MATA54-T01-ESTRUTURAS DE DADOS E ALGORITMOS II

Merkle tree

Equipe: Djair Maykon e Tauane Sales

Introdução

O que são Hash Tree Data Structures?

Uma hash tree, ou árvore de Merkle, é uma estrutura de dados hierárquica em forma de árvore binária na qual cada nó folha contém o hash (resultado de uma função criptográfica) de um bloco de dados. Já cada nó não-folha contém o hash combinado dos hashes de seus nós filhos. Essa estrutura gera uma “raiz” (Merkle Root) que resume toda a integridade dos dados contidos na árvore. Em outras palavras, qualquer alteração em um bloco (ou folha) resulta em uma mudança na raiz, permitindo a detecção rápida de inconsistências.

Introdução

Por que elas são usadas?

- **Verificação de integridade:** Permitem confirmar de forma eficiente se os dados não foram alterados.
- **Escalabilidade:** Comprovam a integridade de conjuntos de dados grandes sem necessidade de transmitir todos os dados.
- **Eficiência:** Através de provas compactas, apenas uma pequena parte (os hashes dos irmãos do caminho) precisa ser verificada para confirmar a integridade de um dado.
- **Aplicações críticas:** São essenciais em sistemas distribuídos, blockchains, controle de versões distribuídas (como no Git) e em processos de sincronização de dados.

Casos de Uso

Data Verification (Verificação de Dados):

Merkle Trees geram uma raiz (Merkle Root) que resume todo o conjunto de dados. Ao verificar essa raiz contra a esperada, é possível detectar alterações ou corrupções sem precisar reprocessar todos os dados.

Data Synchronization (Sincronização de Dados):

Em sistemas distribuídos, Merkle Trees permitem comparar rapidamente conjuntos de dados. Apenas os ramos divergentes são sincronizados, reduzindo o tráfego e acelerando a reconciliação dos dados.

Casos de Uso

Sistemas de Armazenamento Distribuído:

Essas árvores garantem que réplicas armazenadas em diferentes nós sejam consistentes, pois cada nó pode verificar localmente a integridade dos dados. Essa verificação rápida ajuda a identificar discrepâncias e a manter a confiabilidade do sistema.

Blockchain

Blockchains organizam transações em uma Merkle Tree, cuja raiz é armazenada no cabeçalho do bloco. Isso permite que os nós verifiquem a integridade das transações sem precisar acessar o bloco completo.

E no mundo real?

Bitcoin Blockchain

No Bitcoin, cada bloco contém uma Merkle Tree que agrupa todas as transações realizadas. Os nós podem verificar se uma transação específica está incluída no bloco usando uma prova de ramo, sem precisar baixar todo o bloco. Essa abordagem é essencial para garantir a segurança e a descentralização do sistema, detectando qualquer alteração indesejada.

E no mundo real?

Git

O Git utiliza uma árvore de Merkle para armazenar commits, onde cada commit é identificado por um hash que depende do conteúdo atual e dos commits anteriores. Essa estrutura assegura a imutabilidade e a integridade da história do repositório, permitindo que qualquer modificação seja detectada imediatamente. Dessa forma, a confiança no controle de versões é mantida.

Outras Estruturas de Verificação de Integridade

Hash List

Uma hash list é uma estrutura linear onde cada elemento é processado por uma função hash, formando uma sequência encadeada de hashes. Qualquer alteração em um elemento altera todos os hashes subsequentes, facilitando a detecção de inconsistências.

Verkle Tree

Uma verkle tree é uma evolução da Merkle Tree que utiliza compromissos vetoriais para reduzir o tamanho das provas. Ela gera provas mais compactas e eficientes, ideal para aplicações escaláveis como blockchains.

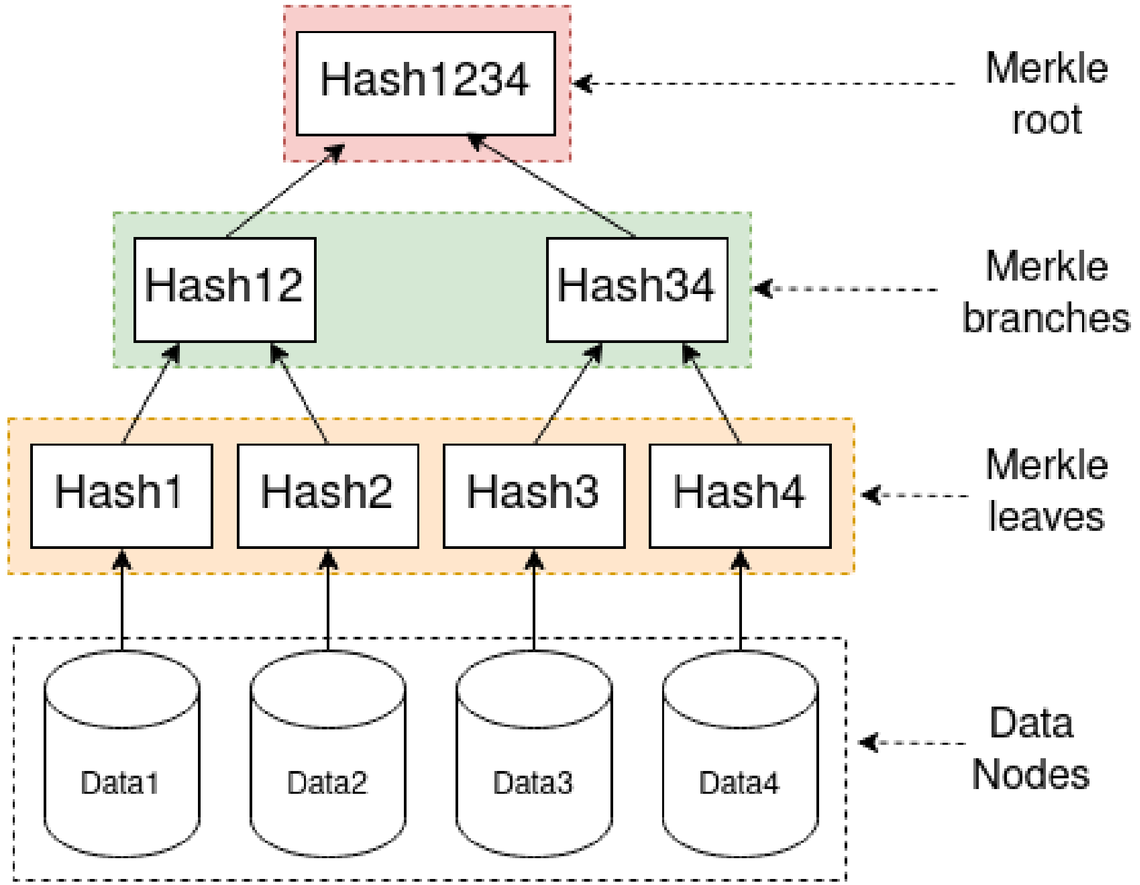
Funcionamento

Uma Merkle Tree é uma estrutura hierárquica em forma de árvore binária, onde:

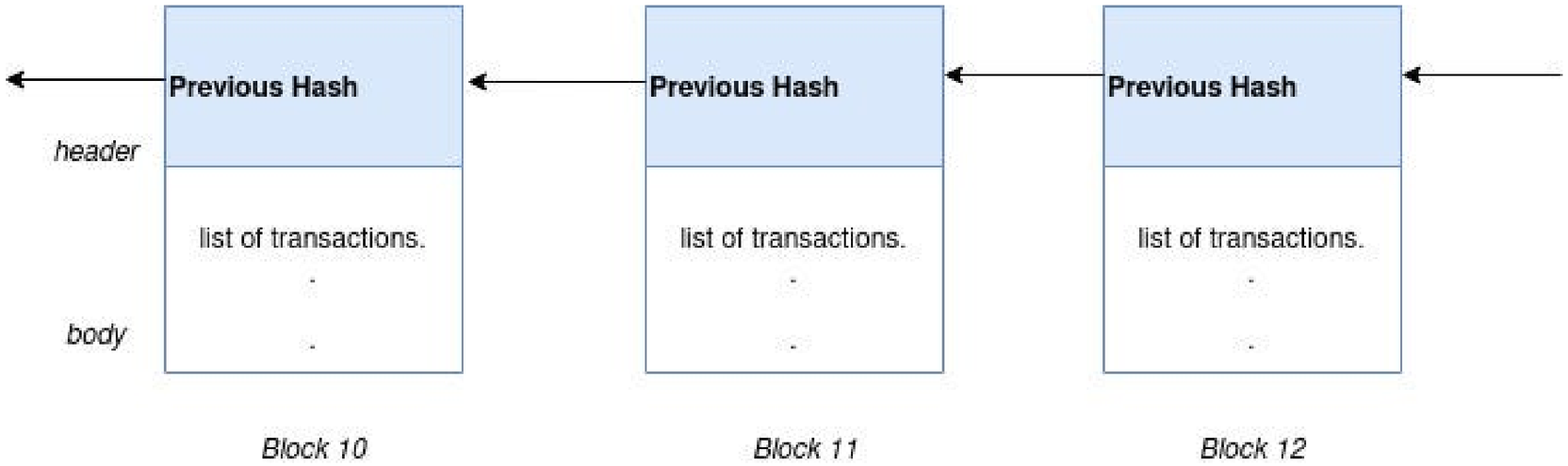
- **Folhas:** Cada folha armazena o hash de um bloco de dados.
- **Nós Internos:** Cada nó interno contém o hash resultante da concatenação dos hashes de seus dois filhos.
- **Merkle Root:** A raiz da árvore, ou Merkle Root, é um resumo único de todos os dados.

Para verificar a integridade, recalcula-se os hashes partindo das folhas até a raiz e compara-se com a Merkle Root original. Qualquer alteração em um dado modifica o hash correspondente e propaga a inconsistência até a raiz, permitindo a detecção rápida de adulterações.

Exemplo


Hash1234
Merkle root
Hash12
Hash34
Merkle branches
Hash1
Hash2
Hash3
Hash4
Merkle leaves
Data1
Data2
Data3
Data4
Data Nodes

Exemplo


Previous Hash
header
list of transactions.
.
body
.
Block 10
Previous Hash
list of transactions.
.
.
Block 11
Previous Hash
list of transactions.
.
.
Block 12

Código

Merkle Tree

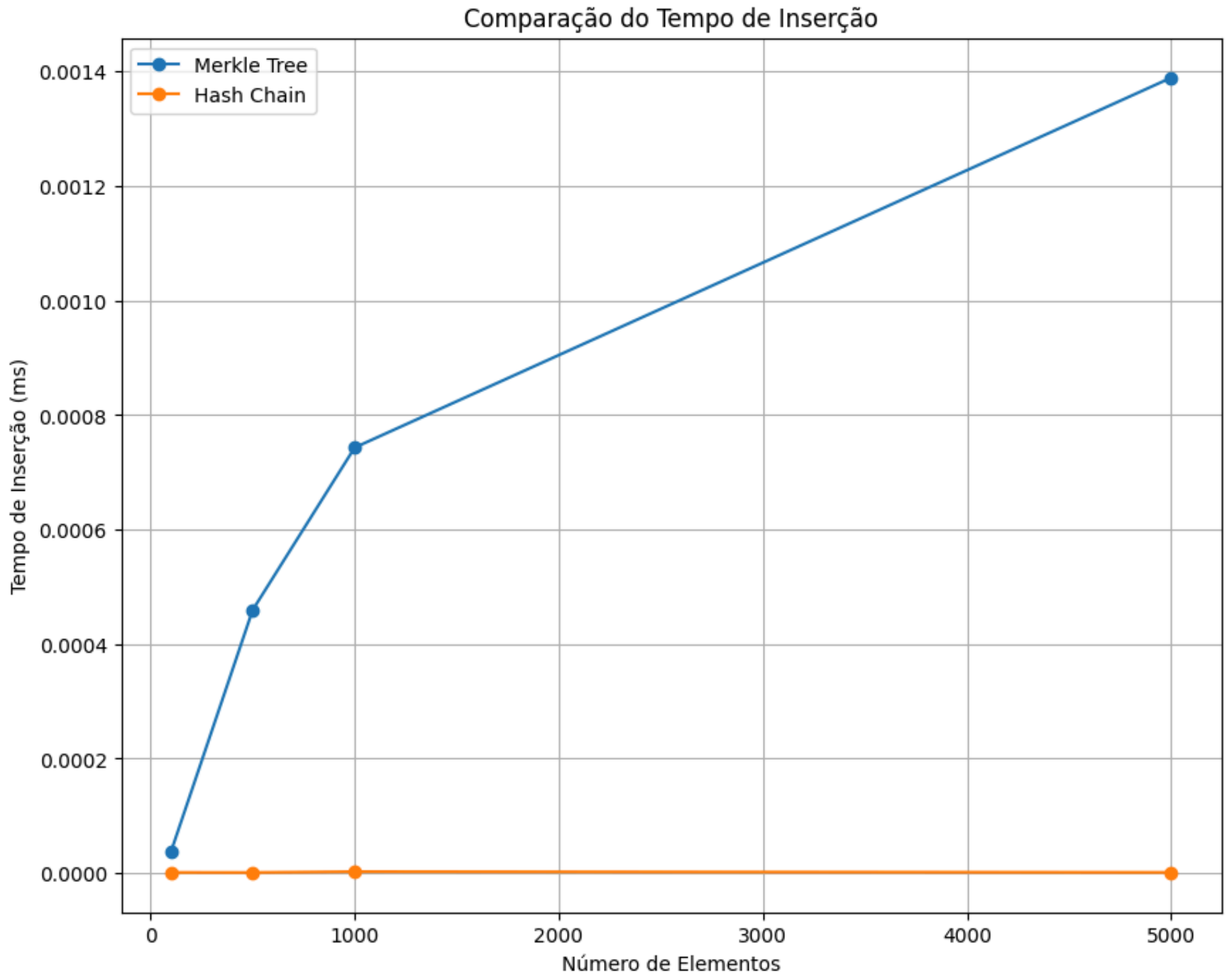
Cada inserção adiciona o hash do valor (calculado por $h(x)=x \bmod m$) à lista de folhas e reconstrói a árvore para atualizar a Merkle Root. A função de validação recalcula a raiz a partir das folhas e a compara com a raiz armazenada.

Hash Chain

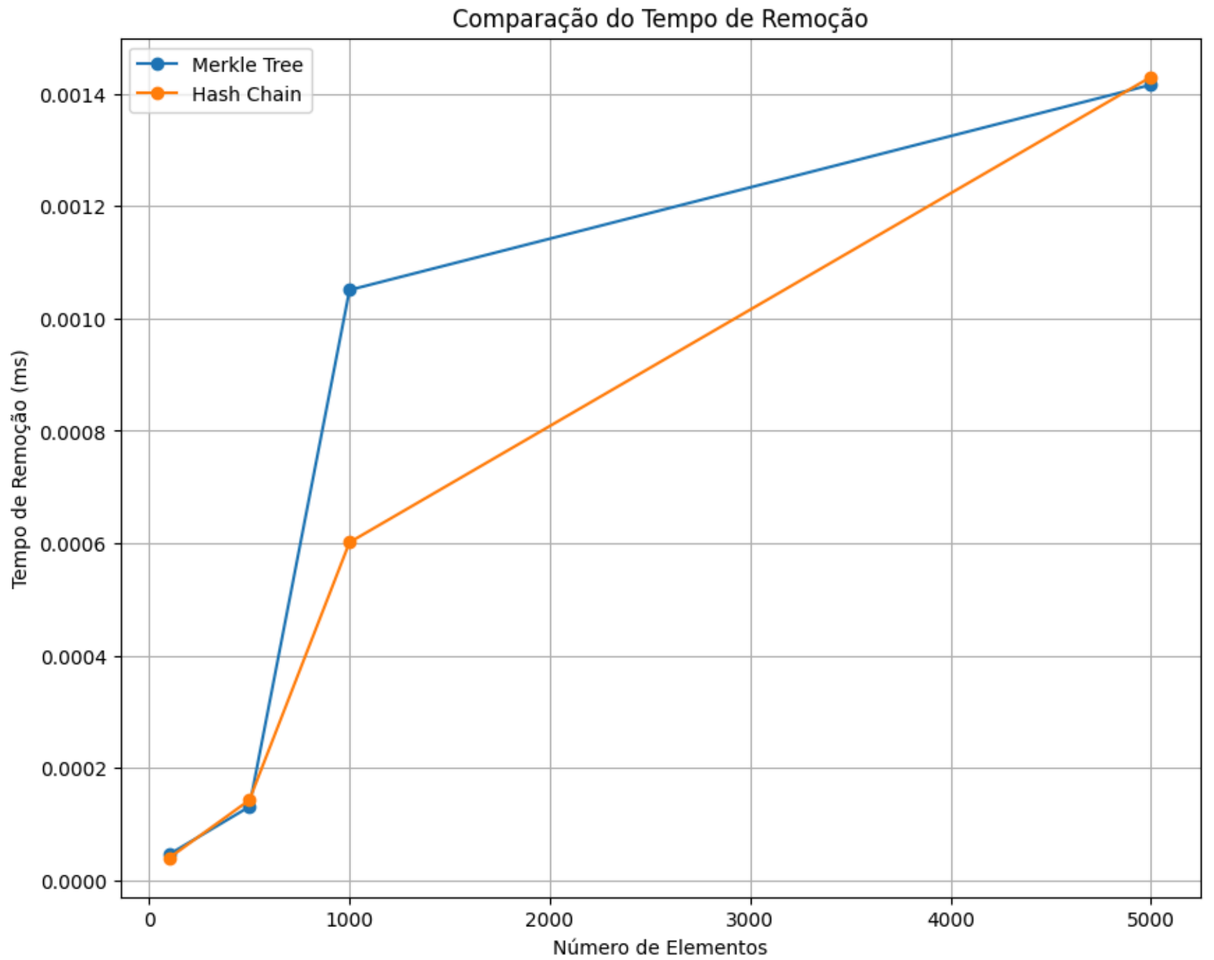
A hash chain armazena os dados originais e gera uma cadeia de hashes onde cada novo hash depende do hash anterior concatenado com o novo valor. A validação consiste em recalcular a cadeia completa e compará-la com a cadeia armazenada.

Link do Código para execução: Colab
Link do repositorio: Github

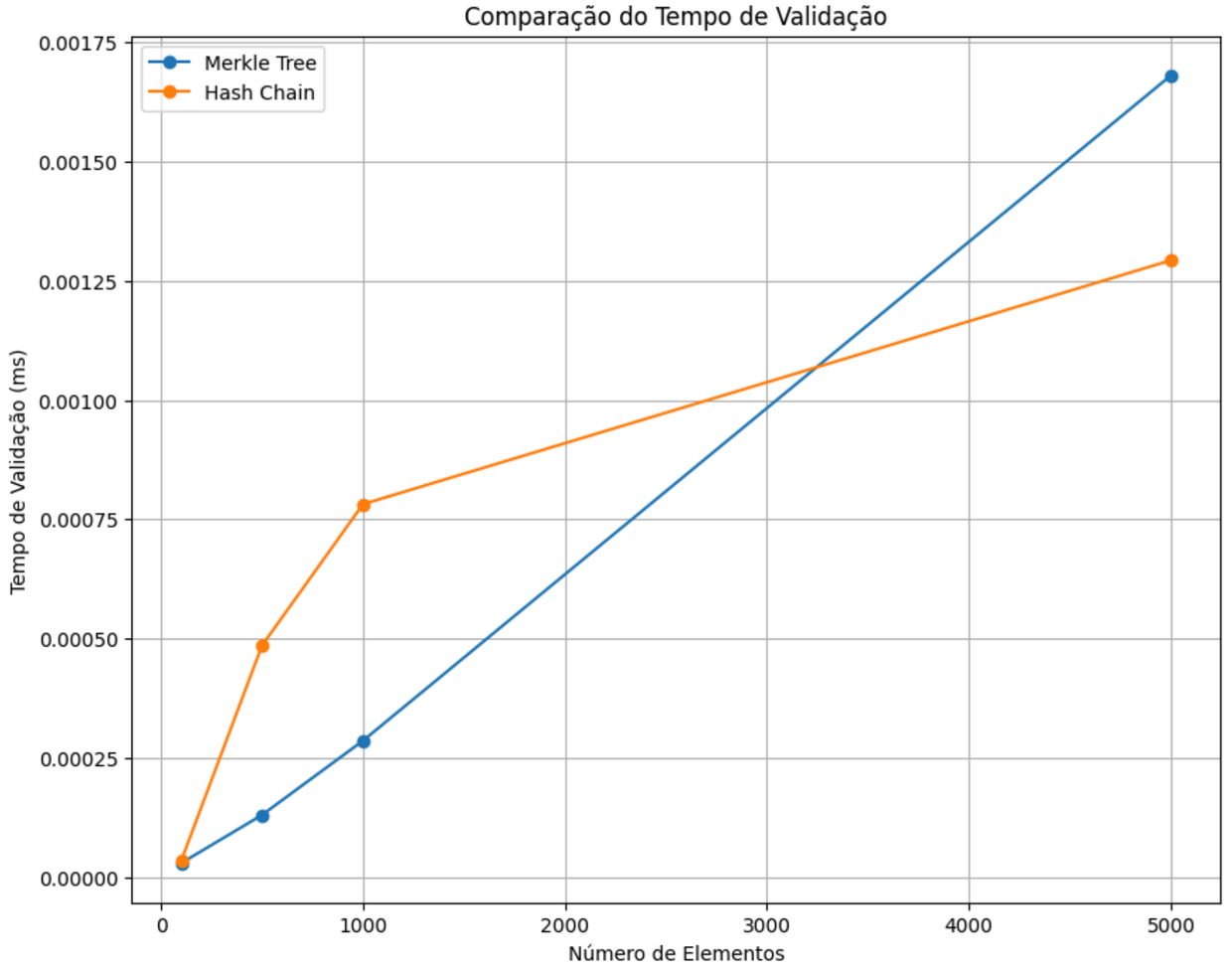
Resultados


Comparação do Tempo de Inserção
Merkle Tree
Hash Chain
Tempo de Inserção (ms)
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
Número de Elementos

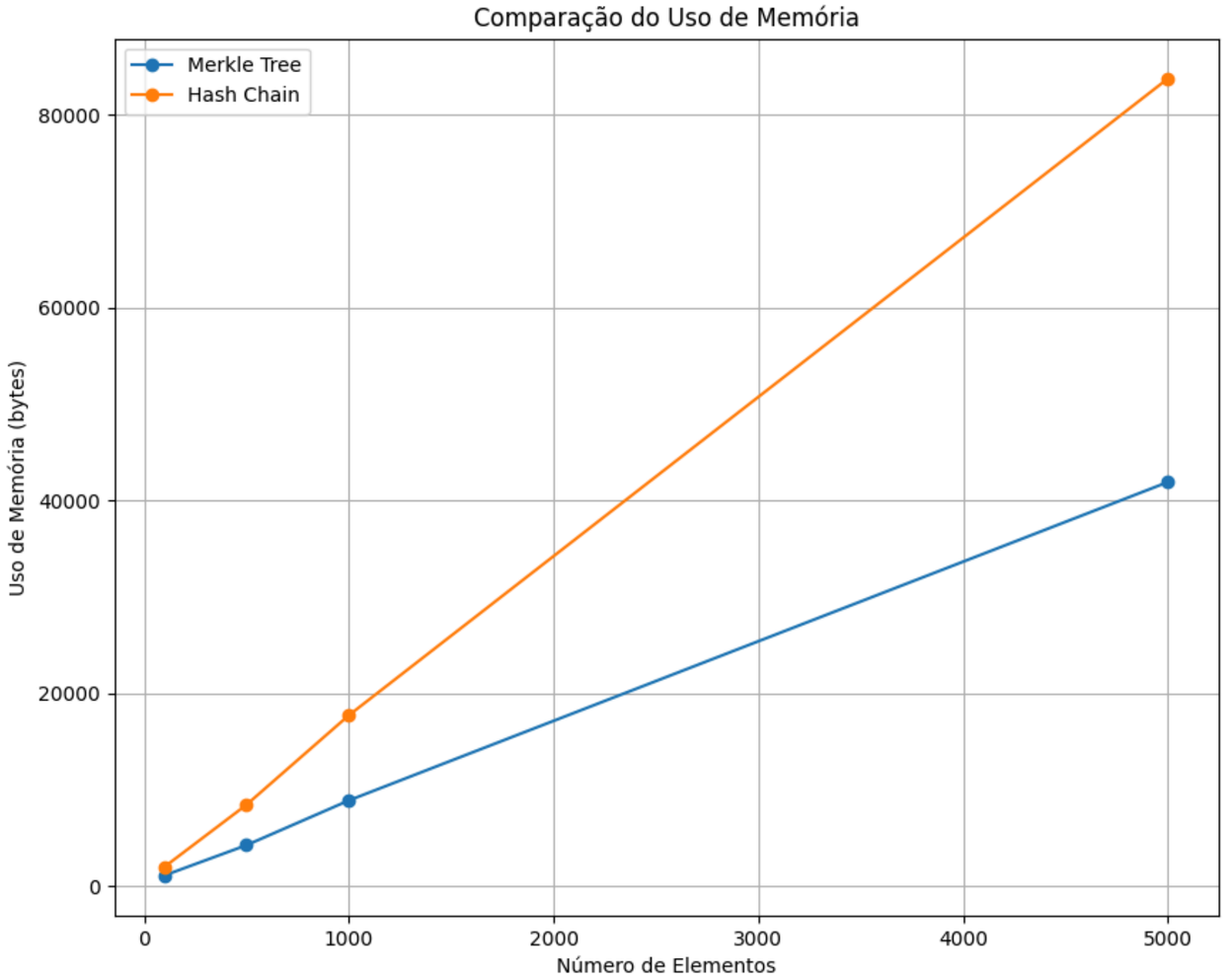
Resultados


Comparação do Tempo de Remoção
Merkle Tree
Hash Chain
Tempo de Remoção (ms)
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
5000
Número de Elementos

Resultados


Comparação do Tempo de Validação
Merkle Tree
Hash Chain
Tempo de Validação (ms)
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
0
1000
2000
3000
4000
5000
Número de Elementos

Resultados


Comparação do Uso de Memória
Merkle Tree
Hash Chain
Uso de Memória (bytes)
0
20000
40000
60000
80000
0
1000
2000
3000
4000
5000
Número de Elementos

Conclusão

As Merkle Trees são estruturas de dados que garantem a integridade e verificação eficiente de grandes volumes de informações, sendo amplamente utilizadas em sistemas distribuídos e blockchain. Comparadas às Hash Lists, que são mais simples e adequadas para sequências lineares de dados, as Merkle Trees oferecem maior escalabilidade e eficiência na verificação, tornando-se a escolha ideal para aplicações que lidam com grandes volumes de dados distribuídos.

Referências

- Carminati, B. (2009). Merkle Trees. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_1492
- Merkle Tree: "Merkle Tree." Brilliant Math & Science Wiki. Disponível em: https://brilliant.org/wiki/merkle-tree/. Acesso em: 14 fev. 2025.
- Horne, D. (2011). Hash Chain. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_780
- Verkle Trees: KUSZMAUL, William. "Verkle Trees." MIT Mathematics. Disponível em: https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf. Acesso em: 14 fev. 2025.

Referências

- Merkle Trees vs. Verkle Trees: Principais Diferenças Explicadas: "Merkle Trees vs. Verkle Trees: Key Differences Explained." CCN.com. Disponível em: https://www.ccn.com/education/crypto/merkle-vs-verkle-trees-key-differences-explained/. Acesso em: 14 fev. 2025.
- Como Funcionam as Merkle Trees?: "How Do Merkle Trees Work?" Baeldung on Computer Science. Disponível em: https://www.baeldung.com/cs/merkle-trees. Acesso em: 14 fev. 2025.
- Entendendo as Merkle Trees: Um Guia Prático para Hashing: "Understanding Merkle Trees: A Practical Guide to Hashing." Daisie Blog. Disponível em: https://blog.daisie.com/understanding-merkle-trees-a-practical-guide-to-hashing/. Acesso em: 14 fev. 2025.

Agradecimentos

computação
UFBA

VIRTURE SPIRITUS