Implementação e Análise de Árvores de Merkle

Gustavo Marques Pina m15838

November 2, 2025

1 Introdução

Neste relatório é apresentada a implementação de uma árvore de Merkle em Java, incluindo funcionalidades de escolha dinâmica do algoritmo de hash, a geração de transações aleatórias, impressão estruturada da árvore, criação de provas de Merkle e validação dessas provas. A análise foca-se nos resultados de desempenho obtidos na alínea f e nas possíveis otimizações para aumentar a velocidade do código.

2 Resultados da comparação das validações

Foram medidos os tempos médios de validação de provas e comparação com a abordagem *brute force* para um conjunto de 2^{10} transações (1024 transações). Os resultados foram:

- Validar uma prova **VÁLIDA**: 4.38 μs/op (4384.90 ns)
- Validar uma prova **FALSA**: 3.88 μs/op (3883.65 ns)
- *Brute force* : 0.95 ms/op (945115.00 ns)

Quantas vezes mais rápido foi em relação ao *brute force*?:

- Válida: 215.5x
- Falsa: 243.4x

3 Análise dos Resultados

Os resultados confirmam a vantagem teórica da validação com prova, que tem uma complexidade de apenas $O(\log n)$, já a reconstrução completa da árvore tem uma complexidade de $O(n)$. Para 1024 transações, a validação é centenas de vezes mais rápida. O ganho aumenta exponencialmente com o tamanho do conjunto.

4 Possíveis Otimizações

Para conseguirmos aumentar a velocidade do código, poderiamos fazer as seguintes melhorias :

1. **Uso de arrays de bytes em vez de Strings**: Evita concatenações com elevado custo computacional e reduz a pressão no *garbage collector* do Java.

2. **Paralelização da construção da árvore**: Utilizar o *ForkJoinPool* ou threads paralelos para processar níveis inferiores.

5 Conclusão

Esta implementação serve para demonstrar a eficiência das árvores de Merkle para verificação de integridade em grandes conjuntos de dados. As otimizações propostas podem melhorariam ainda mais o desempenho.