Instituto de Computação da UNICAMP

Disciplina MC202: Segundo Semestre de 2014

Laboratório Nº 12

Prof. Tomasz Kowaltowski

O objetivo deste laboratório é a implementação e teste de algumas rotinas que implementam o método de compressão de Huffman. A implementação deve seguir as idéias expostas em aula. O arquivo `huffman.c` já contém as declarações de tipos e de algumas funções. Os comentários indicam a finalidade de cada uma. Algumas observações importantes:

- O arquivo de entrada pode conter qualquer caractere (um byte tem 8 bits e, portanto, 256 possibilidades).

- A função *ConstroiHuffman* deve construir, a partir de um texto original, um vetor de frequências dos caracteres e dois objetos globais (as variáveis já estão declaradas): a árvore de Huffman *Arvore* e o vetor de apontadores para as folhas *Folhas* da mesma árvore. Estas estruturas são usadas por outras funções. As posições não utilizadas do vetor *Folhas* devem ser preenchidas com apontadores nulos, para permitir a deteção de caracteres inválidos.

- A construção da árvore deverá utilizar uma fila de prioridades (*heap*) para determinar, em cada passo, a árvore parcial de peso mínimo. Para isto, deve ser construído um vetor *floresta*, inicialmente igual ao vetor *Folhas* mencionado acima e transformado em seguida numa fila de prioridade com o elemento de peso mínimo na raiz. A cada remoção de duas subárvores será realizada uma inserção da nova árvore que as combina. É importante que a construção inicial da fila utilize a versão mais eficiente do algoritmo indicado em aula para produzir resultados iguais aos esperados.

- Opcionalmente, poderá ser usada a implementação geral de filas de prioridade da tarefa 09 (possivelmente corrigida), feita pelo próprio aluno. Neste caso, deverá ser descomentada uma linha no início do arquivo incompleto `huffman.c` fornecido para que seja incluído o arquivo `heap.h` e submetido, além do arquivo `huffman.c`, o arquivo `heap.c`. Esta opção vale dois pontos extra na avaliação da tarefa.

- A função *Comprime* deve percorrer a árvore, para cada letra do texto dado, a partir da folha correspondente, seguindo na direção da raiz através dos campos *pai*. Consequentemente, a cadeia de bits obtida deverá ser ***invertida***.

- O programa principal testa as funções lendo do arquivo padrão de entrada o texto a ser comprimido. Este texto pode ser constituído de várias linhas de caracteres que são concatenadas numa única cadeia. Esta cadeia é usada para compressão. A descompressão é testada com a aplicação ao texto comprimido.

- A implementação da tarefa pode seguir uma simplificação na qual, ao invés de verdadeiros bits, deve ser produzida uma sequência de caracteres '0's e '1's que simulam os bits. Esta opção depende da definição opcional da constante PSEUDO_BITS no arquivo `hufman.c` que pode ser comentada ou não. Caso seja comentada, haverá um acréscimo de até dois pontos na nota da tarefa que utilizar bits verdadeiros. Os bits verdadeiros da representação comprimida devem ser gerados ao longo do processo de compressão. Uma solução que primeiro gera uma sequência auxiliar de pseudo-bits e depois a converte para bits verdadeiros não ganhará pontos extra. Para os que implementarem esta opção, uma dica é o uso no **gdb** do seguinte comando para imprimir uma variável em binário:

```
(gdb) p /t bits[0]
$1 = 110001
```

- A administração de memória deve utilizar, em lugar das funções `malloc` e `free`, as macros `MALLOC` e `FREE` definidas no pacote `balloc`.

- Caso seja utilizada a implementação separada da fila de prioridade, devem ser submetidos dois arquivos: `huffman.c` e `heap.c`; caso contrário, deve ser submetido apenas o primeiro.

- O número máximo de submissões é 10.
