UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br/

Professor: Rodrigo Fernandes de Mello (mello@icmc.usp.br)
Alunos PAE: Lucas Pagliosa (lucas.pagliosa@usp.br)
Felipe Duarte (fgduarte@icmc.usp.br)

Trabalho 03: Compactação de Huffman

1 Prazos e Especificações

O trabalho descrito a seguir é individual e não será tolerado qualquer tipo de plágio ou cópia em partes ou totalidade do código. Caso seja detectada alguma irregularidade, os envolvidos serão chamados para conversar com o professor responsável pela disciplina.

A entrega deverá ser feita única e exclusivamente por meio do sistema run.codes no endereço eletrônico `https://run.codes` até o dia **30 de setembro de 2016 às 30 horas e 59 minutos**. Sejam responsáveis com o prazo final para entrega, o run.codes está programado para não aceitar submissões após este horário e não serão aceitas entregas fora do sistema.

Leia a descrição do trabalho com atenção e várias vezes, anotando os pontos principais e as possíveis formas de resolver o problema. Comece a trabalhar o quanto antes para que você não fique com dúvidas e que consiga entregar o trabalho a tempo.

2 Descrição do Problema

A compactação de arquivos está presente na maioria dos programas e sistemas de computador, mesmo sem você perceber. Por exemplo, compacta-se um arquivo na hora de mandar um anexo em um email, ao fazer um donwload, no armazenamento de imagens e vídeos, etc. Este processo consiste em usar a redundância ou repetição dos dados a fim de criar uma sintetização dos mesmos e diminuir a quantidade necessária de informação (bits, por exemplo) necessária para representá-los. Existem compactações onde o processo ocorre sem perda de energia, como Fourier, por exemplo, e processos irreversíveis nos quais não é possível obter o mesmo arquivo após a compactação, como quantização de um vetor, por exemplo.

Neste trabalho, você deve implementar a técnica de compactação/descompactação sem perda de energia de Huffman, aplicada sobre arquivos de texto simples (contendo apenas da tabela ASCII, não computando símbolos como `\n`, `\t`, `\r`, etc). Para tanto, seu programa deve ser capaz de executar os seguintes comandos:

- `compactar nome.txt` - Este comando deverá compactar o arquivo de texto `nome.txt` utilizando o algoritmo de Huffman. O arquivo gerado `nome.huff` contém, além do texto binário compactado, uma tabela de símbolos e bits que servirão para descompactar o arquivo;
- `descompactar nome.huff` - Este comando deverá descompactar o arquivo `nome.huff` tomando como base um texto binário e uma tabela de símbolos/bits contidas no arquivo lido. O arquivo descompactado `nome.txt` conterá o texto legível.

IMPORTANTE: Para facilitar, como entrada serão assumidos apenas arquivos de extensão `.txt` e `.huff`. Logo, se o arquivo tiver extensão `.txt`, você deve compactar este arquivo e gerar o arquivo `.huff` correspondente. Por outro lado, se o arquivo de entrada tiver extensão `.huff`, você deve descompactá-lo e gerar o arquivo `.txt` correspondente.

3 Huffman

A codificação de Huffman usa as probabilidades de ocorrência dos símbolos (caracteres) do conjunto de dados para determinar códigos de menor tamanho aos respectivos símbolos. Para tanto, cria-se uma árvore binária quase completa (com 0 ou 2 nós filhos), chamada árvore de frequência, para moldar a probabilidade de ocorrência de cada símbolo. Nesta árvore, cada nó contém um símbolo e a sua probabilidade de ocorrência. Os nós intermediários representam a soma das probabilidades de ocorrência de todos os símbolos presentes em suas ramificações e a raiz representa a soma da probabilidade de todos os símbolos no conjunto de dados.

O processo se inicia pela junção dos dois símbolos de menor probabilidade, que são então unidos em um único nó ao qual é atribuído a soma de suas probabilidades e cujo símbolo é dado pela concatenação dos símbolos dos seus filhos. Este novo nó é então tratado como se fosse uma folha da árvore, isto é, um dos símbolos do alfabeto, e comparado com os demais de acordo com sua probabilidade. O processo se repete até que todos os símbolos estejam unidos sob um único nó raiz.

Ao determinar a frequência dos termos, monta-se a árvore de Huffman tal que nós a esquerda recebem o bit 0 e nós a direita o bit 1. Para determinar a posição dos nós e a hierarquia entre eles, monta-se a árvore baseado na frequência dos termos de menor frequência, sendo que o menor dentre eles vira o filho da esquerda e o maior o filho da direita.

Caso um arquivo possua símbolos com frequências iguais é necessário uma métrica de desempate para nos auxiliar na criação da árvore de frequência. Para este trabalho, decidiu-se "desempatar" nós pegando primeiro aqueles cujos símbolos contêm o primeiro menor carácter ASCII. A Figura 1 exemplifica o processo de criação da árvore para o texto "o oh do borogodo":

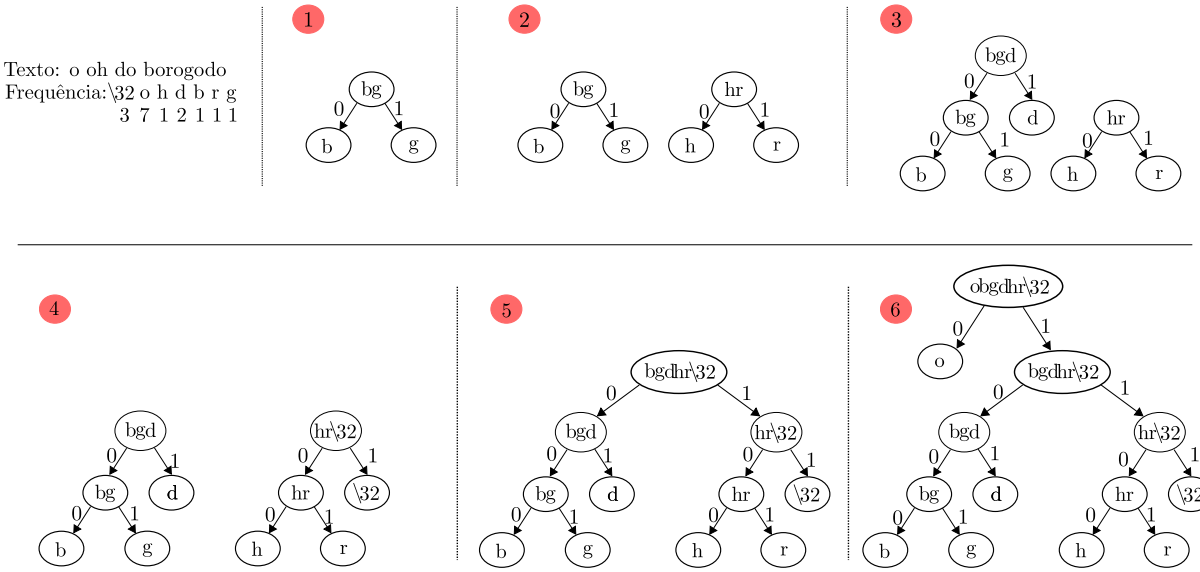

Figura 1: Processo de criação da árvore de Huffman. O símbolo \32 é utilizado apenas para facilitar na ilustração do carácter de espaço em branco.

Para ajudar na implementação do algoritmo de Huffman, utilize a função recursiva a seguir que imprime toda árvore de frequência na seguinte ordem:

```
void inOrder(Node* node)
{
  if (node)
  {
    inOrder(node->left);
    printf("%d - %s\n", node->frequence, node->symbol);
    inOrder(node->right);
```

```
  }
}
```

Utilizando o exemplo da Figura 1, o resultado da chamada da função `inOrder` seria:

```
7 - o
16 - obgdhr\32
1 - b
2 - bg
1 - g
4 - bgd
2 - d
9 - bgdhr\32
1 - h
2 - hr
1 - r
5 - hr\32
3 - \32
```

3.1 Compactação de Huffman

O comando `compactar filename.txt` deverá gerar, portanto, a árvore de frequência de Huffman a fim de auxiliar na criação de uma tabela que converta cada caractere do texto em um código binário único. Uma importante característica do processo de construção da tabela é que os caracteres com maiores frequências recebem os menores números de bits, de forma a minimizar o novo tamanho do texto. Dessa forma, cada símbolo folha (carácter) será representado pela concatenação dos bits do caminho desde a raiz até o seu nó.

A tabela de símbolos/bits tem função de auxiliar na descompactação do arquivo. Esta tabela deverá ser exibida ordenada pelo ASCII dos seus símbolos e deverá conter o caractere '-' ao seu final para indicar o fim da tabela e o inicio do texto compactado. Abaixo está a tabela de símbolos/bits para arquivo `borogodo.txt`, que contém o texto "o oh do borogodo":

```
\32  - 111
b - 1000
d - 101
g - 1001
h - 1100
o - 0
r - 1101
```

A compactação é feita substituindo-se os caracteres pelos bits correspondentes de acordo com a tabela já calculada. A cada 8 bits concatenados, salva-se o byte no arquivo de texto compactado tal como ilustrado na Figura 2.

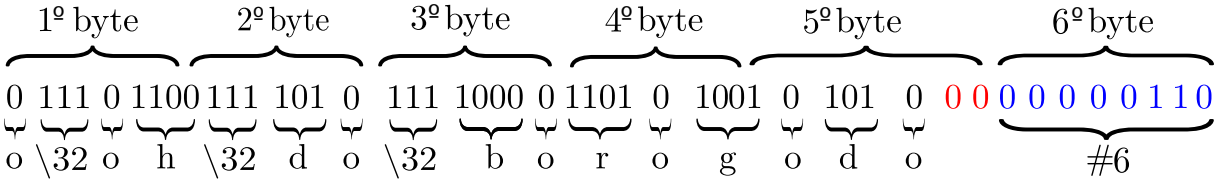

Figura 2: Geração de código binário através da compactação de Huffman.

Um problema comum neste tipo de compactação ocorre quando o número total de bits não é múltiplo de 8, de modo que não podemos salvar um byte no arquivo binário compactado. Em outras palavras, como só podemos salvar informações no disco caso estas estejam em unidades de bytes e caso o número de bits do arquivo compactado não seja múltiplo de 8, iremos adicionar

um byte extra ao final do arquivo compactado para informar a quantidade de bits que foram utilizados no último byte do texto compactado.

A Figura 2 ilustra este cenário para o texto "o oh do borogodo", que utiliza apenas 5 bits no último byte. Dessa forma, os últimos 3 bits (em vermelho) são preenchidos com zeros (pra facilitar) e um outro byte (em azul) é acoplado ao texto, indicando o número de bits válidos no agora penúltimo byte. Após a compactação, o resultado final foi:

```
\32 - 111
b - 1000
d - 101
g - 1001
h - 1100
o - 0
r - 1101
-
Binário ilegível
```

4 Descompactação de Huffman

O comando `descompactar filename.huff` deverá fazer o serviço contrario da compactação, abrindo o arquivo e retornando a mensagem original contida neste documento. Como padrão, todo arquivo `.huff` dado como entrada tem a mesma estrutura gerada pelo algoritmo de compactação, na seguinte forma:

```
Símbolo '-' bit
Símbolo '-' bit
       .
       .
       .
Símbolo '-' bit
Delimitador de fim da tabela '-'
Texto compactado
```

O resultado da descompactação do arquivo `borogodo.huff` será, por razões obvias, o texto original "o oh do borogodo".

5 Observações importantes

- Programe as impressões na tela EXATAMENTE como exemplificado no decorrer deste documento. Tome cuidado com pulos de linha, tabs, espaços, etc.
- Coloque dentro do zip todos os arquivos de código (`*.h *.c`), o makefile e um arquivo texto com o nome e número USP.