Implementação e Análise de Estruturas de Dados e Algoritmos de Criptografia, Compressão e Busca em uma Base de Dados de Animes

Antonio Neto, Henrique Lara

[1]Instituto de Computação – PUC Minas
Belo Horizonte, MG, Brasil

antonio.couto@sga.pucminas.br, henrique.lara@sga.pucminas.br

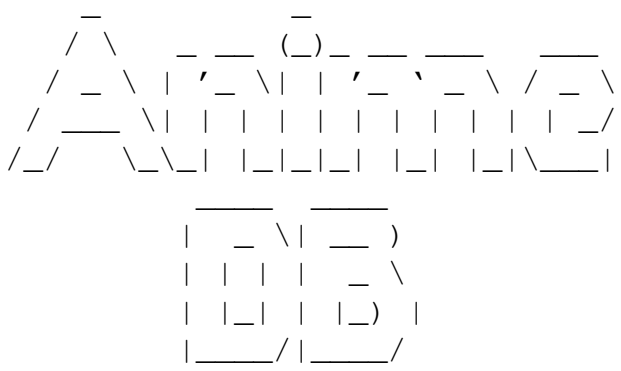
Abstract. *This paper presents the implementation and analysis of various data structures and algorithms, such as B+ Tree, Extended Hash, and Inverted Lists, RSA encryption, data compression using Huffman and LZW, and pattern searching using Boyer Moore.*

Resumo. *Este artigo apresenta a implementação e análise de diversas estruturas de dados e algoritmos, incluindo Árvore B+, Hash Estendido e Listas Invertidas, criptografia RSA, compressão de dados utilizando Huffman e LZW, e busca por padrões usando Boyer Moore.*

Index

1. Introdução

Este trabalho teve como objetivo implementar e analisar diferentes estruturas de dados e algoritmos em um sistema de registro de animes. As principais áreas de foco foram criptografia, compressão de dados e recuperação de informações. Segue o ”front-end” e as funcionalidades do aplicativo:



```
1. Carregar base de dados original (ALL)
2. Criar novo registro (CREATE ALL)
3. Busca por ID (READ SEQUENCIAL)
4. Deletar um registro por ID (DELETE ALL)
5. Update em um registro por ID (UPDATE ALL)
6. Pesquisa pelo index (HASH ESTENDIDO)
7. Pesquisa pelo index (ARVORE B+)
8. Pesquisa por titulo (LISTA INV)
9. Pesquisa por genero (LISTA INV)
10. Imprimir (HASH ESTENDIDO)
11. Imprimir (ARVORE B+)
12. Compactar (HUFFMAN)
13. Descompactar (HUFFMAN)
14. Compactar (LZW)
15. Descompactar (LZW)
16. Pesquisa por padrao (BM)
17. Criptografar (RSA)
18. Descriptografar (RSA)
19. Sair
>> 19
Obrigado e tchau, Hayala!
```

2. Desenvolvimento

2.1. Estruturas de Dados

Diferentes estruturas de dados foram implementadas para suportar operações eficientes de armazenamento e recuperação de informações na base de dados de animes.

2.1.1. Árvore B+

A Árvore B+ é uma estrutura de dados balanceada que permite busca, inserção e deleção eficientes. É especialmente útil para bases de dados e sistemas de arquivos.

2.1.2. Listas Invertidas

Listas invertidas foram utilizadas para indexar dados textuais, permitindo buscas rápidas por palavras-chave. Cada termo é associado a uma lista de documentos onde ele ocorre.

2.1.3. Hash Estendido

O Hash Estendido é uma técnica de hashing que utiliza buckets, diretórios e páginas para distribuir os dados de forma eficiente. Esta estrutura permite operações de inserção, busca e deleção rápidas, sendo ideal para bases de dados que exigem alta performance.

2.1.4. Justificativa da Escolha

- **Árvore B+**: A Árvore B+ foi escolhida devido à sua eficiência em operações de leitura e escrita, o que é crucial para grandes bases de dados como a de animes.
- **Listas Invertidas**: Optamos pelas listas invertidas para melhorar a velocidade das buscas textuais, uma vez que elas permitem uma recuperação rápida e eficiente dos registros associados a um termo específico.
- **Hash Estendido**: Escolhemos o Hash Estendido por sua capacidade de distribuir dados de forma eficiente e permitir operações rápidas de inserção e busca.

2.2. Compressão de Dados

Foram implementados dois algoritmos de compressão: Huffman e LZW. Ambos os métodos visam reduzir o tamanho dos dados para armazenamento eficiente na base de dados de animes. Como nossa base de dados tem caracteres japoneses, usamos dicionários com escopo Unicode, o que impactou negativamente a eficiência do LZW e, posteriormente, do RSA (criptografia).

2.2.1. Algoritmo de Huffman

O algoritmo de Huffman é um método de compressão que utiliza uma árvore binária para codificar os dados de forma eficiente. Cada caractere é representado por uma sequência de bits de comprimento variável, com caracteres mais frequentes recebendo códigos mais curtos.

2.2.2. Algoritmo LZW

O LZW (Lempel-Ziv-Welch) é um algoritmo de compressão que substitui substrings repetidas por códigos de tamanho fixo, construindo um dicionário dinâmico durante a

compressão e a descompressão.

2.2.3. Justificativa da Escolha

- **Huffman**: Escolhemos o algoritmo de Huffman devido à sua eficiência em comprimir dados com distribuição de frequência desigual, o que é comum em textos e dados de caracteres.
- **LZW**: O algoritmo LZW foi escolhido por sua capacidade de lidar eficientemente com padrões repetitivos em dados, o que é benéfico para grandes conjuntos de dados de texto como uma base de dados de animes.

3. Busca por Padrões

Para otimizar a recuperação de informações, foi implementado o algoritmo de Boyer-Moore para busca por padrões na base de dados de animes.

3.1. Boyer-Moore

O algoritmo de Boyer-Moore foi utilizado para busca por padrões na base de dados de animes. Usamos tanto a busca por caráter ruim quanto por sufixo bom, analisando qual era a melhor movimentação para cada caso e percorrendo os registros de forma eficiente.

3.2. Justificativa da Escolha

O algoritmo de Boyer-Moore foi escolhido devido à sua eficiência em buscas textuais, especialmente quando se lida com grandes textos ou bases de dados. Ele é conhecido por seu bom desempenho em cenários práticos.

4. Criptografia RSA

A criptografia RSA foi utilizada para criptografar um dos arquivos de registros. A implementação envolveu a escolha de números primos grandes, o cálculo das chaves pública e privada, e a implementação das funções de criptografia e descriptografia.

4.1. Implementação

A implementação do RSA incluiu a geração de chaves, criptografia de arquivos e descriptografia de arquivos. Usamos uma classe chamada BigInteger para conseguir executar o algoritmo mesmo usando caracteres do unicode, que são bem mais custosos para calcular. A seguir estão os passos principais:

1. Escolha de dois números primos grandes p e q.
2. Cálculo de $n = p \times q$ e $\phi(n) = (p-1) \times (q-1)$.
3. Escolha de um expoente público e tal que $1 < e < \phi(n)$ e e seja coprimo com $\phi(n)$.
4. Cálculo da chave privada d tal que $d \equiv e^{-1} \mod \phi(n)$.
5. Implementação das funções de criptografia $C = M^e \mod n$ e descriptografia $M = C^d \mod n$.

4.2. Justificativa da Escolha

Escolhemos o algoritmo RSA devido à sua ampla aceitação e segurança comprovada para criptografia de dados. Apesar de estar em risco perante à ascensão dos computadores quânticos, ele é adequado para o nosso programa, pois é um padrão difícil de ser quebrado usando força bruta.

5. Testes e Resultados

O aplicativo foi testado tanto em um container Docker quanto em computadores. As estruturas de dados todas funcionaram completamente, com testes usando todas operações do CRUD e ainda permitindo visualizar a estrutura de dados. A criptografia e descriptografia foram exatas, e o algoritmo de Boyer-Moore mostrou-se eficiente.

Os resultados da compressão utilizando os algoritmos de Huffman e LZW são apresentados a seguir:

5.1. Compressão Huffman

- Tempo de compressão: 385 ms
- Número de bytes comprimidos: 525917 bytes
- Porcentagem de compressão: 37.77%

5.2. Compressão LZW

- Tempo de compressão: 27425 ms
- Número de bytes comprimidos: 643852 bytes
- Porcentagem de compressão: 46.24%

6. Conclusão

Este trabalho apresentou a implementação de diversos algoritmos e estruturas de dados, demonstrando sua eficácia em um sistema de registro de animes. A criptografia RSA garantiu a segurança dos dados, enquanto os algoritmos de compressão, de busca e as estruturas de dados otimizadas permitiram armazenamento e recuperação eficientes. O código fonte pode ser encontrado em: `https://github.com/AnimeAEDS3/aeds3`.