Relatório: Trabalho Prático n.º 2

Bruno Almeida | Daniel Pereira | Tiago Conceição

1.

Usa-se o método readBits() da classe GZIP para ler a quantidade de bits necessária para o HLIT, HDIST e HCLEN, sendo 5, 5 e 4 respetivamente.

2.

Usa-se o método comp_codigo() usando HCLEN+4 como parâmetro que vai usado como o tamanho de um array onde vai sendo adicionados 3 bits lidos do ficheiro, sendo que esse array vai conter o comprimento dos códigos (no formato da sequência 16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15), sendo depois chamado o método code_order() que cria um novo com a sequência organizada e devolve esse array que também é o array devolvido pelo método comp_codigo().

3.

Usa-se o método Huff_code() que recebe o array do ponto anterior para criar os códigos da árvore de Huffman usando os elementos do array como os tamanhos dos códigos. O método primeiro cria um array (next_code) com os números onde se começam a escrever os códigos de Huffman, sendo que depois se criam os códigos da árvore de Huffman que são adicionados a outro array (tree_code) sendo esse array devolvido pelo método.

Usa-se o método Huff_tree() para criar a árvore de comprimentos de códigos Huffman usando os códigos de Huffman resultantes do método Huff_code().

4.

Usa-se o método comp() que recebe a árvore de comprimentos de códigos Huffman e HLIT+257 que vai ser usado como o tamanho do array final. O método vai lendo do ficheiro bit a bit até encontrar uma folha na árvore e depois adicionando esse valor ao array final ou ler bits extra se esse valor for 16, 17 ou 18, retornando o leitor da árvore para o topo da mesma sempre que é encontrado um valor. O método devolve o array com tamanho HLIT+257 que contém os comprimentos dos códigos de Huffman do alfabeto de literais

5.

Usa-se o mesmo método do ponto anterior, mas usando HDIST+1 como o tamanho do array final que vai conter os comprimentos dos códigos de Huffman do alfabeto de distâncias.

6.

Usam-se para os métodos Huff_code() e Huff_tree() para criar os códigos de Huffman e as respetivas árvores para o alfabeto dos literais e para o alfabeto das distâncias.

7.

Usa-se o método decode_data() para descompactar a informação do ficheiro, sendo que recebe as árvores dos literais e das distâncias como parâmetros. O método vai lendo o ficheiro bit a bit até encontrar uma folha de literais, depois, se o valor encontrado for inferior a 256 adiciona o literal ao array final (data) e retorna o leitor da árvore para o topo da mesma; se o valor for igual a 256, o ciclo termina e devolve data; se o valor for superior a 256, chama-se um método extra_lenght() com o valor obtido que vai determinar o número de bits extra que se tem de ler e a distância a que corresponde o valor obtido de acordo com a tabela dos literais e retorna o leitor da árvore para o topo da mesma, depois entra-se num ciclo que lê bit a bit no ficheiro até encontrar um valor na árvore das distâncias, chamando depois o método extra_distance() com o valor obtido da árvore das distâncias e retorna-se o leitor da árvore das distâncias para o topo da mesma. Após as leituras serem feitas copiam-se os símbolos correspondentes ao valor de lenght (valor devolvido por extra_lenght()) do array data recuando-se o valor de distance (valor devolvido por extra_distance())

O método extra_lenght() usa um dicionário pré-definido com o valor mínimo para o número de bits que é preciso ler nas chave e com os respetivos comprimentos nos itens, depois vai-se comparando o valor obtido com as chaves do dicionário e calcula-se a distância usando a fórmula: lenght=d[keys[x-1]]+(num-keys[x-1])*2**(x-1)+bits, sendo x a chave do dicionário (d) mais próxima, maior que o valor recebido pela função (num) e bits é o número obtido dos bits extra lidos, no entanto, se o valor recebido for igual a 285 define-se lenght=258. O método devolve lenght, que corresponde ao número de símbolos a ser lidos

O método extra_distance() usa o mesmo princípio do método extra_lenght() e devolve a distância que se tem de recuar para copiar no array de saída (data).

8.

Usa-se o método load_file() que recebe o array data do método decode_data() para criar o ficheiro descompactado com o mesmo nome que o ficheiro original (obtido através de self.gzh.fName). O método cria um ficheiro binário no modo de escrita e escreve cada elemento do array que recebe na forma de 1 byte, fechando o ficheiro quando acabar de escrever