

Decodificador Huffman PPM

Considerações Iniciais

Primeiramente, quero dizer que o codificador funciona. Entretanto, o decodificador possui um bug no sistema que implementa a exclusão, que impede o funcionamento do decodificador.

É possível atestar o funcionamento do codificador por meio de exemplos pequenos que pode ser feito a mão e comparados com a saída do codificador.

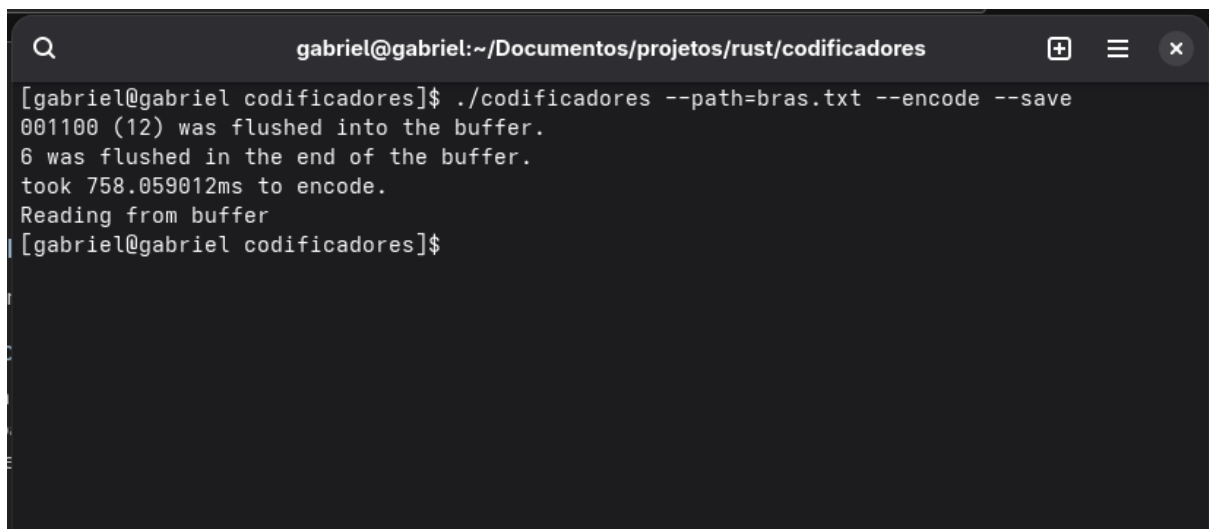
Introdução

Foi implementado um codificador de Huffman adaptativo usando ppm.

Uso

Codificar

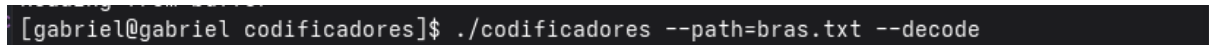
Para o codificador funciona basta passar a ele um caminho, para um arquivo, por meio do parâmetro “--path=” e passar o modo “--encode” ou “--decode”. Também é necessário passar o parâmetro “--save”.

A terminal window with a dark background. The title bar shows the user 'gabriel' at host 'gabriel' in the directory '~/Documentos/projetos/rust/codificadores'. The terminal text shows the command './codificadores --path=bras.txt --encode --save' being executed. The output includes: '001100 (12) was flushed into the buffer.', '6 was flushed in the end of the buffer.', 'took 758.059012ms to encode.', and 'Reading from buffer'. The prompt '[gabriel@gabriel codificadores]\$' is visible at the end of the line.

```
gabriel@gabriel:~/Documentos/projetos/rust/codificadores
[gabriel@gabriel codificadores]$ ./codificadores --path=bras.txt --encode --save
001100 (12) was flushed into the buffer.
6 was flushed in the end of the buffer.
took 758.059012ms to encode.
Reading from buffer
[gabriel@gabriel codificadores]$
```

Decodificar

Basta passar “--decode” ao em vez de “--encode”.

A terminal window showing the command './codificadores --path=bras.txt --decode' being executed. The prompt '[gabriel@gabriel codificadores]\$' is visible at the end of the line.

```
[gabriel@gabriel codificadores]$ ./codificadores --path=bras.txt --decode
```

Implementação

Codificador

O primeiro passo para implementar foi criar um buffer na memória do arquivo passado.

O codificador é um loop que passa por cada caractere da entrada. A cada interação do loop ele chama uma função chamada “*encoder*”, depois uma função chamada “*update_tables*” e por fim uma chamada “*update_context*”.

- Encoder é responsável por codificar o símbolo e salvar na saída desejada.
- Update_tables é responsável por atualizar a tabela de contextos com o caractere codificado.
- Update_context é responsável por atualizar a frase de contexto.

A frase de contexto é um buffer janela de tamanho K, que é utilizado para atualizar a tabela de contextos.

Encoder

O encoder é uma função recursiva encapsula 4 'ifs'.

1. Se a frase de contexto tiver tamanho menor que k, 'encoder' é chamado recursivamente diminuindo k.
2. Se a frase de contexto tiver tamanho maior que k 'encoder' é chamado recursivamente removendo o primeiro caractere da frase.
3. Se a frase tiver tamanho igual a k e k for igual a zero, a função irá verificar se o caractere de entrada existe na tabela de k = 0. Se existir, codifica o caractere usando essa tabela para criar uma arvore de Huffman. Se não existir, usa a tabela para codificar RO, e escreve o caractere bruto na saída.
4. Se a frase tiver tamanho igual a k, e k for diferente de zero a função irá procurar o caractere de entrada dentro de algum contexto. Se o programa encontrar o caractere de entrada dentro de uma tabela de um contexto, o programa irá usar esta tabela para criar a arvore de Huffman e codificar a entrada. Caso exista o contexto, mas esse contexto ainda não tem o caractere de entrada, ele irá usar a tabela desse contexto para codificar RO e vai passar para o próximo K usando uma função para aplicar a exclusão nos contextos seguintes até encontrar um contexto que possua o caractere e usá-lo para codificar, ou chegar na tabela de K = 0, onde ele vai executar o caso 3.

Exclusion Table

Exclusion table é a função responsável por criar uma tabela de contexto aplicando o princípio de exclusão. Ela funciona pegando o contexto do $k - 1$ e excluindo todos os elementos que também aparecem.

Decoficador

Ele recebe um vetor de bytes e o transforma em um vetor de bits, e usa esse vetor de bits para alimentar um buffer, que é usado em um loop para decodificar a mensagem.

Dentro loop é chamado a função 'decoder', 'update_tables', 'update_context', as duas últimas são as mesmas chamadas no codificador.

Decoder

É uma função recursiva que engloba 4 'ifs' nos mesmos moldes do decoder. A única função diferente é a responsável pela aplicação do princípio de exclusão, que é a 'decode_exclusion'.

Decode_Exclusion

Essa função deve possuir um bug de lógica que impossibilita o decoder de funcionar. Esse bug está no caso em que a frase de contexto é igual a k e k é diferente de 0.

Resultados e Conclusões finais

O codificador apresentou desempenho descente diminuindo o tamanho do arquivo de 367,3kB para 111,7kB numa média 758ms.

```
[gabriel@gabriel codificadores]$ ./codificadores --path=bras.txt --encode --save
001100 (12) was flushed into the buffer.
6 was flushed in the end of the buffer.
took 762.563639ms to encode.
Reading from buffer
[gabriel@gabriel codificadores]$ S
```

Entretanto, o decodificador para de executar em algum momento.

```
[gabriel@gabriel codificadores]$ ./codificadores --path=bras.txt --decode
m
e
m
o
r
i
a
s
p
o
s
t
u
m
a
s
d
e
b
r
a
s
c
u
b
a
s
m
a
c
h
a
d
o
t
u

thread 'main' panicked at src/huffman/utils.rs:40:67:
called `Result::unwrap()` on an `Err` value: 0
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
[gabriel@gabriel codificadores]$ S
```