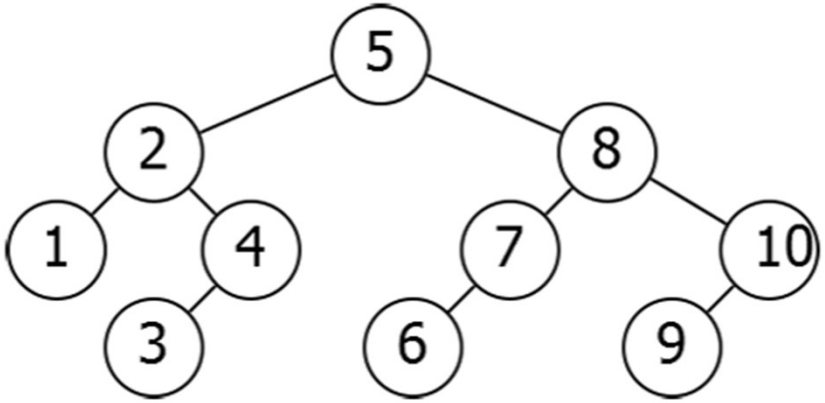
BST Runtime

How long do Binary Search Tree operations take?

Find(5)



Number of operations=O(Depth)



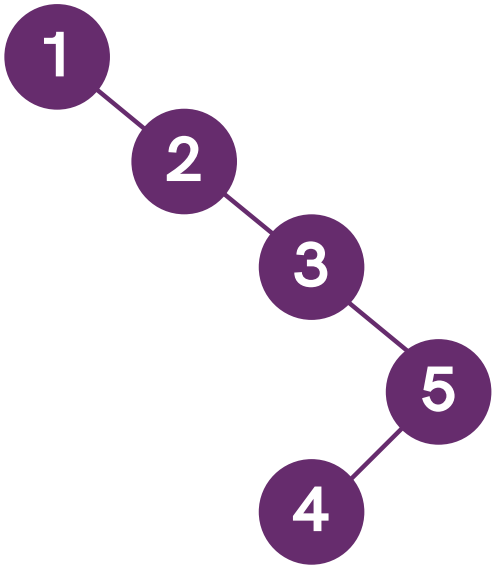
Depth can be much smaller

Balance

- Want left and right subtrees to have approximately the same size.
- Suppose perfectly balanced:
  - Each subtree half the size of its parent.
  - After log2(n) levels, subtree of size 1.
  - Operations run in O(log(n)) time.

Problem

Insertion and Deletion can destroy balance



Rebalancing

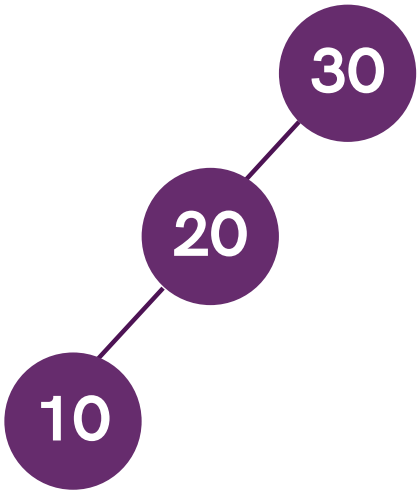
Idea: Rearrange tree to maintain balance.

Problem: How do we arrange tree while maintaining order?

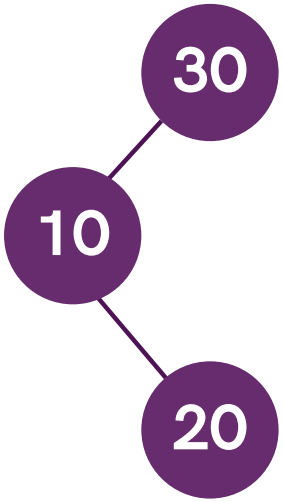
Rotation

Keys: 30,20,10

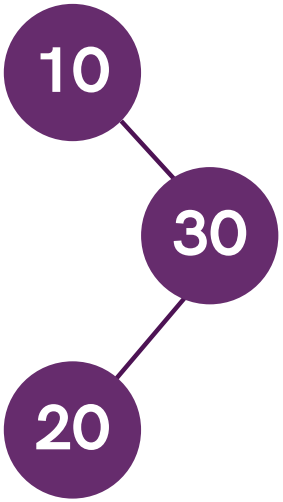
Orders: 30,20,10



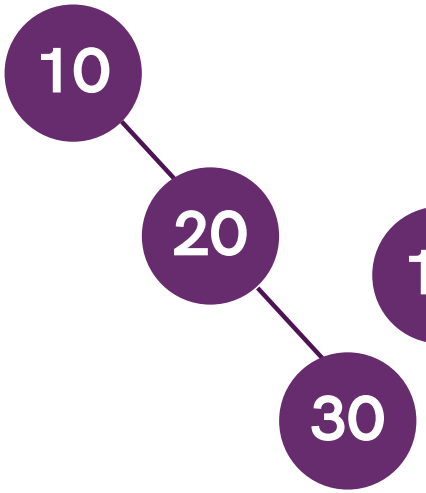
30,10,20



10,30,20

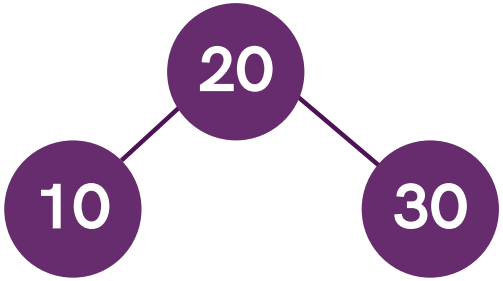


10,20,30



20,10,30

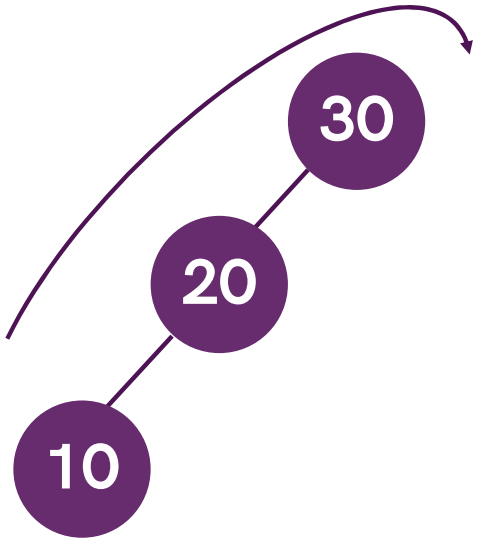
20,30,10



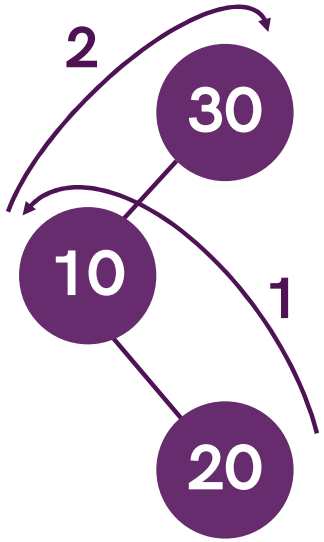
Rotation

Keys: 30,20,10

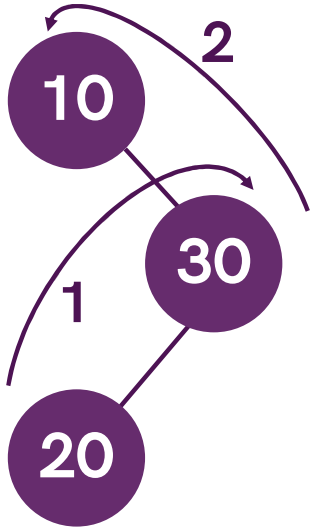
Orders: 30,20,10



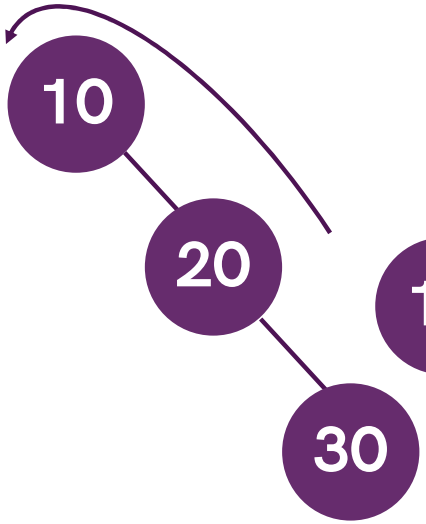
30,10,20



10,30,20

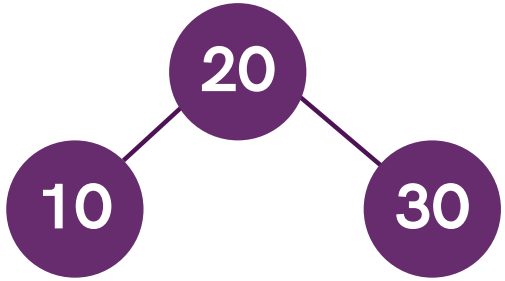


10,20,30



20,10,30

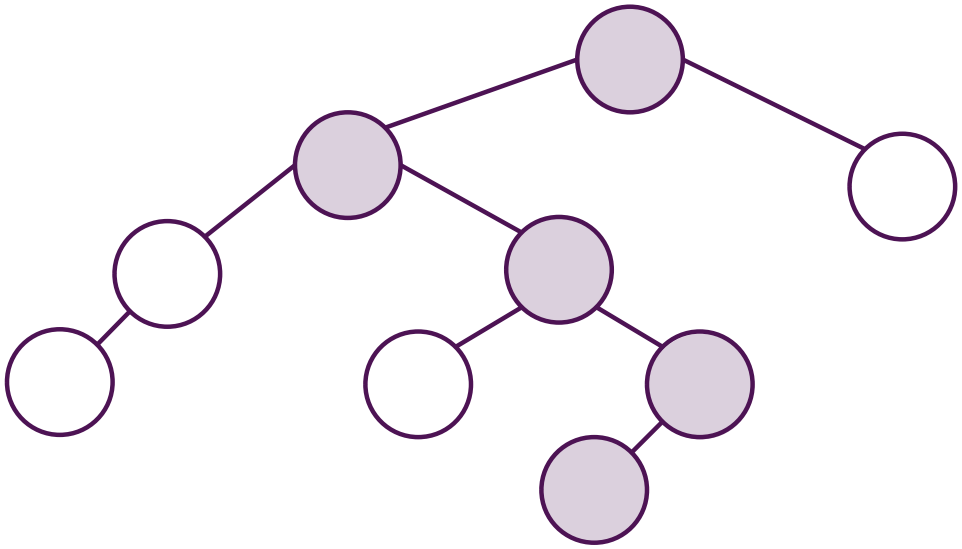
20,30,10



AVL Tree

Height

The **height** of a node is the maximum depth of its subtree.



Height = 4

Height – Recursive Definition

N.Height equals:

- 1 if N is a leaf,
- 1+max(N.Left.Height,N.Right.Height) otherwise

Balance

- Height is a rough measure of subtree size.
- Want size of subtrees roughly the same.
- Force heights to be roughly the same.

AVL Tree (Adelson-Velskii and Landis)

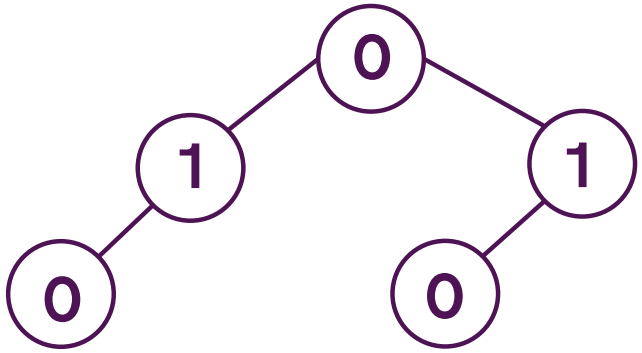
A BST with the following additional property: for every node N, the height of its left and right subtrees differ by at most 1.

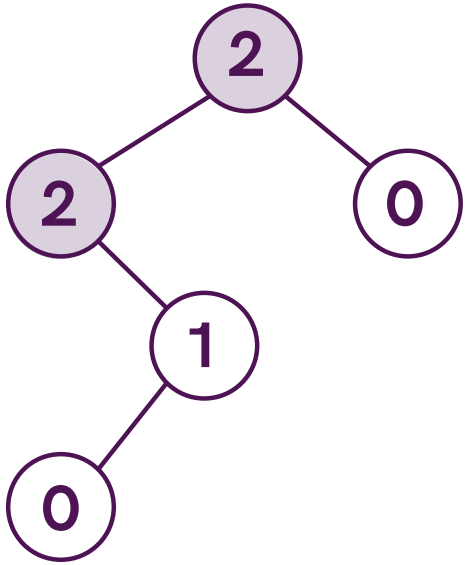
$$|N.Left.Height - N.Right.Height| \leq 1$$

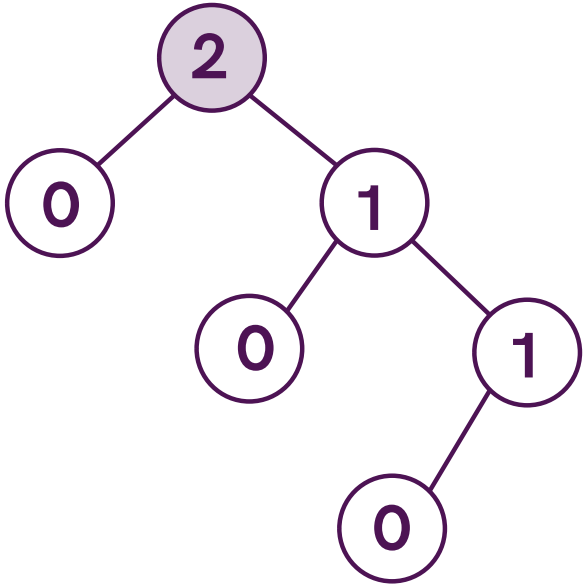
If you can maintain the AVL property, you can perform operations in **O(log(n))** time.

Balance

$$|N.Left.Height - N.Right.Height| \leq 1$$





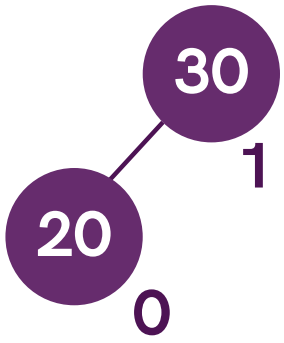


Unbalanced Cost

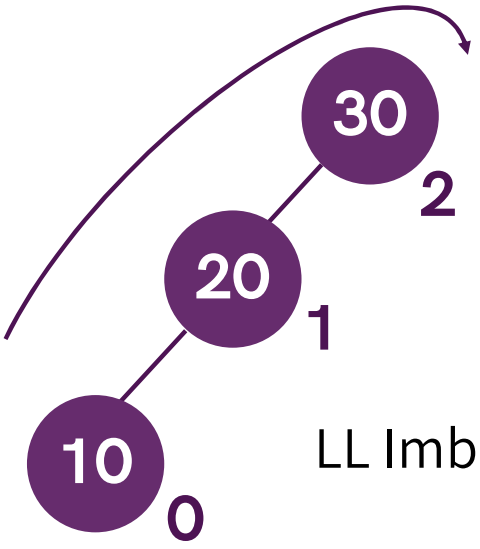
1. The extra node is in the left child of the left child of S (LL) ➡ Single Rotation
2. The extra node is in the right child of the left child of S (RL) ➡ Double Rotation
3. The extra node is in the left child of the right child of S (LR) ➡ Double Rotation
4. The extra node is in the right child of the right child of S (RR) ➡ Single Rotation

Single Rotation

Initially

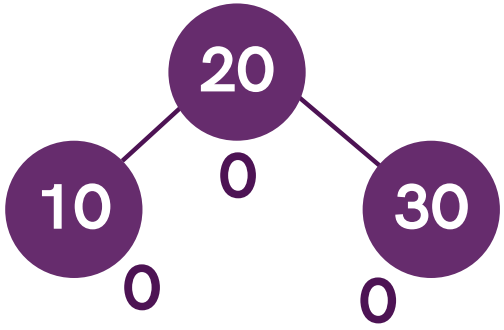


Insert 10



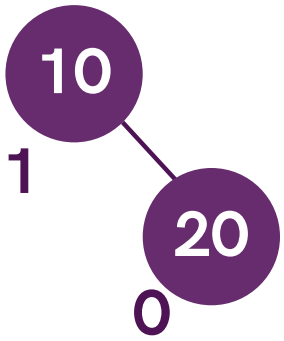
LL Imbalance

After Rotation

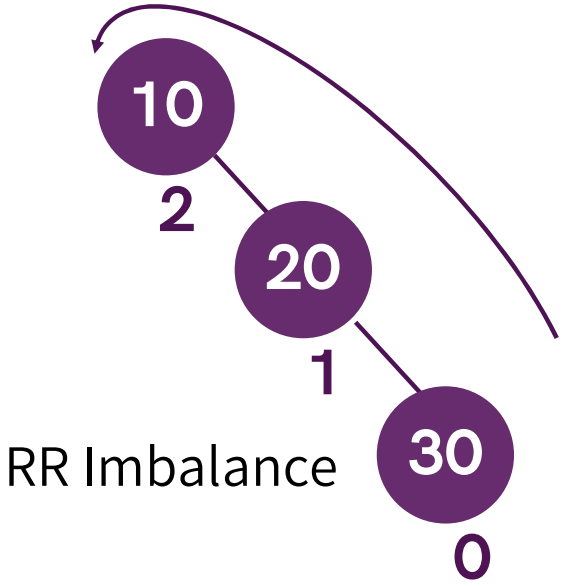


Single Rotation

Initially

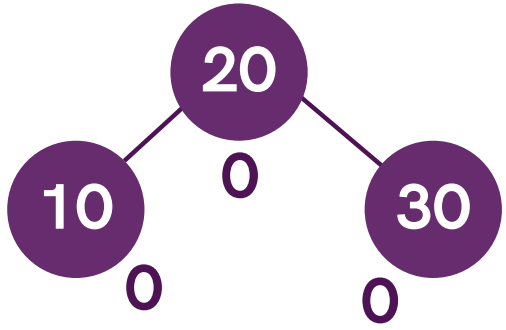


Insert 30

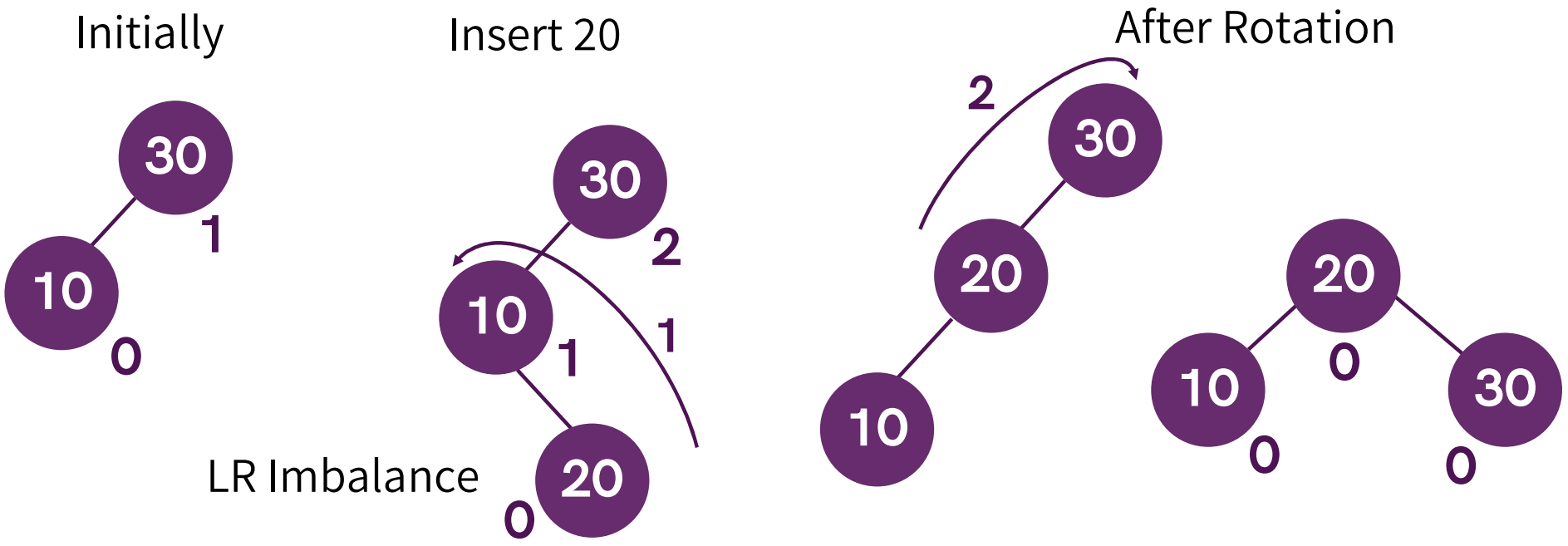


RR Imbalance

After Rotation

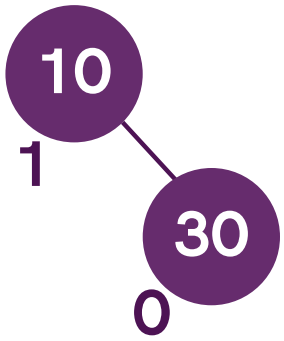


Double Rotation

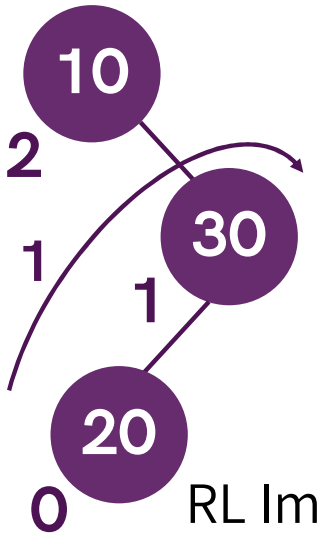


Double Rotation

Initially

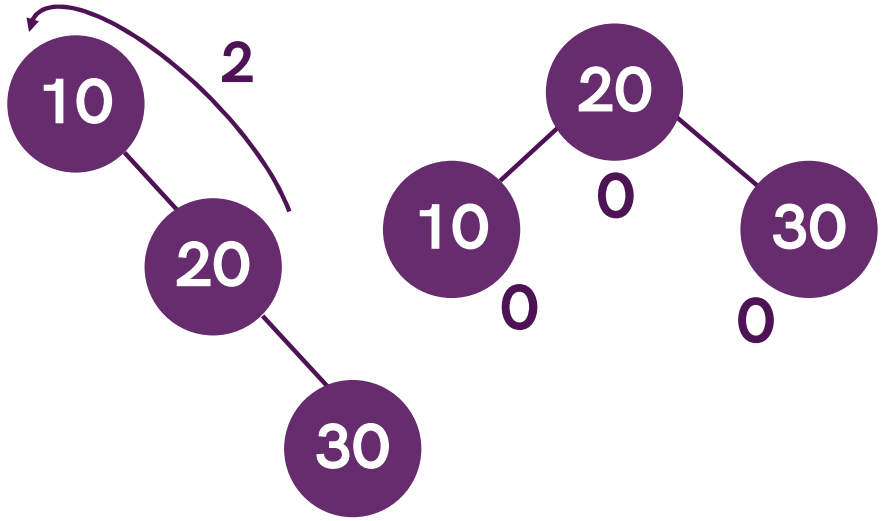


Insert 20

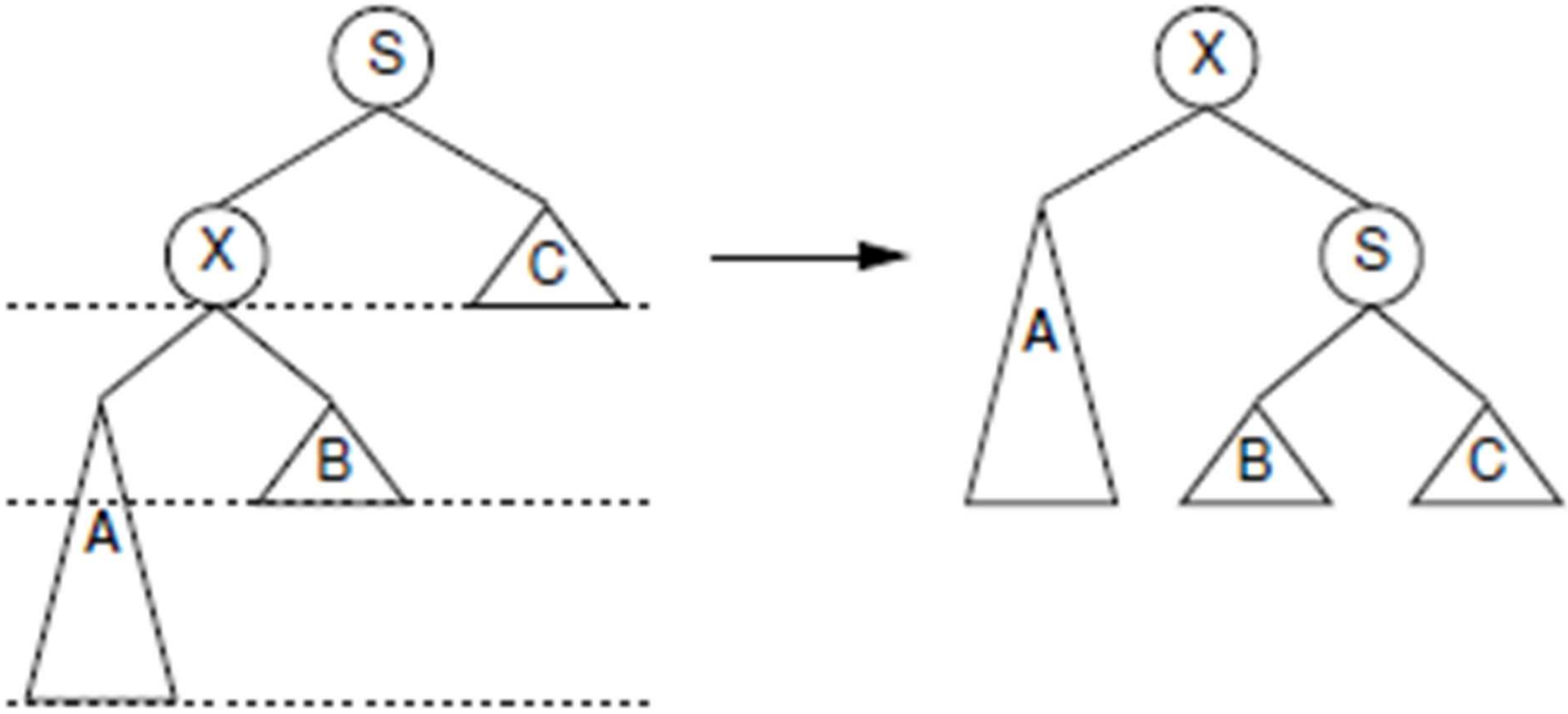


RL Imbalance

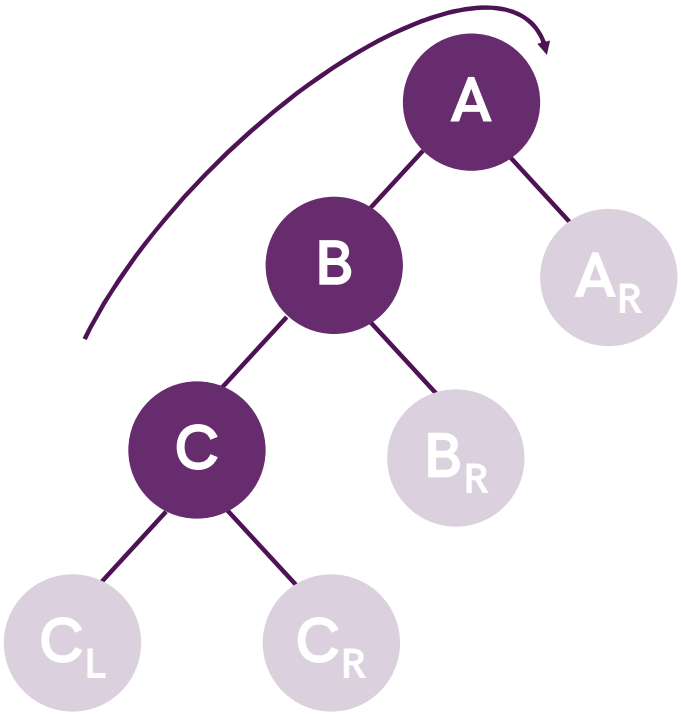
After Rotation

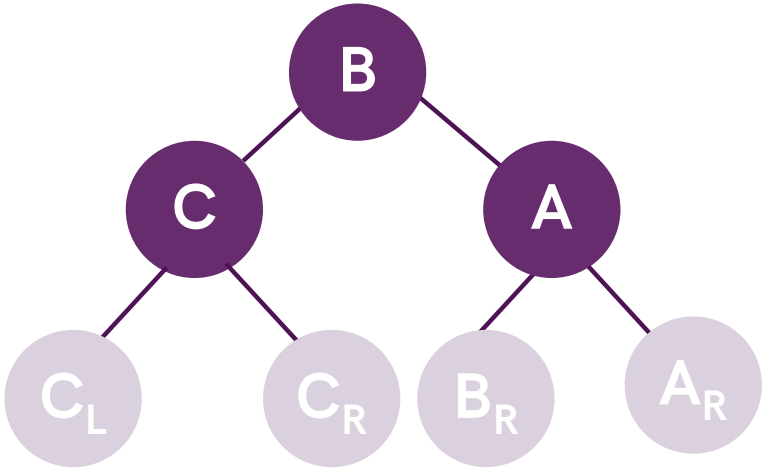


Single Rotation

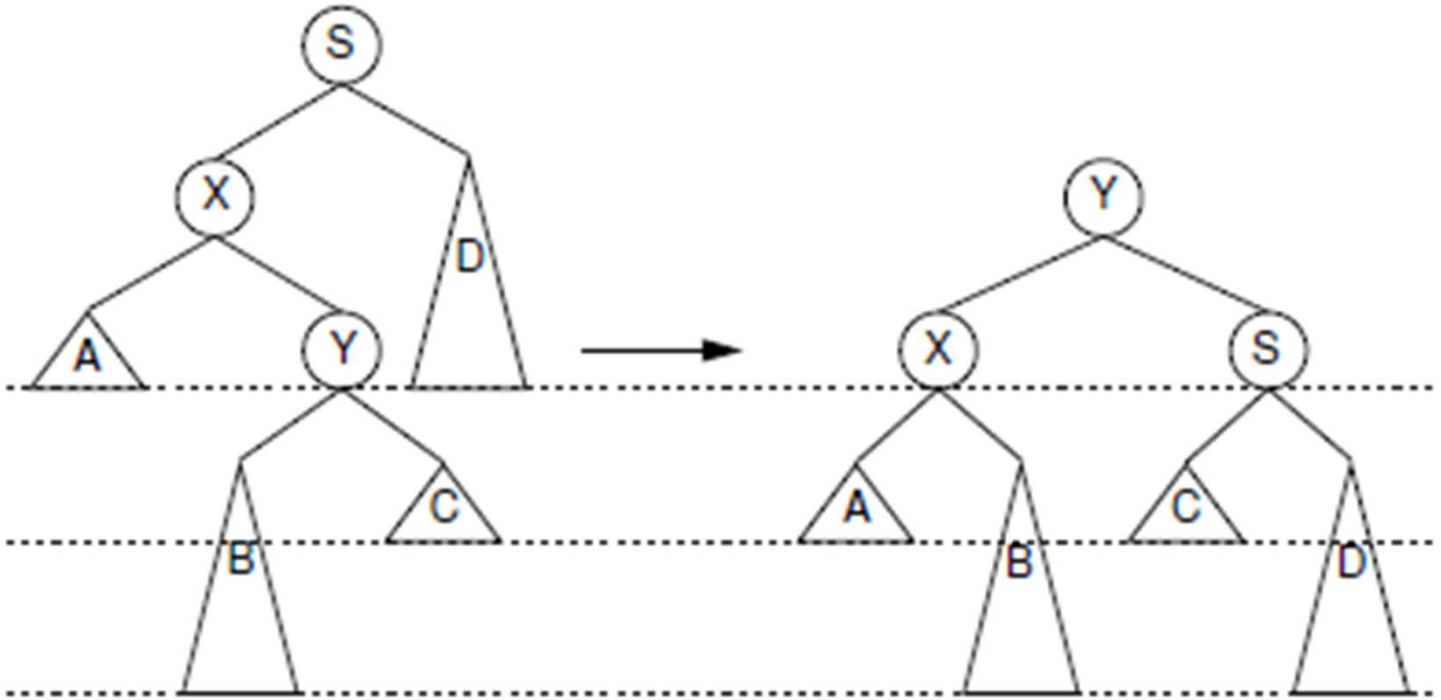


Single Rotation

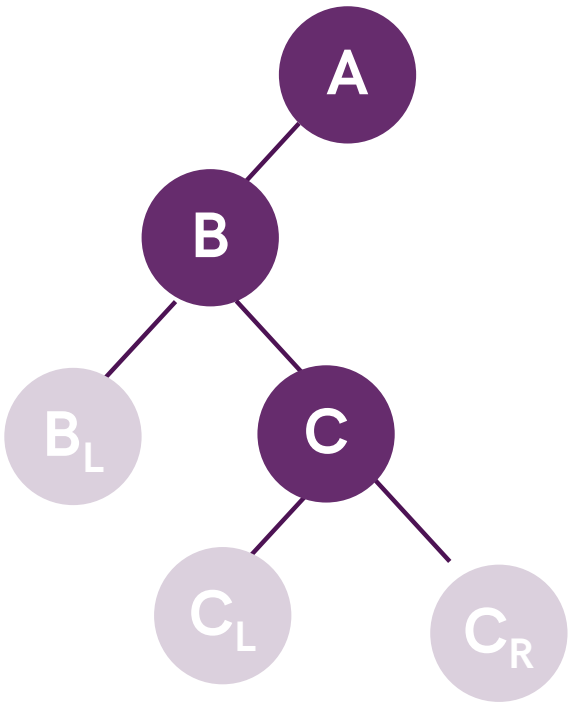


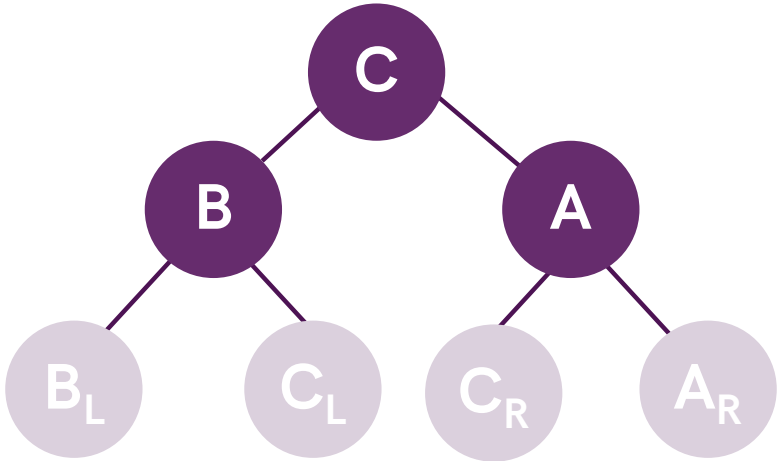


Double Rotation



Double Rotation



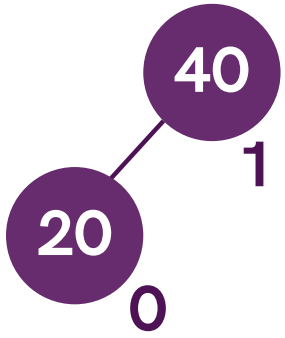


AVL Tree Implementation

- Normal BST insert/delete
- Perform appropriate rotation on any node that is found to be unbalanced

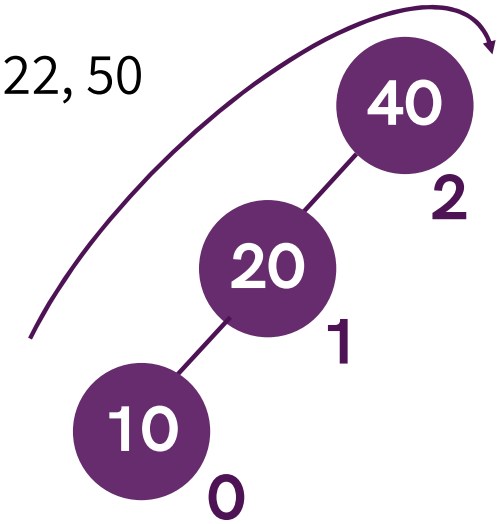
Example

Keys: 40, 20, 10, 25, 30, 22, 50



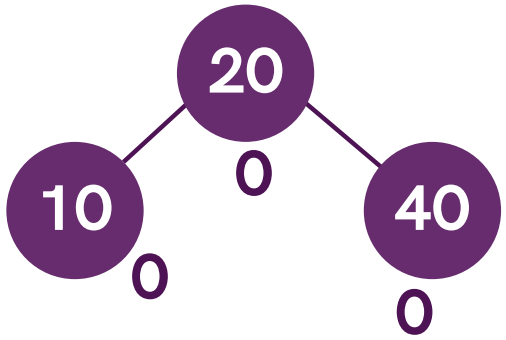
Example

Keys: 40, 20, 10, 25, 30, 22, 50



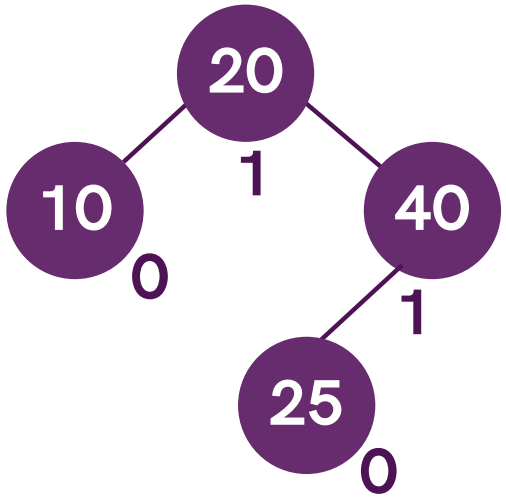
Example

Keys: 40, 20, 10, 25, 30, 22, 50



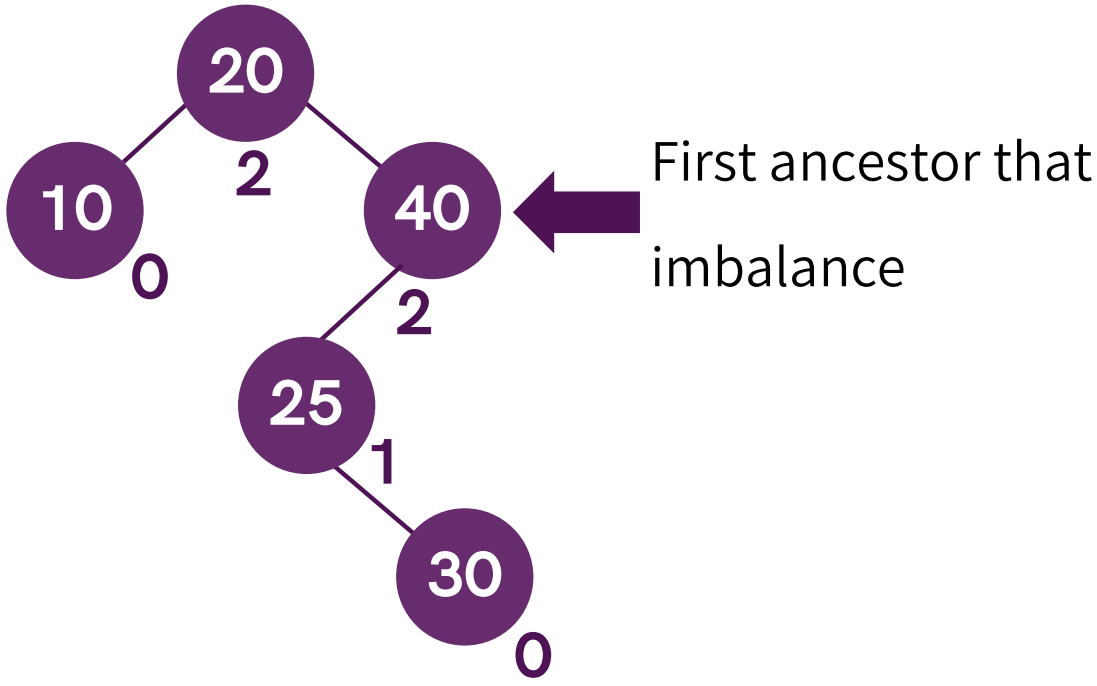
Example

Keys: 40, 20, 10, 25, 30, 22, 50



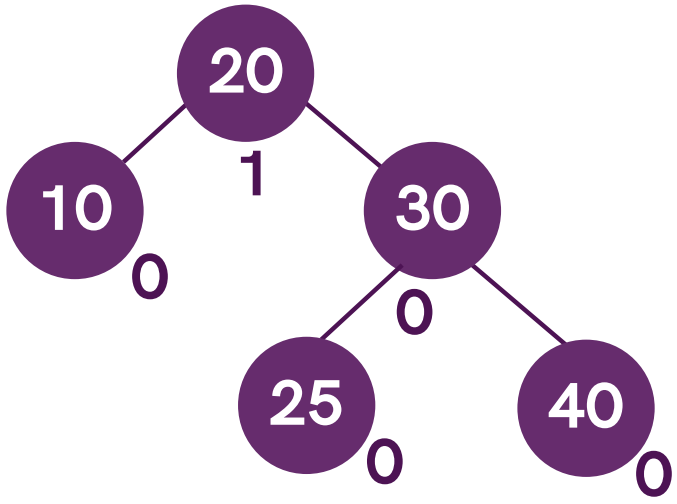
Example

Keys: 40, 20, 10, 25, 30, 22, 50



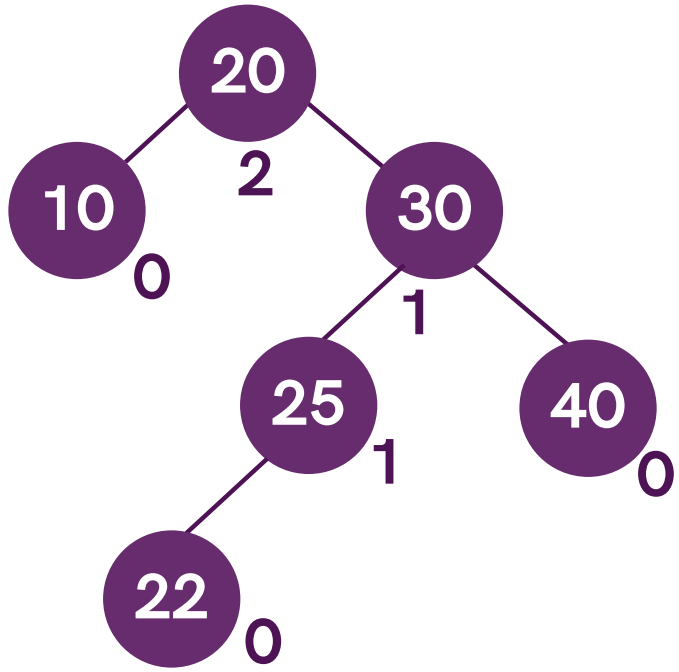
Example

Keys: 40, 20, 10, 25, 30, 22, 50



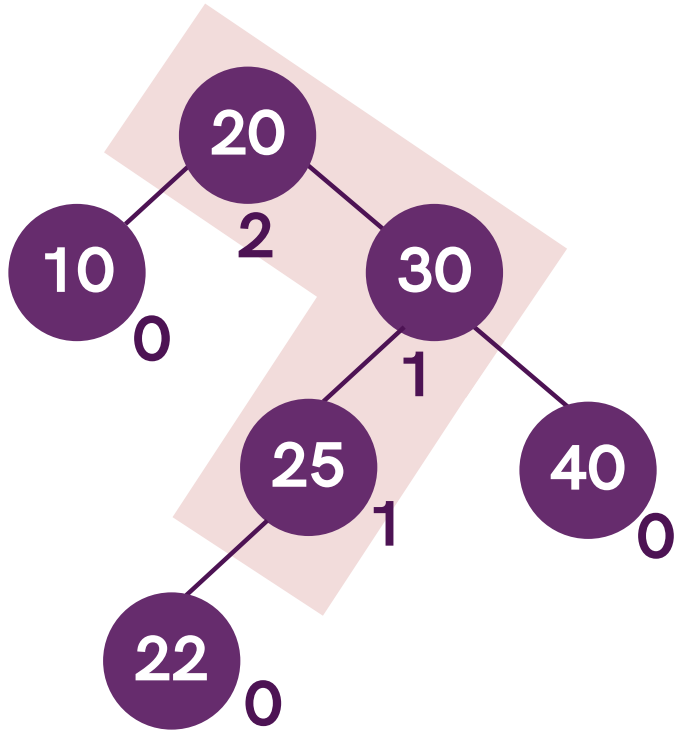
Example

Keys: 40, 20, 10, 25, 30, 22, 50



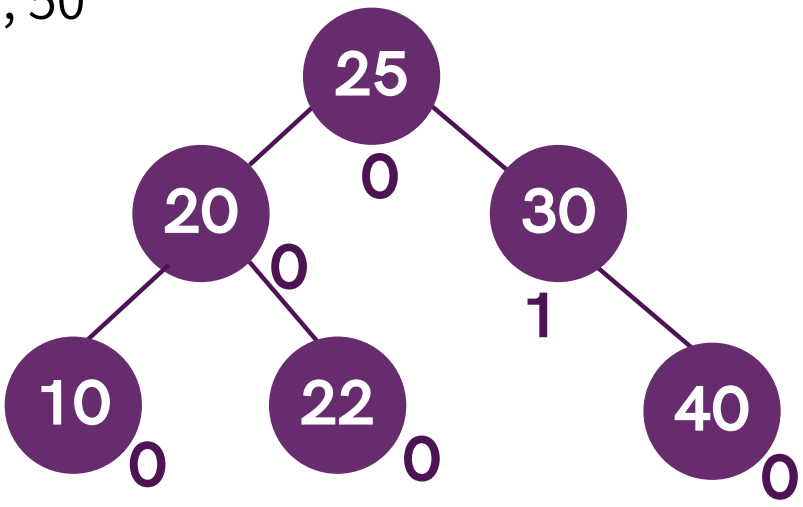
Example

Keys: 40, 20, 10, 25, 30, 22, 50



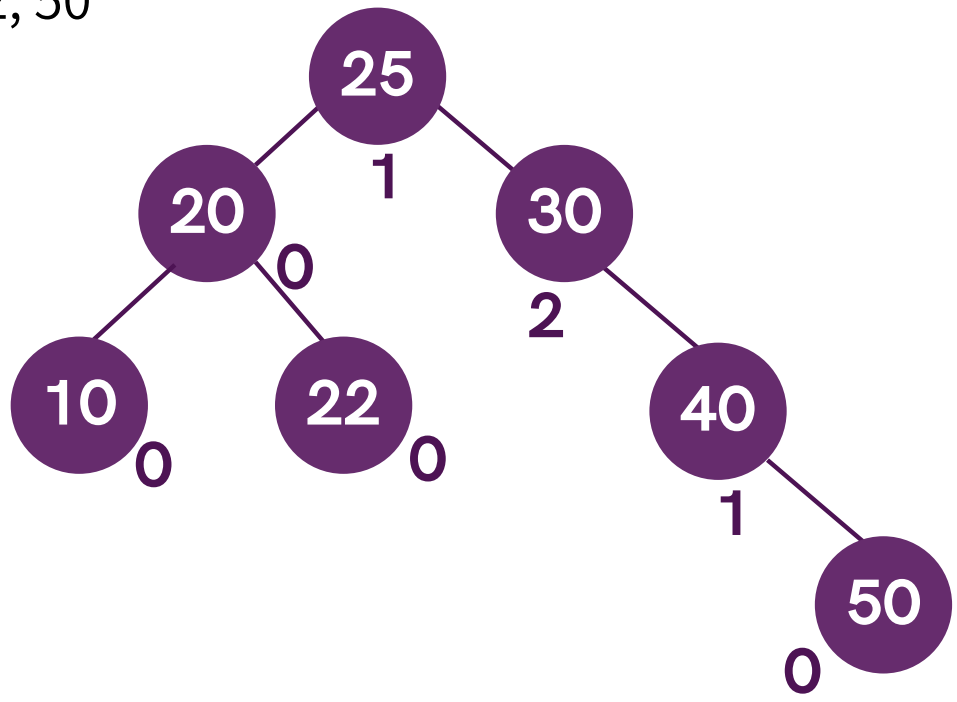
Example

Keys: 40, 20, 10, 25, 30, 22, 50



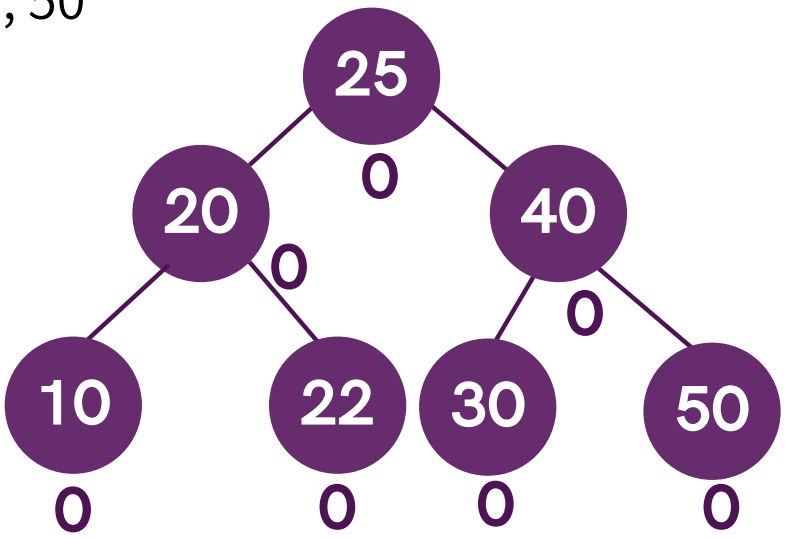
Example

Keys: 40, 20, 10, 25, 30, 22, 50



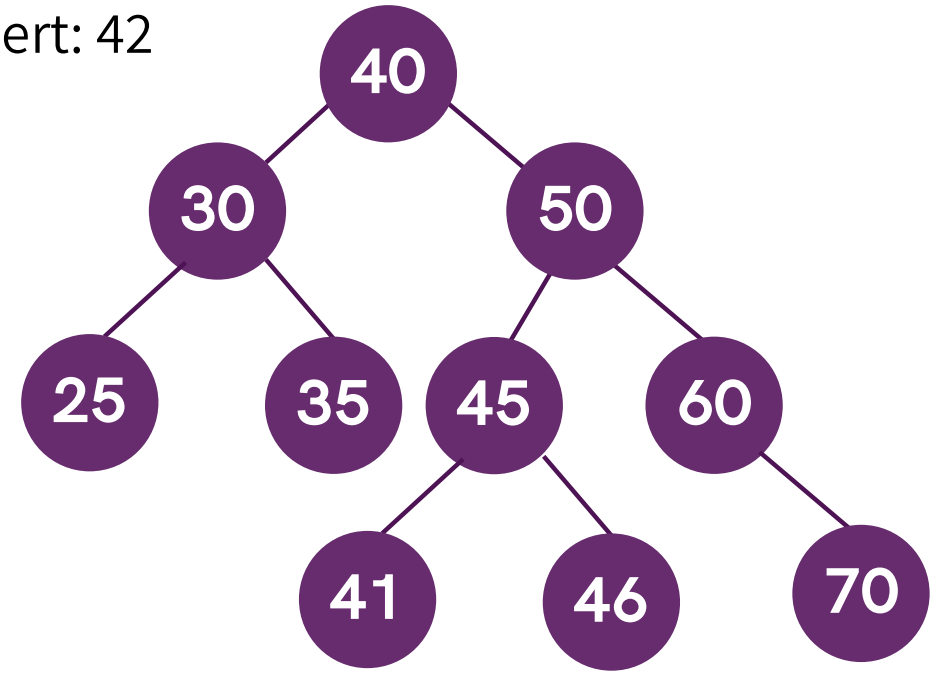
Example

Keys: 40, 20, 10, 25, 30, 22, 50



Latihan Soal

Insert: 42



Tugas 4

Pemilik kedai kopi menyimpan data menu agar mudah dicari oleh pelanggan. Berikut data menu yang ingin disimpan:

Café Long Black
Café Cappuccino
Café Oat Cappuccino
Café Mocha
Café Latte
Vanilla Latte
Caramel Latte
Hazelnut Latte
Cinnamon Coffee Latte
Café Espresso
Café Macchiato
Café Piccolo
Café Flat White
Babyccino

1. Buatlah program untuk menyimpan data menu menggunakan trie!
2. Buatlah program untuk mencari apakah suatu item ada dalam menu!
3. Buatlah program untuk menghapus menu tertentu dari trie!

Terapkan *modular programming* dalam mengimplementasikan program