**Função da tabela de ocorrência**array com no máximo 256 bytes de dados, que é escrito no header do arquivo. Cada byte é escrito no header seguido de sua frequência, no formato unsigned int (4 bytes).

o motivo de ser 256 é simples. Um arquivo é composto de bytes, cada byte possui 8 bits que são 0 ou 1. Calculando a combinação de todos os bits, você sabe que existem apenas **256 bytes** diferentes.

nosso alfabeto contém 26 letras, mas o “alfabeto de um arquivo” possui 256 bytes diferentes. por isso você tem que calcular a frequência dos 256 caracteres diferentes para codificar um arquivo.

quando um bloco é incrementado: o **tamanho** dele e a **frequencia de ocorrencias** também são incrementados a partir da contagem de bytes

chama a função de compressão

busca e adiciona as ocorrências na árvore

**Função de Compressão**-Função BitWriter()

O bitwriter cria um escritor que deve ser usado para escrever os dados a serem comprimidos num arquivo de saída.

Cada bit escrito no arquivo, é adicionado diretamente no buffer. Um header é criado para identificação posterior e agrupamento dos bits escritos no buffer.

Cada byte escrito, é lançado no header/buffer e sua frequência é lançada no contador.

O último byte pode ter zeros na direita, então um novo byte é escrito no final só pra dizer quantos bits são validos no último byte dos dados comprimidos.

Caso o buffer esteja cheio, ele será descarregado no arquivo.  
Força a escrita de todo o conteúdo do buffer para o arquivo.

**Função de Descompressão**-Função BitReader()

A árvore não é escrita no arquivo. O que é necessário é que você possa construir a árvore a partir dos dados que estão gravados no arquivo.   
A frequência dos caracteres é armazenada no header do arquivo. Ao ler essas frequências é possível reconstruir a arvore de Huffman e assim decodificar o restante do arquivo.

O bitreader cria um leitor que deve ser usado para ler o header (PRIMEIRAMENTE) juntamente dos dados comprimidos, que estão armazenados no buffer.

O ultimo byte dos dados deve ter zeros na direita que não são significantes. Então o leitor vai ler o ultimo byte do arquivo para indicar quantos bits válidos (significantes) tem no ultimo byte dos dados.

Na descompressão sempre verifica se ele chegou no último byte e garante que o código dele será lido corretamente.

A função é executada até o fim do arquivo ser alcançado.

No processo de leitura, a partir do header, a árvore de Huffman é reconstruída.

A procura (DFS) vai até o nó correspondente ao código armazenado, e o adiciona em um buffer, que vai armazenando até sua capacidade máxima.

Quando sua capacidade máxima é atingida, um realocamento de memória é executado.

O conteúdo vai sendo escrito de forma gradativa juntamente com as descargas do buffer.