Teoria da Informação

Trabalho prático nº2
Descompactação de Ficheiros `gzip´

Professor: Marco Simões

Alunos: Simão Correia Santos
Francisco José Coelho
Alexandre Gonçalves Rodrigues Ferreira

Simão Correia Santos	2021216084
Francisco José Coelho	2021226534
Alexandre Gonçalves Rodrigues Ferreira	2021236702

Introdução

Este trabalho tem como objetivo implementar o descodificador do algoritmo *deflate,* em particular levar a cabo a descompactação de blocos comprimidos com códigos de Huffman dinâmicos. Deste modo, os exercícios propostos permitem-nos adquirir conhecimento acerca da codificação das árvores de Huffman e dicionários LZ77.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Leitura do HLIT, HDIST e HCLEN

O principal objetivo do exercício 1 é ler o formato do bloco, para isso usamos a função *readbits,* que nos foi fornecida. Deste modo, para o HLIT lemos 5 bits, para o HDIST lemos 5 bits, para o HCLEN lemos 4 bits.

Obtenção da árvore HCLEN

Primeiramente, para obter a árvore HCLEN tivemos que armazenar num array os comprimentos dos códigos do "alfabeto de comprimentos de códigos", numa certa posição (*'comprimentosCodigosHCLEN'*). Segundamente, tivemos que converter os comprimentos dos códigos em códigos de Huffman do "alfabeto de comprimentos de códigos (*`conversaoCodigosHuffma*n´). Terceiramente, com os valores obtidos nas funções anteriores foi criada uma string com valores binários, com auxílio dos comprimentos (HCLEN_lens) e dos valores (HCLEN_values) (*`decimalToBinario´*). Por fim, criamos uma nova folha na árvore de Huffman com o código de cada string criada ('*fillTree*').

Funções

'comprimentosCodigosHCLEN'

De forma a obtermos os comprimentos dos códigos foi criado um array com o comprimento 19, de seguida lemos 3 bits em cada iteração usando a função *readbits* e armazena-se num array o valor lido, na sua devida posição.

'*conversaoCodigosHuffma*n'

Para converter os comprimentos dos códigos de Huffman em códigos de Huffman obtemos o bl_count, que é um array com a frequência dos comprimentos dos códigos, na primeira posição temos quantas vezes ocorre o comprimento '1', neste caso o comprimento máximo é 8 uma vez que a leitura foi de 3 em 3 bits (de 0 a 7), assim obtemos os primeiros valores de cada comprimento.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

'*decimalToBinario*'

Para passar criar uma string com valores binários usámos os comprimentos da primeira função (cLens) e os valores da segunda função (cValues), calcula-se o resto da divisão do valor de cValues por 2, enquanto o valor de cValues for maior que zero, e adiciona-se a uma lista o resto da divisão obtido, caso no final o comprimentos binário seja inferior ao respetivo valor de cLens, são adicionados zeros no final até que os dois comprimentos sejam iguais.

'*fillTree*'

Nesta função criamos uma nova folha na árvore de Huffman com o símbolo da posição onde se encontra a string no array, com o código de Huffman da string, para isto foi usada uma função que nos foi fornecida.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Obtenção da árvore HCLIT

Para obter a árvore HCLIT tivemos que ler e armazenar os HLIT + 257 comprimentos dos códigos referentes ao alfabeto de literais e de comprimentos segundo o código de Huffman de comprimentos de códigos. Para esse efeito, guardamos num array denominado HLIT_lens os comprimentos dos códigos referente ao alfabeto de comprimentos (*'arrayHLIT'*), de seguida tivemos que converter os comprimentos dos códigos em códigos de Huffman do "alfabeto de comprimentos de códigos ('*conversaoCodigosHuffma*n'), seguidamente com os valores obtidos nas funções anteriores foi criada uma string com valores binários, com auxílio dos comprimentos (HLIT_lens) e dos valores (HCLEN_values) (*'decimalToBinario'*). Por fim, criamos uma nova folha na árvore de Huffman com o código de cada string criada ('*fillTree*').

Funções

'arrayHLIT'

Nesta função criamos um array denominado HLIT_lens com comprimento HLIT + 257, enquanto a posição (pos, variável que vai ser incrementada) for menor que HLIT + 257 lemos 1 bit por cada iteração e de seguida usamos uma função que nos foi fornecida para descobrir o seu valor, caso o valor seja menor que zero fazemos novamente este procedimento, caso o valor seja menor que 16, adiciona-se ao HLIT_lens, no índice do valor de pos (HLIT_lens[pos]), o seu valor e incrementa-se 1 à pos, caso o valor seja igual a 16 vai-se buscar o valor imediatamente anterior no HLIT_lens (HLIT_lens[pos-1]), lê-se mais 2 bits e adiciona-se o valor ao HLIT_lens[pos], e incrementa-se a posição, caso seja 17 ou 18 adiciona-se ao HLIT_lens o valor 0 as vezes necessárias.

'*conversaoCodigosHuffma*n'

Semelhante à obtenção da árvore HCLEN, a única diferença é na leitura dos bits, que neste caso é de 4 em 4.

'*decimalToBinario*'

'*fillTree*'

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Obtenção da árvore HDIST

Para obter a árvore HDIST tivemos que criar um método para ler e armazenar num array os HDIST + 1 comprimentos de códigos referentes ao alfabeto de literais e de comprimentos segundo o código de Huffman de comprimentos de códigos. Para esse efeito foi usado um método muito semelhante à obtenção da árvore HCLIT, guardamos num array denominado HDIST_lens que tem comprimento HDIST + 1 os comprimentos dos códigos referentes ao alfabeto de comprimentos (o procedimento é igual ao da árvore HCLIT) (*'arrayHDIST'*), de seguida tivemos que converter os comprimentos dos códigos em códigos de Huffman do "alfabeto de comprimentos de códigos ('*conversaoCodigosHuffma*n'), seguidamente com os valores obtidos nas funções anteriores foi criada uma string com valores binários, com auxílio dos comprimentos (HDIST_lens) e dos valores (HDIST_values) (*`decimalToBinario´*). Por fim, criamos uma nova folha na árvore de Huffman com o código de cada string criada ('*fillTree*').

Funções

'arrayHDIST'

Nesta função criamos um array denominado HDIST_lens com comprimento HDIST + 1, enquanto a posição (pos, variável que vai ser incrementada) for menor que HDIST + 1 lemos 1 bit por cada iteração e de seguida usamos uma função que nos foi fornecida para descobrir o seu valor, caso o valor seja menor que zero fazemos novamente este procedimento, caso o valor seja menor que 16, adiciona-se ao HDIST_lens, no índice do valor de pos (HDIST_lens[pos]), o seu valor e incrementa-se 1 à pos, caso o valor seja igual a 16 vai-se buscar o valor imediatamente anterior no HDIST_lens (HDIST_lens[pos-1]), lê-se mais 2 bits e adiciona-se o valor ao HDIST_lens[pos], e incrementa-se a posição, caso seja 17 ou 18 adiciona-se ao HDIST_lens o valor 0 as vezes necessárias.

'*conversaoCodigosHuffma*n'

Igual à obtenção da árvore HCLIT.

'*decimalToBinario*'

'*fillTree*'

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Descompactação de Dados

Com base nos códigos de Huffman previamente obtidos, do algoritmo LZ77 e das tabelas fornecidas do Doc.2 dos "Compressed blocks", seguimos para o descompactamento dos dados comprimidos. Para o fazer, criamos uma função que coloca numa lista os valores pretendidos.

Funções

'*descompactacao*'

Nesta função criamos uma lista denominada '*output_stream*' e percorremos a árvore HLIT até encontrar uma folha. Lemos 1 bit por cada iteração, mas enquanto o valor obtido for menor que 0, continua-se a fazer este processo. Caso este seja maior ou igual que 0 posicionamos o ponteiro curNode na raiz da árvore. Se o valor for menor que 256, significa que é um literal e podemos adicioná-lo diretamente à lista. Se for igual a 256, verifica-se que é o fim do bloco e contém um break para parar o ciclo. Caso este seja maior que 256 precisa-se de se encontrar o length e a distância a recuar de cada extra bit de diferentes codes, para se aplicar o algoritmo LZ77. Com o auxílio da tabela de lenghts, calcula-se o length para diferentes casos do valor obtido anteriormente. A lógica usada para obter o length correto para o caso em que o valor seja maior que 264 e menor que 285 foi: leitura do número de extra bits correspondente + value - (code - primeiro length de cada extra bit) + (value - primeiro code de cada length) * (diferença entre dois primeiros length consecutivos do mesmo extra bit - 1).

Se for maior que 256 e menor que 265 o length é igual ao valor - 254, por último se este for igual a 285 o length é 258. Para se obter a distância a recuar utiliza-se a árvore HDIST e o seu processo de percorrimento é similar ao da árvore anterior. No entanto, neste caso, temos como auxílio as tabelas de distâncias. Se o valor obtido da folha for maior ou igual que 0 e menor que 4 a dist é o próprio valor-1. Caso seja maior que 3 e menor que 29 a dist é obtida através da seguinte lógica: leitura do número de extra bits correspondente + value1 + (diferença entre o primeiro length com o primeiro code de cada extra bit) + (value1 - primeiro code de cada extra bit) * (2**extrabits - 1).

Deste modo, temos a distância a recuar na lista e o número de elementos a copiar a partir dessa posição. Esses elementos copiados adicionam-se à mesma.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Gravação de Dados

Por fim, para se gravar os dados descompactados obtidos do passo anterior, utilizamos a função '*gravarDados*' que recebe como argumento o nome do ficheiro original e a lista '*output_stream*'. Esta função abre o ficheiro como escrita e escreve os caracteres da tabela ascii correspondentes a cada elemento do array.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702

Conclusão

Em suma, a realização deste trabalho permitiu-nos consolidar melhor a matéria teórica e prática sobre os conceitos de codificação usando árvores de Huffman e dicionários LZ77, como também alargar a aprendizagem sobre o algoritmo deflate.

Simão Correia Santos 2021216084
Francisco José Coelho 2021226534
Alexandre Gonçalves Rodrigues Ferreira 2021236702