UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO

LABORATÓRIO DE PROGRAMAÇÃO – CK0215 2023.1 T01A

TRABALHO 3: COMPACTAÇÃO VIA HUFFMAN (VERSÃO 1: 2023-05-26)

PROFESSOR: PABLO MAYCKON SILVA FARIAS

1 Resumo

Você deverá implementar em C++ um compactador e descompactador baseado na codificação de Huffman. O compactador deve gerar arquivos compactados de tamanho compatível com a codificação de Huffman, e o descompactador deve produzir saídas idênticas aos arquivos originais fornecidos ao compactador. Além disso, ambos devem executar em tempo aceitável, conforme os exemplos apresentados mais à frente. A implementação será entregue ao professor através de tarefa no SIGAA, e depois será apresentada em horário agendado. Tanto a implementação submetida via SIGAA quanto a apresentação contam para a nota. O trabalho é individual e cada estudante deve produzir seu código por conta própria.

Atenção:

1. Evite gastar tempo excessivo na engenharia interna do seu programa. O objetivo central é entender os algoritmos envolvidos e saber implementá-los de maneira correta e eficiente. Procure começar garantindo que o seu programa implementa de forma correta e eficiente os algoritmos em questão, e que ele atende aos requisitos do trabalho. Uma vez que as funcionalidades essenciais estejam prontas, você poderá refinar o seu código e aplicar as técnicas de programação do seu interesse.
2. Colabore para o bom andamento da disciplina e do aprendizado de todos, não usando códigos que não tenham sido escritos por você nem compartilhando seu próprio código, nem mesmo em estágio inicial de desenvolvimento. Em caso de dificuldade sua ou de algum colega, fale com o professor.

2 Compilação do Programa

O seu programa deverá ser escrito em C++17 padrão e ser compilável, sem erros e preferivelmente também sem avisos, através de uma linha como

```
g++ -Wall -Wextra -std=c++17 -pedantic -o programa  main.cpp
```

Caso algo além disso seja necessário para compilar o programa (por exemplo, caso o seu programa possua mais de um arquivo que precise ser explicitamente fornecido como entrada para o compilador), por favor inclua um arquivo `explicacoes.txt` com as devidas explicações.

3 Execução do Programa

A entrada do programa será necessariamente informada através dos argumentos fornecidos na chamada do mesmo, a qual terá um dos seguintes formatos, relativos à compactação e à

descompactação, respectivamente:

```
./programa -c ARQUIVO_DE_ENTRADA ARQUIVO_DE_SAÍDA
./programa -d ARQUIVO_DE_ENTRADA ARQUIVO_DE_SAÍDA
```

O formato de chamada do programa deve ser **exatamente esse acima**, pois o seu programa será submetido a uma sequência automatizada de testes que pressupõem os formatos acima para executar o programa.

Este trabalho não faz exigências sobre o comportamento do programa nos casos em que a chamada não estiver no formato esperado; em particular, portanto, você pode, mas não precisa, tratar erros no formato de chamada do programa.

4 Instâncias de Teste

O seu programa deve ser capaz de, em princípio, compactar e descompactar arquivos quaisquer,[1] mas este trabalho possui um conjunto padrão de instâncias de teste, que estarão disponíveis no SIGAA e servirão para fornecer parâmetros de tamanho de arquivo e tempo de execução. Seguem abaixo resultados para o programa escrito pelo professor neste semestre:[2]

Instância	Compressão	Descompressão	Tamanho Original	Tamanho Compactado
1.txt	0,002s	0,000s	6 bytes	31 bytes
2.txt	0,001s	0,001s	100 bytes	10 bytes
3.txt	0,001s	0,000s	0 bytes	0 bytes
4.txt	0,628s	0,426s	6,2M	3,6M
5.pdf	0,027s	0,020s	172K	172K
6.bmp	3,687s	2,746s	27M	26M
7.txt	0,090s	0,076s	980K	584K
8.txt	0,062s	0,027s	1,0M	260K

5 Critérios de Avaliação

A corretude do programa é critério básico de avaliação: o programa deve compactar e descompactar arquivos sem incorrer em erros de execução, e o resultado da descompactação deve ser igual ao arquivo inicialmente submetido à compactação.

O tempo de execução do programa também será considerado: tempos de execução tão longos que dificultem a realização de testes gerarão perda de pontos na avaliação. Observe que o professor também pode testar o seu programa para instâncias diferentes daquelas listadas acima.

O professor também avaliará o código-fonte à luz do conteúdo ensinado na disciplina. Trechos de código excessivamente ineficientes poderão levar à perda de pontos.

Por fim, a realização do trabalho de forma individual e de acordo com as regras da disciplina são elementos básicos para a nota.

[1] Exceto, claro, por restrições de transbordo de tipos como `int`, mas, numa arquitetura de 32 bits ou mais, um `int` é capaz de contar os bytes de arquivos com mais de 1 gigabyte.

[2] Os resultados são para uma compilação sem opções explícitas de otimização, conforme ilustrado pelo comando de compilação mencionado anteriormente. O seu programa não precisa gerar exatamente os mesmos resultados, até porque o tempo exato obviamente varia com a máquina, o compilador e as opções de compilação, mas de qualquer forma os números acima servem de parâmetro: se o seu programa executar num tempo muitas vezes maior, então você deve analisar se há algo implementado de maneira particularmente ineficiente. O mesmo vale para os tamanhos dos arquivos compactados.

6 Submissão do Trabalho

A solução do trabalho deverá consistir num arquivo `[Matrícula].zip` (sem colchetes) entregue através do SIGAA em tarefa a ser cadastrada pelo professor. Naturalmente, apenas o código-fonte deve ser submetido, sem qualquer arquivo executável. Prezando pelo tamanho do arquivo `zip` final, **também não devem ser incluídas instâncias para o programa**. Quaisquer explicações adicionais podem ser incluídas através de um arquivo `explicacoes.txt`.

Em caso de dúvida, contate o professor rapidamente.

– Que você tenha uma prática rica em aprendizados. Bom trabalho! –