**U19EC045 | DCOM | LAB 5**

**AIM**

To study and simulate Shanon-fano coding as a source coding technique.

**THEORY**

## Shanon-Fano coding is also one of the source coding technique.

## It is a variable length encoding scheme, that is, the codes assigned to the symbols will be of varying length.

## Shanon-Fano coding constructs a prefix code based on a set of symbols and their probabilities.

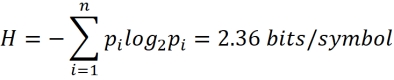
## Shanon-fano coding uses cumulative distribution function in the generation of code word.

## Shanon-fano coding is not a optimal technique as it is not possible to achieve lowest possible expected codeword length like Huffman.

## Shannon–fano coding can be used in the IMPLODE compression method, where it is desired to apply a simple algorithm with high performance and minimum requirements for programming.

## **FORMULAS:**

### *ENTROPY:*

**

### *AVERAGE CODEWORD LENGTH:*

### *IMG_257*

### *EFFICIENCY:*

### *IMG_258*

## **Steps of Shanon Fano Coding Algorithm:**

1. Arrange the probabilities of source symbols in descending order.
2. Partition the set into two sets that are as close to equiprobable as possible.
3. Assign 0 to the upper set and 1 to the lower set.
4. Recursively apply the steps 2 and 3, such that each time sets are partitioned into nearly equal probabilities as possible and bits are assigned to partitioned sets .
5. This has to be continued until further partitioning not possible

**MATLAB CODE**

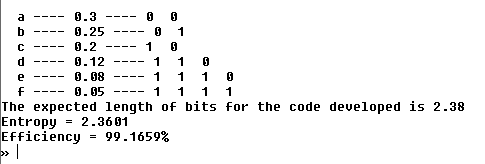
generateCode.m //generator function

|  |
| --- |
| **function output = generateCode(prevRes, array, offset)**  ***% base case if lengeth is 1 return prevRes***  **if(length(array) == 1)**  **output = prevRes;**  **return;**  **end**  ***% get the index for equi partition of array***  **equiIndex = equiPartitionIndex(array);**  ***% assign 0 to left half of array***  **for i = 1:equiIndex**  **prevRes{offset+i-1} = [prevRes{offset+i-1}, 0];**  **end**  ***% assign 1 to right half of array***  **for i = equiIndex+1:length(array)**  **prevRes{offset+i-1} = [prevRes{offset+i-1}, 1];**  **end**  ***% recursively call generateCode for left and right half of array***  **prevRes = generateCode(prevRes, array(1:equiIndex), offset);**  **prevRes = generateCode(prevRes, array(equiIndex+1:end), equiIndex+offset);**  ***% return the result***  **output = prevRes;**  **end**  ***% function to get equi partition index of array***  **function index = equiPartitionIndex(array)**  **minDiff = Inf;**  **for i = 1:length(array)**  **diff = abs(sum(array(1:i)) - sum(array(i+1:end)));**  **if diff < minDiff**  **minDiff = diff;**  **index = i;**  **end**  **end**  **end** |

lab5.m //driver code

|  |
| --- |
| ***% symbols and corresponding probability***  **letters=['a', 'b', 'c', 'd', 'e', 'f'];**  **probabilities = [0.3, 0.25, 0.20, 0.12, 0.08, 0.05];**  ***% empty cells to store the generated code***  **resultCode=cell(1,length(probabilities));**  ***% sort the probabilities in descending order***  **[sortedProb, sortedIndices] = sort(probabilities, 'descend');**  ***% generate the code***  **resultCode=generateCode(resultCode, sortedProb, 1);**  ***% display the generated code***  **for i=1:length(probabilities)**  **disp(['  ',letters(sortedIndices(i)), ' ---- ' ,num2str(sortedProb(i)), ' ---- ', num2str(resultCode{i}) ]);**  **end**  **resultCodelength = [];**  **for index = 1:length(resultCode)**  **resultCodelength=[resultCodelength,length(resultCode{index})];**  **end**  ***% display the average length of the code***  **avgLength = sum(sortedProb.\*resultCodelength);**  **disp(['The expected length of bits for the code developed is ' num2str(avgLength)]);**  ***%calculate entropy***  **ent = -sum(probabilities.\*log2(probabilities));**  ***%display result***  **disp(['Entropy = ' num2str(ent)]);**  ***%calculate efficiency***  **disp(['Efficiency = ' num2str(ent\*100/avgLength) '%']);** |

**OUTPUT**



**CONCLUSION**

In this practical, we have implemented Matlab code for generating Shanon Fano code and also calculated average codeword, entropy and efficiency for the same. The efficiency observed was bit lower than huffman coding.