Trabalho Prático de Algoritmos 2

Utilizar o algoritmo LZ78 para comprimir arquivos

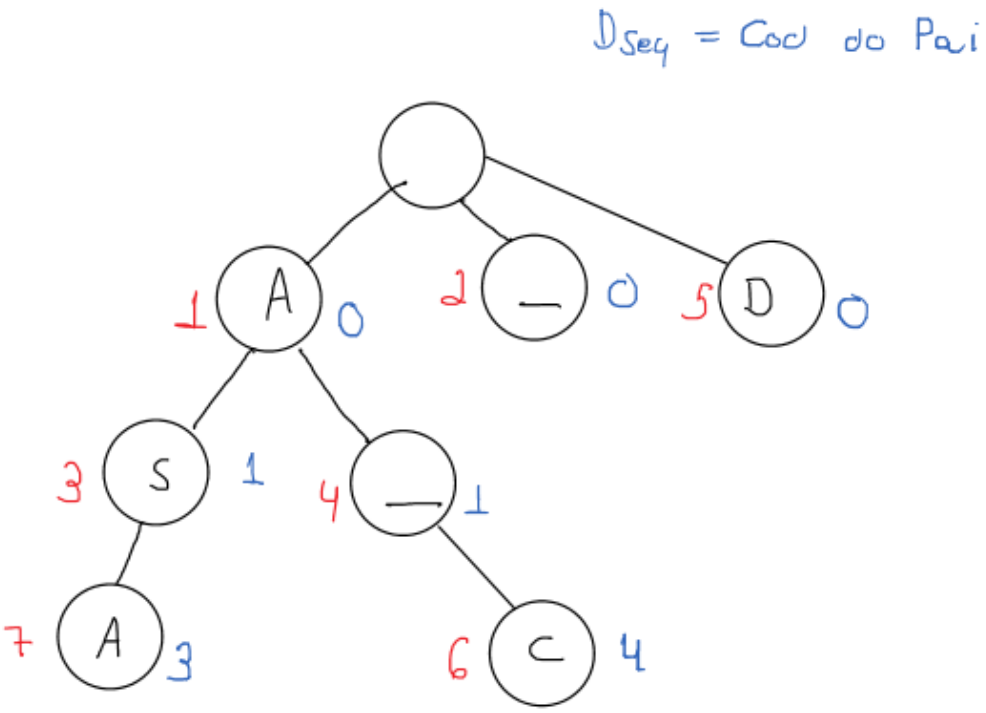
Nome: Marco Túlio Machado França

Introdução:

Nessa prática trabalhos o conceito de árvores de prefixos junto com o algoritmo LZ78 para comprimir arquivos. A ideia do algoritmo é codificar as strings que aparecem no texto e ao invés de salvar todos os caracteres em um arquivo, podemos salvar só partes realmente importantes para no futuro poder decodificar o arquivo.

Entendimento:

Para entender o algoritmo segui a referência da página da wikipedia sobre o LZ78 e fiz o seguinte desenho para tentar enteder como a string dada como exemplo, "A_ASA_DA_CASA", ficaria armazenada na trie:



Em azul temos o código do “pai” do nó, em vermelho temos o código do nó e de preto temo o “valor” do nó, que no meu algoritmo chamei de caractere.
Observe que é importante saber o código do nó pai. Por exemplo, quando o sétimo caractere "A" é adicionado, a saída é (3,'A'), exatamente para termos o mapeamento do pai desse caractere e poder trabalhar para reconstruir a string.

Desenvolvimento

Eu não tinha familiaridade com python, até o momento, praticamente tudo que fiz na faculdade foi em c++, mas por ser uma linguagem bem mais simples que c++, por mais que seja tudo novo, decidi fazer em python.

Primeiramente trabalhei com a parte de como receber as entradas da linha de comando, depois leitura e escrita em arquivos e por fim, de fato a implementação dos algoritmos.

Utilizando a ideia do algoritmo e a sugestão do professor, criei uma função para varrer o arquivo uma vez, *insercaoParaContarBytesInteiros*, e assim, inserir criar a trie e verificar quantos bytes seriam necessários para armazenar todos os inteiros. Então, por exemplo, se Dseq = 1000, que é maior 2^8 e menor que 2^16, seriam necessários 2 bytes para salvar os inteiros que representam os códigos dos nós.

Com essa ideia já deu para comprimir bem os arquivos. Então, depois de passar uma vez verificando, utilizei a função *inserirComprimido* que reseta a trie e insere novamente, mas agora escrevendo no arquivo binário.

Tive problemas ao converter caracteres que tinham tamanho em bytes maior que 1, apareceram só de 2 e 3 bytes, e a estratégia que utilizei para driblar essa situação foi, toda vez que encontrar um caractere que em bytes terá tamanho igual 2 vou inserir o caracter ~ (til) e se o tamanho for igual a 3, inserir ^ (circunflexo) no arquivo binário antes de inserir o caractere. Utilizei esses caracteres porque não é comum termos esses caracteres normalmente utilizado para acentuar palavras “soltos” no arquivo. E em todos os casos de testes realmente não tinha, então foi o suficiente para essa atividade. Uma forma de deixar mais robusto e garantir em qualquer situação, seria verificar na primeira varredura do arquivo os caracteres de 1 bytes que não existem nele e guardar os códigos e na segunda passada, na função *inserirComprimido*, utilizar algum deles. Assim, independente do arquivo daria certo. Mas, repetindo, no caso específico de todos os cenários de teste que utilizei a forma implementada foi suficiente. Para descomprimir o código, sempre que encontrar um dos dois caracteres, til ou circunflexo, saberemos quantos bytes teremos que ler.

O trabalho ficou com três arquivos .py, a main.py que tem a parte de tratar a entrada do terminal e executar o programa em si. Trie.py que tem a estrutura para compactar o arquivo e as funções que precisei implementar para tratar o texto. E, finalmente, o arquivo descomprimir.py, onde coloquei algumas funções que precisei utilizar para descomprimir o arquivo .z78.

Análise

Ao executar o programa para comprimir obtive uma taxa de compressão variando entre 23% e 53% como podemos ver na tabela abaixo

	tamanho .txt (KB)	tamanho .z78 (KB)	Taxa de Compressão
Ex01	95,5	61,2	35,92%
Ex02	102	66,3	35,00%
Ex03	43,9	27,9	36,45%
Ex04	190	91,3	51,95%
Ex05	222	116	47,75%
Ex06	147	82,6	43,81%
Ex07	169	98,8	41,54%
Ex08	1930	1447	25,03%
Ex09	1962	1465	25,33%
Ex10	1666	1282	23,05%

Conclusão:

O trabalho foi uma excelente oportunidade de concretizar os conhecimentos teóricos e de aprender novas tecnologias. Foi interessante entender como funciona a compressão de arquivos, não só por conta de usar um algoritmo que ajude a comprimir como é o caso do LZ78 mas também de ter que escrever os arquivos binários e ter que tratar os casos em que o caractere é escrito com mais de 1 byte.