🌳 AVL Tree Storage in Gno

Efficient, deterministic smart contract storage

Why AVL Trees?

- 🔥 Self-balancing binary search trees
- ✅ Deterministic and reproducible
- 📈 Efficient lookups, insertions, deletions

> Gno uses AVL trees to **guarantee state consistency**.

How AVL Trees Work

- Every **node stored by hash** 📦
- **State updates** = creating new tree versions
- Ensures **immutability** and **verifiability**

```go
type Tree struct {
  root *Node
}
```

Storage Layer in Gno

```
store package = Abstracts storage

Handles AVL tree structure under the hood

Powers smart contract state management
```

Quick Example

```go
tree := NewTree()
tree.Set([]byte("foo"), []byte("bar"))
value, _ := tree.Get([]byte("foo"))
fmt.Println(string(value)) // bar
```

Simple key-value store backed by AVL logic

Why It Matters

🌐 Every contract's state = AVL tree

🔒 Trustless, deterministic execution

🧠 Enables blockchain reproducibility

🚀 Next Steps

Learn how to query and update your smart contract state

Dive deeper into storage proofs and verifications

Thanks for Learning! 🙏

AVL Trees = 🔥 for blockchain scalability

Follow up: 👉 "How to interact with Gno smart contract storage"