RELATÓRIO DO PROJETO 1

ESTRUTURA DE DADOS II
COMPRESSÃO DE ARQUIVOS COM O ALGORITMO DE HUFFMAN

Disciplina: Estrutura de Dados II
Professor: Dr. Jean M. Laine

Integrantes do Grupo:
Marcelo Luis Simone Lucas - 10332213
Rayane Yumi da Silva Tahara - 10410892
Jully Manuele Dias Lima - 10420556
Guilherme Garcia Marreto - 10403501

1. INTRODUÇÃO

O presente relatório documenta a implementação do algoritmo de compressão de Huffman, um método de codificação sem perdas que utiliza códigos de tamanho variável baseados na frequência dos caracteres. O projeto envolveu a aplicação de estruturas de dados fundamentais como Min-Heap e Árvore Binária para construir um compressor funcional em Java, capaz de codificar e decodificar arquivos.

2. ANÁLISES REQUERIDAS

2.1 Análise de Performance (Tempo de Execução)

Para avaliar a performance do algoritmo, foram medidos os tempos de compressão e descompressão de arquivos com tamanhos variados. Os resultados obtidos estão consolidados na tabela abaixo.

Tamanho Original (Aprox.)	Tempo de Compressão	Tempo de Descompressão	Tamanho Comprimido	Resultado
1 KB	102 ms	58 ms	1562 bytes	**Expansão**
100 KB	841 ms	458 ms	53012 bytes	Compressão
~1 MB	4481 ms	1715 ms	1048611 bytes	**Expansão**
~10 MB	41923 ms	8984 ms	5319242 bytes	Compressão

Análise dos Resultados:

A análise dos dados revela nuances importantes sobre a performance do algoritmo. Conforme o tamanho do arquivo aumenta, o tempo de execução para compressão e descompressão cresce de forma consistente, aproximando-se de uma relação linear, o que está de acordo com a complexidade teórica esperada do algoritmo.

Um dos resultados mais significativos foi a observação de que, para arquivos pequenos (como o de 1 KB), o tamanho do arquivo comprimido resultou em **expansão** (1562 bytes contra 1044 bytes originais). Este fenômeno ocorre devido ao custo fixo do cabeçalho, que armazena a tabela de frequências (1024 bytes). Para arquivos pequenos, o tamanho deste cabeçalho é maior do que a economia de espaço obtida com a codificação, tornando a compressão ineficaz.

Adicionalmente, os dados confirmam que o processo de **descompressão é consistentemente mais rápido que o de compressão**. Isso era esperado, pois a descompressão consiste em um percurso direto na árvore de Huffman (uma operação rápida), enquanto a compressão envolve tarefas mais custosas, como a análise de frequência de todo o arquivo e a construção da árvore e da tabela de códigos.

2.2 Análise da Taxa de Compressão por Conteúdo

A eficiência do algoritmo de Huffman depende diretamente da redundância dos dados. Para investigar essa relação, foram testados quatro tipos de arquivo com conteúdos distintos.

Tipo de Arquivo	Tempo de Execução	Taxa de Compressão
Repetitivo	50 ms	83.03%
Texto Comum	208 ms	43.04%
Código-Fonte	333 ms	47.07%
Aleatório	76 ms	28.06%

Análise dos Resultados:

A eficiência da compressão variou drasticamente conforme o tipo de conteúdo, confirmando a dependência do algoritmo da distribuição de frequências dos caracteres.

- **Arquivo Repetitivo:** Apresentou a melhor taxa de compressão. Este é o cenário ideal para Huffman, onde um alfabeto muito pequeno (poucos caracteres únicos) com altas frequências permite a criação de códigos binários extremamente curtos, maximizando a economia de espaço.
- **Texto Comum e Código-Fonte:** Ambos apresentaram taxas de compressão eficientes. Isso se deve à distribuição desigual de caracteres em linguagens naturais e de programação. Caracteres como 'espaço', 'a', 'e' (em texto) ou símbolos como { }, ( ), ; (em código) são muito frequentes e recebem códigos curtos, enquanto caracteres raros recebem códigos mais longos, gerando uma compressão eficaz.
- **Arquivo Aleatório:** Como esperado, este arquivo obteve a pior taxa de compressão entre os tipos de arquivo testados (28.06%). Embora a teoria preveja que um arquivo perfeitamente aleatório possa ter sua dimensão aumentada, o arquivo de teste gerado, mesmo com alta variedade de caracteres, ainda possuía redundância suficiente para que o algoritmo alcançasse uma compressão positiva. Isso demonstra a sensibilidade do método de Huffman, que ainda consegue explorar pequenas variações de frequência, mas sua eficiência é drasticamente reduzida em cenários de alta entropia, como previsto.

3. CONCLUSÃO

A implementação do algoritmo de Huffman foi uma experiência prática valiosa, consolidando conceitos de estruturas de dados como Min-Heaps e Árvores Binárias. O projeto demonstrou com sucesso a viabilidade da compressão baseada em frequência e evidenciou os cenários onde o algoritmo se destaca.

As análises confirmaram que a eficiência de Huffman está intrinsecamente ligada à redundância dos dados, sendo ideal para arquivos com padrões e distribuição de caracteres desigual, e ineficaz para dados com alta entropia, como arquivos aleatórios ou já comprimidos. O comportamento temporal do algoritmo mostrou-se consistente com a teoria, e a análise revelou aspectos práticos importantes, como o impacto do cabeçalho em arquivos pequenos. O projeto, portanto, cumpriu seus objetivos, resultando em uma ferramenta funcional e em um entendimento aprofundado de um dos algoritmos clássicos da ciência da computação.