UNIVERSITE DE M'SILA

Faculté de Technologie — Département D'électronique

Filière : Télécommunication — 1 ère Année Master (STLC)

Module :

TP Codage et compression

TP N° : 02

Codage de Shannon-Fano

*Un compte rendu par binôme doit être transmis au plus tard le 24 Mars 2022 à 23h59 sur **Moodle***

***Attention :** le plagia (copier/coller) est strictement interdit. Ceci sera rigoureusement contrôlé*

Objectifs du TP :

- Effectuer le codage d'une chaine de symboles en utilisant l'algorithme de Shannon-Fano.
- Calculer les paramètres statistiques de codage Shannon-Fano (H, L, E, R, Taux de compression).
- Comparaison de performance avec le codage de Huffman.
- Exploiter quelques fonctions de Matlab.

Manipulation (sur Matlab toujours, ceux qui travaillent sur Python +3) :

Considérons le message de source suivant :

msg = {she sells sea shells}

1) Calculer analytiquement le code du Shannon-Fano qui compresse **msg**.
2) Trouver le code du Shannon-Fano qui compresse **msg** en utilisant la fonction Matlab (**ShannonFanoFunc.m**).
3) Calculer (en utilisant toujours Matlab) les résultats fournies par l'algorithme de Shannon-Fano en termes de : Entropie de la source, Longueur de message compressé, Taux de compression, Efficacité et Redondance du code. Constater.

Remarque : taux_compression= (1 - (taille_comp/ taille_originale))*100

Démarrage en Matlab :

```
close all; clear; clc;
symbols = {'s', 'e', …….};   % à compléter
p = [0.3 0.2 ……];            % à compléter
% à compléter
```

La fonction Matlab **ShannonFanoFunc.m** :

```
function [code1, average_length] = ShannonFanoFunc(p)
set(0,'RecursionLimit',1e4);
% p1 = probability vector
% code1 = corresponds codewords
% average_length is the expected codeword length
% check if p1 is row vector or column vector

if ((sum(p>=0)~=length(p)))
    error('Enter a probability vector');
end
p = p/sum(p);
if(length(p)>2)

[pdes,idx] = sort(p,'descend');
qsum = (2*cumsum(pdes))-1;
[~,idx1] = min(abs(qsum));
    if((idx1>1)&&(idx1<length(pdes)-1))
        q1 = pdes(1:idx1);                  % break into left half
        q2 = pdes((idx1+1):length(pdes));   % right half
        old_code1 = ShannonFanoFunc(q1);    % recursive call left
        old_code2 = ShannonFanoFunc(q2);    % recursive call right
        new_code = [strcat('0',old_code1) strcat('1',old_code2)]; % code 0 to left
    elseif(idx1==1)
        q1 = pdes(1);
        q2 = pdes(2:length(pdes));
        old_code1 = ShannonFanoFunc(q1);
        old_code2 = ShannonFanoFunc(q2);
        new_code = [old_code1  strcat('1',old_code2)];
    else
        q1 = pdes(1:((length(pdes)-1)));
        q2 = pdes(length(pdes));
        old_code1 = ShannonFanoFunc(q1);
        old_code2 = ShannonFanoFunc(q2);
        new_code = [strcat('1',old_code1)  old_code2];
    end
    code1(idx) = new_code;

elseif(length(p)==2)
    code1 = {'0', '1'};
else
    code1 = {'0'};
end
length1 = cellfun(@length, code1);
average_length = sum(length1.*p);
end
```