UNIVERSIDADE FEDERAL UBERLÂNDIA

CAMPUS MONTE CARMELO

CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RHUAN FLORES CUNHA FERNANDES

CLÉSIO RODRIGUES DA SILVA JÚNIOR

3° TRABALHO DE ESTRUTURA DE DADOS II

Monte Carmelo-MG

05/2023

SUMÁRIO

1 DEFINIÇÃO DE ALGORITMO SHANNON-FANO

O algoritmo de Shanon-Fano é utilizado para realizar a compressão de dados de acordo com a frequência que cada caractere ocorre em um texto, pois dependendo da ocorrência de cada letra o código gerado poderá ter tamanhos diferentes: um caractere mais frequente receberá um código menor.

Para utilizar o Shanon-Fano, deve-se ordenar os caracteres de acordo com a ocorrência de cada um. Em seguida, iremos somar todas as ocorrências e dividir o conjunto das frequências acumuladas em duas partes aproximadamente iguais. Para a primeira parte atribuímos o bit ‘0’ e para a segunda parte atribuímos o bit ‘1’. Continuamos com esse processo de soma e divisão até que reste apenas uma letra.

No final, iremos obter os códigos binários correspondentes à cada caractere que faz parte daquele texto.

2 RESOLUÇÃO DO PROBLEMA

O problema consistia em ler um texto fornecido pelo usuário, contar a ocorrência de cada caractere e codifica-los utilizando o algoritmo de Shannon-Fano. Após isso, deveria imprimir o caractere e quantas vezes ocorreu, além de utilizar essa codificação para reescrever o texto inicial.

Na primeira tentativa tentamos implementar esse problema com um algoritmo em linguagem C que funcionava da seguinte forma: o usuário inseria o texto a partir do teclado, após isso era utilizada função para contar a ocorrência de cada caractere e os armazenava numa tabela com 256 posições. Na próxima etapa, utilizamos uma lista simplesmente encadeada que já inseria os elementos (caractere, ocorrências, code[8]) em ordem decrescente de acordo com as respectivas ocorrências, e por fim utilizamos uma função para achar o valor que está mais próximo à metade da soma das ocorrências: se o elemento da lista estivesse antes ou naquela posição correspondente à metade, seria adicionado ‘0’ na primeira posição de seu código convertido, e, caso contrário seria adicionado ‘1’. Esse código funcionava bem para adicionar o primeiro bit do código, mas ficou complicado pois seria necessário adicionar recursividade para que as novas listas que fossem divididas novamente e subdivididas consecutivamente até que todas as listas terminassem com apenas 1 elemento cada.

Portanto, decidimos por utilizar C++ visto que essa linguagem possui mais métodos e funções que podem ser implementadas usando a classe String, e também optamos por realizar a leitura direto de um arquivo de texto.

Na primeira etapa, utilizamos um vetor de pares que inicialmente lê o texto contido no arquivo e em seguida realiza a contagem da ocorrência dos caracteres, um a um e as armazena no campo ocorrências (que inicialmente foi definido como 0 para todas as posições). Quando chegamos ao fim do arquivo, incrementamos uma unidade na posição 26 que corresponde ao EOF.

```
102     vector<int> frequencia;
103     int totalLetras = 0;
104     char c = fgetc(arq1);
105
106     for(int i = 0; i<256;i++){
107         frequencia.push_back(0);
108     }
109
110     do{
111         frequencia[c]++;
112         texto+=c;
113     }while((c=fgetc(arq1)) != EOF);
114
115     cout << " - Texto a ser Codificado - " << endl;
116     cout << texto << endl;
117
118     for(int i = 0;i<256;i++){
119         if(frequencia[i] > 0){
120             char letra = i;
121             frequenciaLetra.emplace_back(letra,frequencia[i]);
122             totalLetras+=frequencia[i];
123         }
124     }
125     c = 26;
126     frequenciaLetra.emplace_back(c,1);
127     texto += c;
```

Após isso, organizamos o vetor frequencialetra de acordo com a ocorrência de cada caractere e chamamos a função quantasletras para contar quantos caracteres diferentes apareceram no texto fornecido. Sabendo de tudo isso, chamamos a função Shanon que inicia o processo de compressão.

Essa função verifica se o código possui apenas 1 letra, e se for o caso atribui o valor ‘0’ e encerra a execução. Caso contrário, verifica se possui apenas 2 letras e atribui código

‘0’ para a primeira letra e ‘1’ para a segunda, do mesmo modo que realizamos os exercícios práticos em sala de aula. Também realiza a soma das frequências e divide por 2 para achar a posição que está mais próxima da metade e irá servir de referência para as próximasetapas.

Usamos a função Shanon2 que serve como uma auxiliar para recursividade, visto que ela também realiza essas etapas mencionadas e concatena os bits correspondentes ao código de cada caractere (na variável codificação). Separamos

Após isso, já temos o código correspondente a cada caractere e podemos seguir para a última requisição do exercício: percorremos letra por letra do texto fornecido e realizamos a conversão de cada uma delas para código binário e imprimimos essa codificação binária referente ao texto dado.

```
60  string converteTexto(string texto, map<char,string> codigos){
61      string s = "";
62      for(int i = 0; i<texto.size();i++){
63          char c = texto[i];
64          s += codigos[c];
65      }
66      return s;
67  }
68
69  char BytesToBits(string s){
70      int somador = 1;
71      int caractere = 0;
72      for(int i = 0; i<8;i++){
73          if(s[i] == '1'){
74              caractere += somador;
75          }
76          somador = somador * 2;
77      }
78      char c = caractere;
79      return c;
80  }
```

Por fim, separamos cada 8 posições dos bytes e convertemos para 1 bit, os armazenamos em outra string, para que no final possamos também exibir a sequência em bits.

Assim, mostrando o código em funcionamento, primeiramente lemos um texto presente em um arquivo e printamos ele na tela:

```
 - Texto a ser Codificado -
Nao acredite em algo simplesmente porque ouviu. Nao acredite em algo simplesmente porque todos falam a respeito. Nao acr
edite em algo simplesmente porque esta escrito em seus livros religiosos. Nao acredite em algo so porque seus professore
s e mestres dizem que e verdade. Nao acredite em tradicoes so porque foram passadas de geracao em geracao. Mas, depois d
e muita analise e observacao, se voce ve que algo concorda com a razao e que conduz ao bem e beneficio de todos, aceite-
o e viva-o.
```

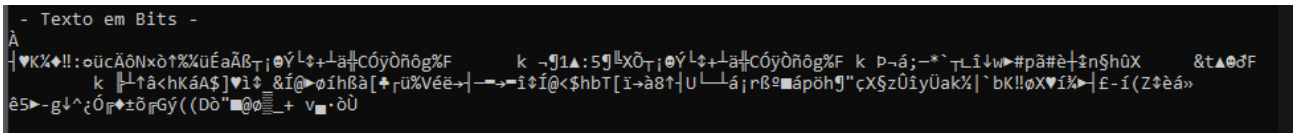
Com o texto devidamente lido começamos a fazer a contagem de todos os seus caracteres e armazenar em um vetor para futura manipulação. Com a frequência das letras devidamente contadas, começamos a conversão para shannon, através da função e depois printamos o caracter, sua frequência e seu devido código de codificação conforme a imagem abaixo:

```
 - Caracter - Frequencia - Codigos -
→ : 1 = 1111111
M : 1 = 1111110
- : 2 = 1111101
, : 3 = 1111100
z : 3 = 111101
b : 3 = 1111001
f : 4 = 1111000
N : 5 = 111011
. : 6 = 111010
n : 7 = 111001
g : 8 = 111000
v : 8 = 11011
q : 8 = 11010
l : 12 = 11001
p : 12 = 110001
u : 14 = 110000
c : 17 = 1011
t : 17 = 10101
d : 18 = 10100
m : 20 = 1001
i : 23 = 10001
r : 25 = 10000
s : 34 = 011
a : 42 = 0101
o : 46 = 0100
e : 71 = 001
  : 82 = 000
```

Agora, com os vetores devidamente preenchidos podemos começar a compressão do texto, primeiramente pegamos aquele texto lido e começamos a converter cada char dele conforme seu código, assim printamos:

```
 - Texto em Bytes -
1110110101010000001011011100000011010010001101010010000011001000010111001111000010000001110001100111000111001001011110010
0111100110101001000110001010010000110101100000010000100110000110111000111000011101000011101101010100000010110111000000011
0100100011010100100000110010000101110011110000100000011100011001110001110010010111001001111001101010010001100010100100000
1101011000000100010101010010100010001100011110000101110010101100100001010001000000101111000100110001101010100111010000011
1011010101000000101101110000001101001000110101001000001100100001011100111100001000000111000110011100011100100101110010001
1110011010100100011000101001000011010110000001000000101110101010100000010111011100001000110101010000000110010000110011100
0011000110011000111011100000010001100010000000111001100011110001000101000110100011111010000111011010101000000101101110000000
0110100100011010100100000110010000101110011110000100000011010000011000101001000011010110000001000011001110000001100011000
1100000100111100000010110110100100000001011000000100010010010111010110000000101100010100100011111010011001000110101100000010
0000100011011001100001010001011010000111101000011101101010100000010110111000000110100100011010100100000110010001011000
0010110100100011011010000101100001101000000110001010010000110101100000010001111000010010000001011001000110001010101011011010
1101000101011000101000010001110000011000000101101101010100000000110010001110000011000000101101101010100111010000111111100101
0111111110000001010000111000101001000101100010100001000100111000010001101010101010000101111001010111001100010110010000010000
1001111001011001100001101101011011010101001111100000011001000110110100101100100011011001000110101100000001000010111001111
0000100000010110100111001101101001000010100010100010110100100100001010001000000101111101010101000000001000110101100000001000
1011010011100110100110000111101000010101000000111100100110010000001000111100100111100100111110001000110111000101000000010100
0010001010101001010001000111111110000001011011001100011010100111111010100000000100011011100011101101011111101010011101011
1111
```

Por fim, com o texto em bits devidamente feito, começamos a agrupar eles de 8 em 8 bits (1 byte) e converter esse byte para seu valor decimal. E a partir desse valor decimal fazemos a compressão, vendo o valor referente na tabela ASCII e escrevendo esse caracter no texto comprimido, sendo o resultado este a baixo:



3 CÓDIGO:

```
#include <bits/stdc++.h>

using namespace std;

vector<string> Codificacao;
int quantasLetras;

//Nessa função de shannon, ela somente escreve no vector de codificação quando
entra
//numa encruzilhada da árvore que irá acabar, ou seja, quando, ao fazer o shannon,
sobra um par ou um elemento sozinho
//assim, nesses cassos ele escreve no vector, entretanto, se isso não acontecer,
ele armazena essa escrita na string escrita
//e vai repetindo o processo até chegar na encruzilhada e escrever todos os chars
void shannon(int inicio, int fim, vector<pair<char, int>> frequenciaLetra, string
escrita){
    if(inicio == fim){
        if(escrita == ""){
            Codificacao[inicio] += "0";
        }
        Codificacao[inicio] += escrita;
        return;
    }
    if(inicio == fim-1){
        Codificacao[inicio] += escrita + "1";
        Codificacao[fim] += escrita + "0";
        return;
    }
    int total = 0;
    for(int i = inicio; i<=fim;i++){
        total += frequenciaLetra.at(i).second;
    }
    total = total/2;
    int index = inicio;
    total -= frequenciaLetra.at(index).second;
    while(total>0){
```

```
        index++;
        total -= frequenciaLetra.at(index).second;
    }
```

```
shannon(inicio, index-1, frequenciaLetra,escrita+"1");
    shannon(index, fim, frequenciaLetra,escrita+"0");
}

//Função para converter uma string texto para um texto de bits (0s e 1s), pegando
cada char na cadeia
//e vendo o valor dele na codificação, convertendo o mesmo para um bit
string converteTexto(string texto, map<char,string> codigos){
    string s = "";
    for(int i = 0; i<texto.size();i++){
        char c = texto[i];
        s += codigos[c];
    }
    return s;
}

//Função que converte um byte com 8 bits (00010101) para um char, assim fazendo uma
compressão
//basicamente eu converto essa cadeia de bits para seu referencial em decimal
//e esse decimal eu converto para a tabela ASCII
char BytesToBits(string s){
    int somador = 1;
    int caractere = 0;
    for(int i = 0; i<8;i++){
        if(s[i] == '1'){
            caractere += somador;
        }
        somador = somador * 2;
    }
    char c = caractere;
    return c;
}

//Função que desconverte o texto, pegando cada char da cadeia e transformando em 0s
e 1s atraves
//da descodificação passada como parâmetro
string desconverteTexto(string texto, map<char,string> descodificarBitsBytes){
    string s = "";
    for(int i = 0; i<texto.size();i++){
        s += descodificarBitsBytes[texto[i]];
    }
    return s;
}

int main (){
    //Abrindo Arquivo
    FILE *arq1;
    arq1 = fopen("Texto.txt","r");
    if(arq1 == NULL){
```

```
        cout << "Arquivo Inexistente";
        return 0;
    }
```

```
    //Contando a Frequencia das Letras
    //primeiro criando um vetor de tamanho 256
    //para comportar todos os chars da ASCII
    vector<pair<char, int>>frequenciaLetra;
    string texto;
    vector<int> frequencia;
    int totalLetras = 0;
    char c = fgetc(arq1);

    for(int i = 0; i<256;i++){
        frequencia.push_back(0);
    }


    //Lê o texto e vai contando a sua frequencia no vetor frequencia
    do{
        frequencia[c]++;
        texto+=c;
    }while((c=fgetc(arq1)) != EOF);

    cout << " - Texto a ser Codificado - " << endl;
    cout << texto << endl;

    //criando um par, char e sua frequencia,
    //e adicinando todos esses que a frequencia é maior que 0
    for(int i = 0;i<256;i++){
        if(frequencia[i] > 0){
            char letra = i;
            frequenciaLetra.emplace_back(letra,frequencia[i]);
            totalLetras+=frequencia[i];
        }
    }
    //adicionando o EOF
    c = 26;
    frequenciaLetra.emplace_back(c,1);
    texto += c;
    //Ordenando o vetor de forma crescente de frequencia
    sort(frequenciaLetra.begin(),frequenciaLetra.end(), [] (const auto &x, const
auto &y) {return x.second < y.second;});

    quantasLetras = frequenciaLetra.size();
    for(int i = 0; i<quantasLetras;i++){
        Codificacao.push_back("");
    }

    //utilizando a função shannon, que precisa do inicio do vector, seu tamanho-1,
a tabela de frequencia
    //que sera usada para criar a codificação e uma string vazia para escrever
depois
```

```
    string escrita = "";
    shannon(0,(quantasLetras-1),frequenciaLetra,escrita);
```

```
map<char,string> codigos;
    cout << "\n - Caracter - Frequencia - Codigos - " << endl;
    for(int i = 0; i<quantasLetras;i++){
        cout << frequenciaLetra.at(i).first << " : " <<
frequenciaLetra.at(i).second << " = " << Codificacao.at(i) << endl;
        codigos[frequenciaLetra.at(i).first] = Codificacao.at(i);
    }

    //Convertendo o texto para Bytes
    string textoBin = converteTexto(texto,codigos);
    cout << "\n - Texto em Bytes - " << endl;
    cout << textoBin << endl;

    //com o texto em 0s e 1s, a gente agrupa ele 8 a 8
    //e vai passando cada um desses para a função que converte 1 byte para char,
    //adicionadno o resultado na string de TextoBits
    string textoBits = "";
    int tamanho = 0;
    int tamanho2 = 8;
    while(tamanho2<=textoBin.size()-1){
        string converte = "";
        if(tamanho2 > textoBin.size()-1){
            int excedente = tamanho2 - textoBin.size()-1;
            for(int i = tamanho; i<excedente;i++){
                converte += textoBin[i];
            }
            for(int i = excedente; i<tamanho2;i++){
                converte += "0";
            }
        }else{
            for(int i = tamanho; i<tamanho2;i++){
                converte += textoBin[i];
            }
        }
        char c = BytesToBits(converte);
        textoBits.push_back(c);
        tamanho += 8;
        tamanho2 += 8;
    }

    cout << "\n - Texto em Bits - " << endl;
    cout << textoBits << endl;

    return 0;
}
```