UNIVERSIDADE DO MINHO

Projecto de Métodos de Programação 1

Espicificação algorítmica de descodificação do codigo de Morse

Grupo 11

O objectivo deste projecto é compreender os principios da execusão das operações da descodofocação dos codigos a partir das suas lógicas da transformação do texto A em texto B. Este trabalho pode parecer bastante interessante devido ao famoso codigo de Morse.

Elementos de Grupo 11:

Nome: Ângelo Abreu

Nº: 57144

e-mail: pimpinizi@gmail.com

Nome: Eduardo Machado

Nº: 58668

e-mail: machadinho30@hotmail.com

Nome: Yaroslav Miakov

Nº: 58652

e-mail: iam.yaroslav@hotmail.com

Algoritmo não refinado:

1. Memorisar o codigo de Morse
2. Memorisar o dicionario das palavras (em Inglês)
3. Ler uma sequencia de codigo de Morse
4. Traduzir a linguagem normal
    a. Comparar o codigo de Morse com alguma sequencia dos caracteres no texto codificado pelo codigo de Morse.
    b. Verificar se esta sequencia corresponde a alguma letra do codigo de Morse
    c. Ver quantas combinações das palavras existentes no dicionario podem haver na sequencia de codigo de Morse
5. Mostrar o numero das traduções possíveis

Algoritmo refinado:

1. Criar o dicionario das palavras Inglesas na forma de uma matriz de 10000 linhas e N colunas, onde N será o numero dos caracteres da palavra mais cumprida, em que cada elemento vai ser uma letra de uma palavra, mas para palavras com numero caracteres menores do que N os espaços sobra. Chamar esta matriz dic[10000][N]
2. Criar um espaço para a matriz do dicionario auxiliar, chamada “savedDic[][]”
3. Criar o abecedario do codigo de Morse para as letras inglesas.
4. Ler o texto codificado em codigo de Morse:
   Isto é criar uma matriz-linha, code[], cujos elementos serão os caracteres: “pontos” e “traços”
5. Contar o numero de caracteres deste texto
6. Atribuir este numero de caracteres a variavel “FimDaSequencia”
7. Atribuir a uma variavel de contagem, i, o valor de -1
8. Criar uma variavel “lengthofWords” que vai guardar o numero das letras descodificadas
9. Criar uma matriz, “savedWord[][]” que vai guardar palavras descodificadas
10. Executar a função recursiva seguinte:
    F(i, l, k, z, x, n, p, q, code[], letter, dic[][], savedDic[][], letter, coun, lengthofWords):

Para i:= i+1 ate que i=FimDaSequencia fazer

<<A variavel “i”, o contador do caracter, teve no inicio o valor de -1 para facilitar a construção do algorismo; então, a contagem começa de caracter numero 0 (que é (-1+1))>>

Passo (a):

Se code[i]=”.” então fazer

letter:=”E”
senão letter:=”T”.

Ir para o passo (b) acabar;

Se code[i]=”.” e code[i +1]=”.“ então fazer

letter:=”I”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ então fazer

letter:=”A”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ então fazer

letter:=”M”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”.“ então fazer

letter:=”N”

Ir para o passo (b) acabar;

Se code[i]=”.” e code[i +1]=”.“ and code[i +2]=”.” então fazer

letter:=”S”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”.“ e code[i +2]=”-” então fazer

letter:=”U”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ e code[i +2]=”.” então fazer

letter:=”R”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ e code[i +2]=”-” então fazer

letter:=”W”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ e code[i +2]=”-” então fazer

letter:=”O”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”.“ e code[i +2]=”.” então fazer

letter:=”D”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ e code[i +2]=”.” então fazer

letter:=”G”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ e code[i +2]=”-” então fazer

letter:=”O”

Ir para o passo (b) acabar;

Se code[i]=”.” e code[i +1]=”.“ e code[i +2]=”.” e code[i +3]=”.” então fazer

letter:=”H”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”.“ e code[i +2]=”.” e code[i +3]=”-” então fazer

letter:=”V”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ e code[i +2]=”-” e code[i +3]=”-” então fazer letter:=”J”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”.“ e code[i +2]=”.” e code[i +3]=”.” então fazer letter:=”B”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ e code[i +2]=”.” e code[i +3]=”.” então fazer letter:=”L”

Ir para o passo (b) acabar;

Senão se code[i]=”.” e code[i +1]=”-“ e code[i +2]=”-” e code[i +3]=”.” então fazer letter:=”P”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ e code[i +2]=”.” e code[i +3]=”-” então fazer letter:=”Q”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”.“ e code[i +2]=”.” e code[i +3]=”-” então fazer letter:=”X”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”.“ e code[i +2]=”-” e code[i +3]=”-” então fazer letter:=”Y”

Ir para o passo (b) acabar;

Senão se code[i]=”-” e code[i +1]=”-“ e code[i +2]=”.” e code[i +3]=”.” então fazer letter:=”Z”

Ir para o passo (b) acabar;

Passo (b):

```
Para l:=0 ate que l=10000 fazer
    Para k:=0 ate que k=N fazer
        savedDic[l][k]:=dic[l][k];
```

acabar;

acabar.

Para l:=0 ate l=N fazer

se dic[l][k]=letter then do

i:=i+1; k:=k+1; retornar F(com variaveis modificadas: i, k); acabar;

senão se dic[l][k-1]!=0 and dic[l][k]=0 então fazer

para q:=0 ate que q=N fazer

savedWord[p][q]:=dic[l][q]; q:=q+1; acabar;

para z:=0 ate que z=P do

para x:=0 ate que x=q fazer

se savedWord[z][x]=(“T” ou “E “) então lengthofwords:=lengthofwords+1;

savedWord[z][x]=(“A” ou “I “ ou “M” ou “N”) então lengthofwords:=lengthofwords+2;

savedWord[z][x]=(“S” ou “U “ou “R“ ou “W” ou “D” ou “G” ou “K” ou “O”) então lengthofwords:=lengthofwords+3;

savedWord[z][x]=(“B” ou “C “ ou “F “ou “H“ ou “J” ou “L” ou “P” ou “Q” ou “V” ou “X” ou “Y” ou “Z”) então lengthofwords:=lengthofwords+4;

acabar;

acabar;

z:=z+1; x:=x+1;

se lengthofwords<= FimDaSequencia então do

para l:=0 ate que l= FimDaSequencia fazer

para k:=0 ate que k=longestWord fazer

savedDic[l][k]:=dic[l][k];

acabar;

acabar;

para l:=0 ate que l=FimDaSequencia fazer

para k:=0 ate que k:=N fazer

dic[l][k]:=0;

acabar;

acabar;

para l:=0 ate que l=FimDaSequencia fazer

para k:=0 ate que k:=N fazer

dic[l][k]:=savedDic[l][k];

acabar;

acabar;

i:=i+1; retornar F(com variaveis modificadas: i);

acabar;

senão se lengthofwords= FimDaSequencia então &count:=&count+1;
senao fazer nada; acabar;

acabar;

senão se dic[l][k]!=letter then fazer

para n:=0 ate que n:=N fazer

dic[l][n]:=0; fazer;

l:=l+1;

acabar;

senão fazer nada; acabar;

acabar.

<<Fim da função "F">>

11. Imprimir "O numero das opções possiveis para descodificar é ", count.
12. Terminar.

Legenda para o alrogitmo:

1. O sinal “:=” significa atribuição do valor para o variavel do lado esquerdo
2. O sinal “=” significa a verificação da igualdade dos dois lados da equação
3. O sinal “!=” significa a verificação da inigualdade dos dois lados da equação
4. O sinal “&” marca o espaço da memoria onde se encontra a variavel de onde ela se tira

Análise crítica

Durante o processo da construção do algoritmo foram encontradas varias dificuldades de achar o caminho pelo qual vai se executar o programa. Foi decidido nao utilizar os vectores das letras e palavras em codigo de Morse para verificar todas as opções possiveis de encontrar frases validas, porque estas contagens de força bruta precisam de ter mais tempo para do que o algorismo que temos. Por exemplo, se tivesse 1000 caracteres na mensagem inicial, era possivel de ter 2^1000 ate 2^250 combinações dos vectores dentro do vector que guarda o texto de codigo de Morse. Por isso utilizamos algumas matrizes que, como nos achamos, facilitam a contagem de numero das frases possiveis.

Se construir uma diagrama de execução deste algoritmo, aparecia uma “arvore”, devido a utilização da função recursiva “F”.