Relatório do trabalho prático 1 da disciplina de Algoritmos 2 - Manipulação de sequências.

Aluno: Alexis Duarte Guimarães Mariz
Professor: Renato Vimieiro

Implementação de compressão e descompressão de arquivos

Fiz a minha implementação em Python. Implementei uma árvore Trie tradicional. Para isso, utilizei duas classes: No e Trie.
No meu main, fiz a lógica para determinar o nome do arquivo de saída, tanto para o caso de compactação, quanto para o caso de descompactação.

Para comprimir, a lógica foi toda implementada na classe Trie, utilizando o método 'comprimir'. Para isso, a Trie recebe como parâmetro o texto a ser compactado e inicializa com o nó 'raiz'(índice 0,caractere vazio). Quando o método 'comprimir' é chamado no main, ele procura o local de inserção do nó na trie e o insere. Além disso, o método 'comprimir' adiciona o índice do nó pai e do caractere no array 'tokens', representados por 4 bytes(código abaixo).

```
binario = np.zeros(1, dtype=np.uint32)
binario[0] |= (self.no_atual.index << 8)
binario[0] |= (caractere)
self.tokens = np.append(self.tokens,binario[0])
```

Cada token são 32bits(4 bytes), onde os 3 primeiros bytes representam o índice do pai e o 4o byte representa o caractere, da seguinte forma:

00000000 00000000 00000000 *00000000*
Sublinhado = índice do nó pai *Itálico = caractere*

Utilizei um np.array de "dtype = np.uint32" para garantir que cada token tivesse 32bits. Ao abrir o arquivo, utilizei o parâmetro "wb"(write binary) para salvar os tokens como binários de 4 bytes. Com isso, na hora de descomprimir basta ler de 4 em 4 bytes.

```
with open(nome_saida, "wb") as arquivo:
    for token in trie.tokens:
        arquivo.write(token)
```

No entanto, durante a minha implementação, descobri que estava escrevendo os bytes no arquivo comprimido em Little Endian. Com isso, na hora de descomprimir, quando acessamos token[0], obtemos o byte menos significativo(caractere). Utilizei a seguinte lógica para o obter o índice do nó pai.

```
index = (token[3] << 16) | (token[2] << 8) | (token[1])
```

A lógica da descompressão está implementada no main. Crio duas listas ‘indices’ e ‘caracteres’ e cada token ‘i’ é representado pelo par (indices[i],caracteres[i]), onde indices[i] representa o nó_pai do nó ‘i’. Primeiro eu leio o arquivo no modo “rb”(read binary) a cada 4 bytes. Em seguida, utilizando a lógica descrita acima, salvo o caractere e o índice do nó_pai do token.
Para escrever o arquivo descomprimido, utilizo um for(começando em 1, porque o 0 é sempre um token que representa a string vazia) para passar por todos os tokens. Para escrever primeiro a substring representada pelo nó_pai e depois o caractere, pensei em criar um método recursivo, mas optei por uma pilha devido à ineficiência da recursão. A pilha armazena a ordem a ser percorrida desde a raíz até o nó ‘i’. Com isso, escrevo os caracteres no arquivo de saída utilizando os tokens de acordo com a ordem definida pela pilha.

Casos de teste:

Primeiramente rodei 12 casos de teste, extraídos do Projeto Gutenberg. Listei-os abaixo, ordenados de acordo com o seu tamanho descomprimido. Observei que quanto maior o tamanho do arquivo, maior é a taxa de compressão.
No entanto, o arquivo “constituicao.1988.txt” teve uma taxa de compressão maior do que o esperado para um arquivo do seu tamanho. Imaginei que pudesse ser devido a uma maior quantidade de palavras e frases repetidas em “juridiquês”. Para tentar explicar isso, realizei dois novos casos de teste: “us_consitution”(constituição dos Estados Unidos) e “the-report-of-the-iraq-inquiry_executive-summary.txt”(um relatório da Inglaterra sobre a guerra no Iraque) e ambos tiveram uma taxa de compressão maior do que outros arquivos de tamanho similar.

1- “alice_in_wonderland.txt”(170.5kB)
Comprimido: 130.9kB
Taxa de compressão: 23%

2- “os_lusiadas.txt”(344.5kB)
Comprimido: 260.3kB
Taxa de compressão: 24%

3- “memorias_postumas.txt”(394.8kB)
Comprimido: 292.4kB
Taxa de compressão: 26%

4- “dom_casmurro.txt”(409.6kB)
Comprimido: 296.4kB
Taxa de compressão: 28%

5- “quincas_borba.txt”(482.5kB)
Comprimido: 343.5kB
Taxa de compressão: 29%

6- “constituicao.1988.txt”(651.8kB)
Comprimido: 356.2kB
Taxa de compressão: 45%

7- “dracula.txt”(865.8kB)
Comprimido: 566.6kB
Taxa de compressão: 35%

8- “moby_dick.txt”(1.3MB)
Comprimido: 818.5kB
Taxa de compressão: 37%

9- “twenty_years.txt”(1.4MB)
Comprimido: 848.7kB
Taxa de compressão: 39%

10- “middlemarch.txt”(1.8MB)
Comprimido: 1.1MB
Taxa de compressão: 39%

11- “anna_karenina.txt(2.0MB)
Comprimido: 1.2MB
Taxa de compressão: 40%

12- “tom_jones.txt”(2.0MB)
Comprimido: 1.2MB
Taxa de compressão: 40%

Extra: “us_consituition”(298.1kB)
Comprimido: 166.2kB
Taxa de compressão: 44%

Extra: “the-report-of-the-iraq-inquiry_executive-summary.txt”(388,1kB)
Comprimido: 257.6kB
Taxa de compressão: 34%