Trie in Substrate

Yuanchao Sun | Cdot Network

Outline

Warm up with a bunch of data structures

Think about blockchain storage implementation

Merkle patricia trie in substrate

Warm up with a bunch of data structures

Trie

Also called digital tree or prefix tree

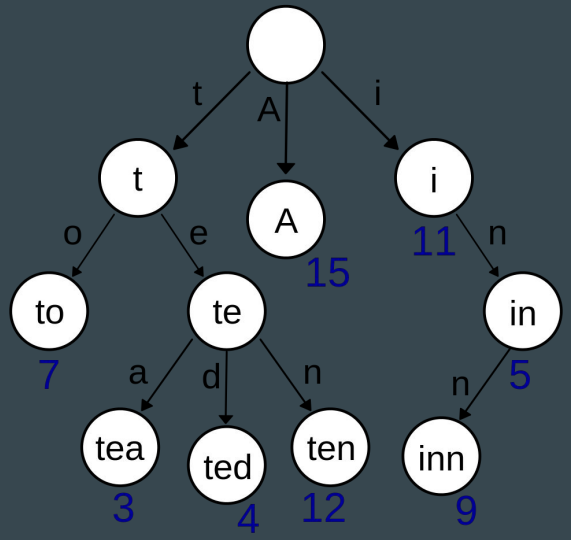
First described by René de la Briandais in 1959, Named by E. Fredkin in 1960

Information retrieval

Pronounced “try”

Insertion, lookup O(k)

Key	Value
A	15
to	7
tea	3
ted	4
ten	12
inn	9



Patricia trie

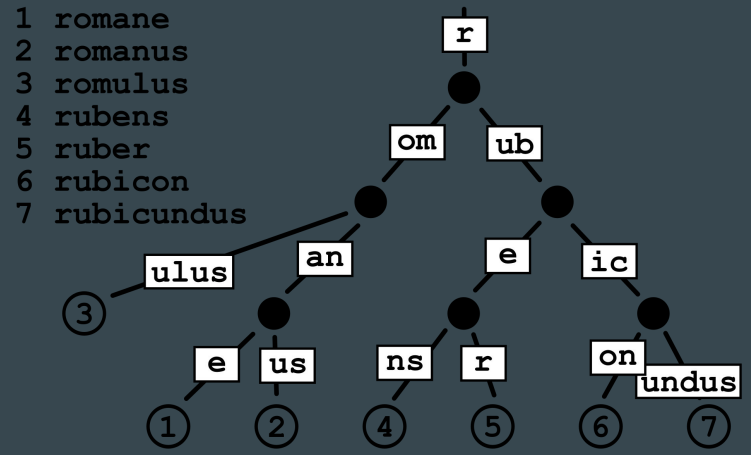
Also called radix trie or compact prefix tree

Discovered by Donald R. Morrison and G. Gwehenberger in 1968

Practical Algorithm To Retrieve Information Coded In Alphanumeric

Edge node and leaf node

Lookup, insertion, and deletion in O(k)



Merkle tree

Also called hash tree

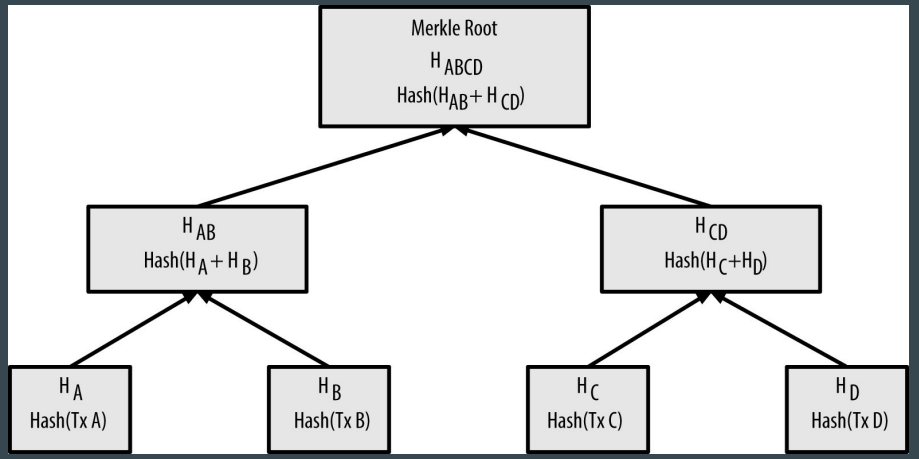
Patented by Ralph Merkle in 1979

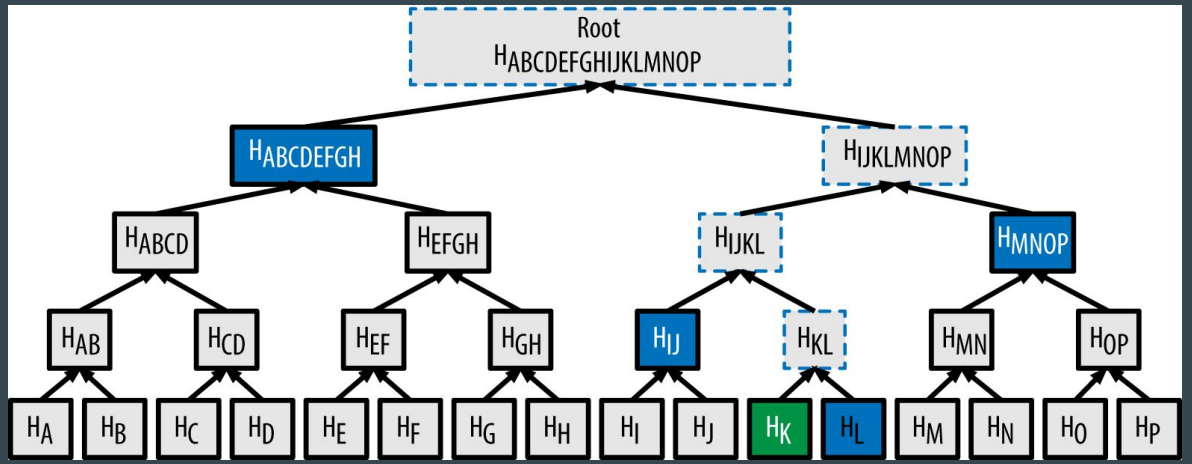
Hash(data) = 0xf904948a ...

Merkle root, branch and leaf

To save disk space

Merkle proof





Think about blockchain storage implementation

On the blockchain side

Blockchain is about to come to agreement on the same data

Calculate a state of all the data and forward it to others

Others can verify the state cheaply

Solution: merkle tree

On the engineering side

The data to be stored is a mapping of structure of a programming language

The structure needs to be serializable into a byte array

Need a unified key to locate the serialized structure

Arbitrary key and value length

Solution: patricia trie

Merkle patricia trie in substrate

Primitives of substrate storage

facebook/rocksdb: key/value database

BLAKE2-256: hash function

paritytech/parity-scale-codec: binary serialization and deserialization codec

paritytech/trie: Base-16 Modified Merkle Tree

Path generator: StorageValue: twox128(module_prefix) ++ twox128(storage_prefix)

Database schema

Hash	Vec<u8>
0x9a7aa12d ...	vec![99, 100, 111, 116, 45, ...]
0xd9a68a35 ...	vec![110, 101, 116, 119, 111, 114, 107, ...]

Node Kind

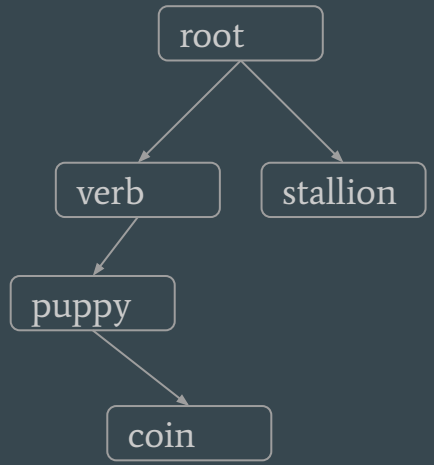
NodeKind	Prefix
Empty	00
Leaf	01
BranchNoValue	10
BranchWithValue	11

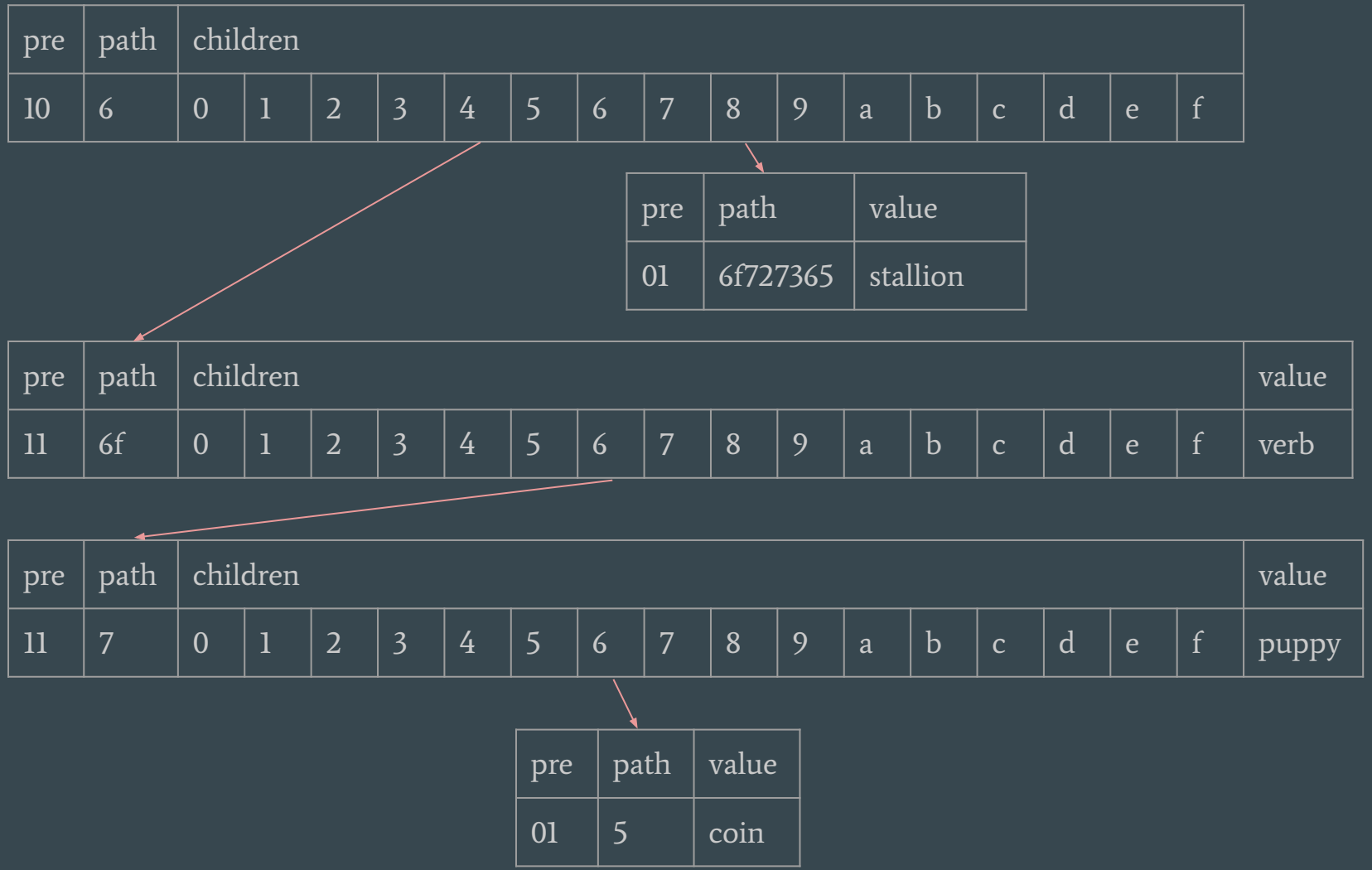
Node Structure

NodeKind	Path	Children	Value

A trie containing four key/value pairs

Key	Path	Value
do	<64 6f>	verb
dog	<64 6f 67>	puppy
doge	<64 6f 67 65>	coin
horse	<68 6f 72 73 65>	stallion




pre
path
children
10
6
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
pre
path
value
01
6f727365
stallion
pre
path
children
value
11
6f
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
verb
pre
path
children
value
11
7
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
puppy
pre
path
value
01
5
coin

Question?