Resumo

Este projeto apresenta a implementação de soluções para problemas clássicos de manipulação de dados e compressão utilizando a linguagem Java. Ele inclui algoritmos de ordenação, compressão, indexação invertida, e estruturas de dados como árvores B e tabelas hash. Além disso, aborda criptografia e decriptografia utilizando as técnicas de Vigenère e DES, bem como operações otimizadas de entrada e saída de dados.

Introdução

A manipulação eficiente de grandes volumes de dados é um desafio essencial em sistemas computacionais modernos. Este projeto foi desenvolvido para explorar soluções baseadas em algoritmos clássicos e estruturas de dados eficientes, com foco em:

- Ordenação eficiente de dados usando o algoritmo Balanced Merge Sort.
- Compressão de dados com os algoritmos de Huffman e LZW.
- Indexação e busca utilizando árvores B e tabelas hash.
- Manipulação de fluxos de bits para entrada e saída otimizadas.
- Criptografia e decriptografia de dados utilizando os métodos de Vigenère e DES.

O projeto processa dados estruturados a partir de arquivos CSV e gera saídas em formatos binários e textuais para posterior análise.

Estrutura do Projeto

O projeto está dividido em módulos independentes, cada um abordando uma funcionalidade específica:

1. **Algoritmos de Ordenação e Compressão**:
    a. `BalancedMergeSort.java`: Implementação de um algoritmo eficiente para ordenar grandes volumes de dados.
    b. `Huffman.java` e `LZW.java`: Algoritmos de compressão para reduzir o tamanho de arquivos com diferentes abordagens.
2. **Estruturas de Dados**:
    a. `BTree.java` e `BTreeStar.java`: Estruturas de árvores B para busca eficiente em grandes volumes de dados.
    b. `Hash.java`: Implementação de uma tabela hash com boa distribuição e baixa taxa de colisões.

3. **Indexação Invertida**:
    a. `FullInvertedIndex.java` e `InvertedIndex.java`: Criam índices para facilitar buscas textuais em grandes conjuntos de dados.
4. **Manipulação de Arquivos**:
    a. `BitInputStream.java` e `BitOutputStream.java`: Classes para leitura e escrita otimizadas de fluxos de bits.
    b. `EraseFiles.java`: Utilitário para exclusão de arquivos temporários ou obsoletos.
5. **Criptografia e Decriptografia**:
    a. `Vigenere.java`: Implementação da cifra de Vigenère, que utiliza uma chave para codificar e decodificar mensagens.
    b. `DES.java`: Implementação do algoritmo DES (Data Encryption Standard), garantindo segurança para dados sensíveis.

Entrada e Saída de Dados

- **Entrada**: O arquivo `Heroes.csv` fornece dados estruturados para processamento.
- **Saída**: Os arquivos `raf.bin` (binário) e `invertedIndex.csv` (texto) armazenam os resultados das operações realizadas.

Testes e Resultados

Os testes utilizaram o arquivo `Heroes.csv`, que contém dados relacionados a heróis fictícios. Os principais resultados foram:

1. **Ordenação**:
    a. O algoritmo Balanced Merge Sort demonstrou alta escalabilidade, ordenando eficientemente grandes volumes de dados.
2. **Compressão**:
    a. O algoritmo de Huffman reduziu o tamanho do arquivo de entrada em cerca de 40%.
    b. O algoritmo LZW apresentou eficiência comparável, com taxas de compressão dependentes da redundância dos dados.
3. **Indexação**:
    a. Os índices invertidos criados facilitaram buscas textuais rápidas e precisas.
4. **Estruturas de Dados**:
    a. As árvores B se mostraram eficazes para buscas e inserções em grandes volumes de dados.

   b. A tabela hash apresentou boa distribuição e baixa taxa de colisões, garantindo alta eficiência.
5. **Criptografia e Decriptografia**:
   a. A cifra de Vigenère foi eficaz para mensagens curtas, mas vulnerável a ataques de força bruta quando utilizada sem precauções adicionais.
   b. O algoritmo DES garantiu alta segurança para dados sensíveis, embora exija maior custo computacional em comparação a métodos modernos.

Conclusão

O projeto demonstrou a aplicação prática de algoritmos e estruturas de dados para resolver problemas de manipulação de grandes volumes de informações. As soluções implementadas foram eficazes e escaláveis, destacando-se na compressão de dados, organização, indexação eficiente e segurança da informação. Os resultados validam a robustez das técnicas empregadas, mostrando seu potencial para aplicações reais.