CCM0118 — Computação I

Curso de Ciências Moleculares — Turma 22 — Segundo Semestre de 2012

Terceiro Exercício-Programa — Prazo de entrega: até **09 de novembro de 2012.**

Compressão de Arquivos

O objetivo deste exercício-programa é implementar um compressor de arquivos utilizando um algoritmo que foi proposto por J. Ziv e A. Lempel, em 1977, e que ficou conhecido como LZ77. Uma variante refinada desse algoritmo, conhecida como DEFLATE, é a base para dois formatos populares de arquivos comprimidos (ZIP e gzip), para a biblioteca de compressão de arquivos zlib e para inúmeros programas, incluindo PKZIP, WinZip, WinRAR e gzip.

Quem estiver curioso para saber mais sobre o LZ77 e sobre a utilização de sua variante DEFLATE em programas de compressão de arquivos, visite as seguintes páginas:

- `http://en.wikipedia.org/wiki/LZ77_and_LZ78`
- `http://en.wikipedia.org/wiki/DEFLATE`
- `http://en.wikipedia.org/wiki/Gzip`
- `http://en.wikipedia.org/wiki/Zlib`
- `http://en.wikipedia.org/wiki/ZIP_file_format`

Descrição do Algoritmo LZ77

A idéia fundamental é muito simples: à medida que um arquivo é lido, seqüências repetidas de caracteres são substituídas por *referências* para ocorrências prévias dessas seqüências. Essas referências são pares $(dist, compr)$, onde *dist* é a distância (número de caracteres) entre uma ocorrência da seqüência e a sua ocorrência prévia, e *compr* é o comprimento dessa seqüência. Quando um caractere c ocorrer pela primeira vez no texto, a "referência" será o par $(0, c)$.

Exemplo

Arquivo original: `le=0;while(le==0){hi=0;while(hi==0){hi=`

Arquivo comprimido: `0l` `0e` `0=` `00` `0;` `0w` `0h` `0i` 8 2 `0(` 11 3 12 2 `0)` `0{` 12 2 18 9 11 3 18 7

Repare que o arquivo original possui 39 caracteres (39 bytes), mas o arquivo comprimido possui 18 pares (36 bytes). A 9ª referência, ou seja, o par 8 2, está indicando que os caracteres `le` da primeira ocorrência da palavra `while` já ocorreram no texto a uma distância de 8 caracteres. Note agora a 11ª referência, ou seja, o par 11 3. Apesar da seqüência `le` já ter ocorrido no final da palavra `while`, é mais vantajoso registrar a repetição da seqüência `le=`, que possui 3 caracteres e ocorre a uma distância de 11 caracteres. Repare também na economia de caracteres causada pela antipenúltima referência: a seqüência `=0;while(` foi substituída pelo par 18 9.

O algoritmo de expansão (ou descompressão) é ainda mais simples: basta trocar cada referência por uma cópia do trecho referenciado por ela. No exemplo anterior, após a descompressão dos 8 primeiros pares, todos começando com 0, teremos o texto `le=0;whi`. A descompressão do 9º par, ou seja, do par 8 2, pode ser interpretada como: "olhe oito caracteres para trás e copie dois caracteres a partir daí". Obtém-se, então, o texto `le=0;while`.

Implementação do Algoritmo

Na prática, os comprimentos das repetições procuradas têm um limite máximo e as buscas não são feitas em toda a parte já vista do arquivo original, mas apenas numa 'janela' contendo o trecho mais recentemente examinado desse arquivo. Como essa 'janela' parece deslizar sobre o arquivo original, o algoritmo LZ77 é também conhecido como "compressão com janela deslizante".

Declare um vetor de caracteres *janela*, com 510 posições, indexadas de 0 a 509. A primeira metade desse vetor (posições de 0 a 254), denominada *dicionário*, armazena o trecho mais recentemente examinado do arquivo original; a segunda metade (posições de 255 a 509), denominada '*look-ahead*', armazena o próximo trecho do arquivo original a ser comprimido.

Compressão

Cada passo do processo de compressão consiste das seguintes ações:

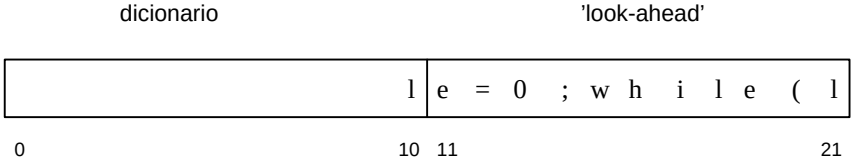
a) procure nos caracteres do '*look-ahead*' (começando da posição 255) o segmento de maior comprimento que seja uma repetição de algum segmento de caracteres do *dicionário* (posições 0 a 254) e escreva no arquivo comprimido o par $(dist, compr)$ correspondente;

b) desloque todos os caracteres do vetor *janela* de *compr* posições para a esquerda (ou uma posição, no caso de $dist = 0$;

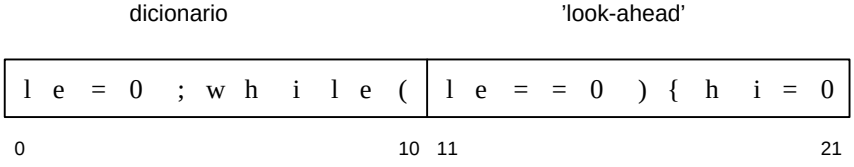
c) leia do arquivo original os próximos *compr* caracteres necessários para preencher o '*look-ahead*' (ou apenas um caractere, no caso de $dist = 0$.

Apenas para facilitar a explicação, vamos supor que o vetor *janela* tem apenas 22 posições. No exemplo visto anteriormente, teremos:

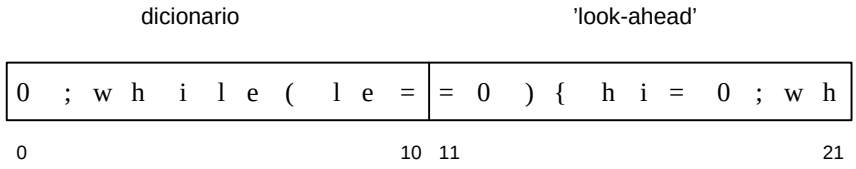
- Situação após o 1º passo (quando foi gerado o par 0 l):



- Situação após o 10º passo (já foram gerados os 10 primeiros pares, de 0 l a 0 ():



- No 11º passo ocorrerá o seguinte:
  - será descoberto que o segmento de maior comprimento do '*look-ahead*' que seja uma repetição de algum segmento do *dicionário* é `le=`, o que gera o par $(dist, compr) = (11, 3)$;
  - todos os caracteres de *janela* serão deslocados de 3 posições para a esquerda;
  - serão lidos mais 3 caracteres do arquivo original:



Note que tanto a distância *dist* quanto o comprimento *compr* podem variar apenas de 0 a 255, ou seja, cabem em um byte.

Descompressão

Para a descomprimir um arquivo que foi comprimido pelo método acima, também é preciso utilizar uma *janela* com um *dicionário* (contendo os últimos 255 caracteres já descomprimidos) e um '*look-ahead*' (onde são armazenados os caracteres que estão sendo gerados pela descompressão do par $(dist, compr)$ que acabou de ser lido).

Operação do programa

Inicialmente, seu programa deve perguntar ao usuário se deseja comprimir ou descomprimir um arquivo. Em seguida, o programa deve pedir os nomes dos arquivos de entrada e de saída. Finalmente, o programa deve realizar a tarefa desejada.

Note que o arquivo resultante de uma compressão pode ter mais bytes do que o arquivo original. (Quando ocorre esta situação?) Seu programa deve avisar caso isto ocorra. Em caso contrário, o programa deve determinar também o fator de compressão *fc*, definido como

$$fc = ((nbao - nbac)/nbao) * 100$$

onde

$nbac$ = número de bytes do arquivo comprimido e
$nbao$ = número de bytes do arquivo original.

Requisitos

(1) Organize seu programa em funções. Use funções para dividir as tarefas do seu programa em tarefas menores e mais simples. Idealmente cada função deve ser curta e fácil de entender. Se alguma função parecer longa ou complexa, crie uma ou mais funções auxiliares de modo a dividir o trabalho realizado por aquela função!

Seu programa deve conter pelo menos uma função para cada uma das seguintes tarefas:

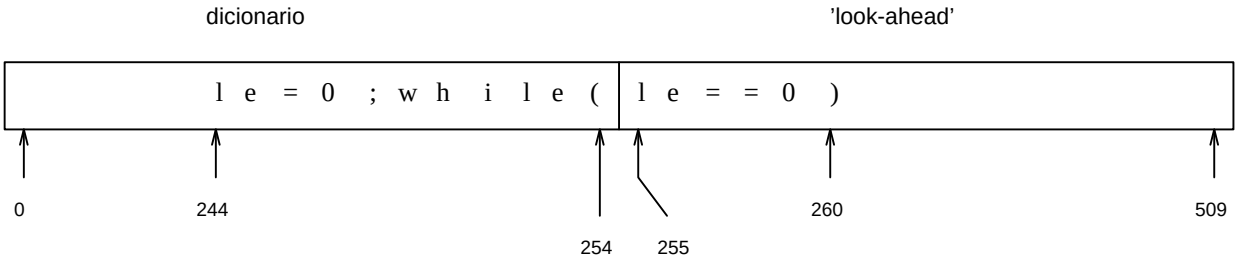
- comprimir um arquivo;

– descomprimir um arquivo;

– deslocar todos os caracteres no vetor *janela* de n posições para a esquerda;

– determinar o par $(dist, compr)$ para o próximo trecho a ser comprimido.

Para esta última funcão, considere como parâmetros o vetor *janela*, a *posição inicial* do texto que já passou para a parte do dicionário (repare que no início o dicionário está vazio e, portanto, esta posição é 255), a *posição do último caractere* do 'look-ahead' (repare que no final da compressão o 'look-ahead' vai esvaziando), e o par $(dist, compr)$ a ser determinado. Exemplo:



Nesse exemplo, a posição inicial é 244, a posição do último caractere é 260, e o par $(dist, compr)$ devolvido é (11, 3).

(2) Teste seu programa com os seguintes arquivos:

– um arquivo contendo um texto literário;

– o arquivo-fonte deste exercício-programa;

– um arquivo cujo arquivo comprimido correspondente tenha muito mais bytes do que o original;

– um arquivo cujo arquivo comprimido correspondente tenha muito menos bytes do que o original.

(3) Não é possível examinar o conteúdo de um arquivo binário diretamente num editor de textos. Para que você possa conferir os pares $(dist, compr)$ gerados durante a fase de compressão, seu programa deverá criar também um arquivo de texto contendo esses pares.

(5) Não devem ser usadas variáveis globais em nenhuma das funções do programa. (Esta recomendação é para quem já estudou mais sobre a linguagem C e sabe o que são variáveis globais.)

Bom trabalho!