* **CLASS ExtraInfo**
  → [**Logic/Idea**]
  - use to create 3D array of codes and 2D array of characters which will be printed at beginning of file
  - file this is not required anywhere in encoding but this structure is important for decoding

  - eg....
  - suppose our file contain this line.... "**Huffman coding is a data compression algorithm.**"
  - our tree will generate following codes which we will store in 3D array
  - and corresponding characters will be stored in 2D array

3D array of codes..
finally….

```
dim1   dim2
 []--->  []-->[0|1|0]
         []-->[0|1|1]
 []--->  []-->[1|0|0|0]
         []-->[1|0|0|1]
         []-->[1|0|1|0]
         []-->[1|0|1|1]
         []-->[1|1|0|0]
 []--->  []-->[0|0|0|0|0]
         []-->[0|0|0|0|1]
         []-->[0|0|0|1|0]
         []-->[0|0|0|1|1]
         []-->[0|0|1|0|0]
         []-->[0|0|1|0|1]
         []-->[0|0|1|1|0]
         []-->[0|0|1|1|1]
         []-->[1|1|0|1|0]
         []-->[1|1|0|1|1]
         []-->[1|1|1|0|0]
         []-->[1|1|1|0|1]
         []-->[1|1|1|1|0]
         []-->[1|1|1|1|1]
```

2D array of characters..
Finally….

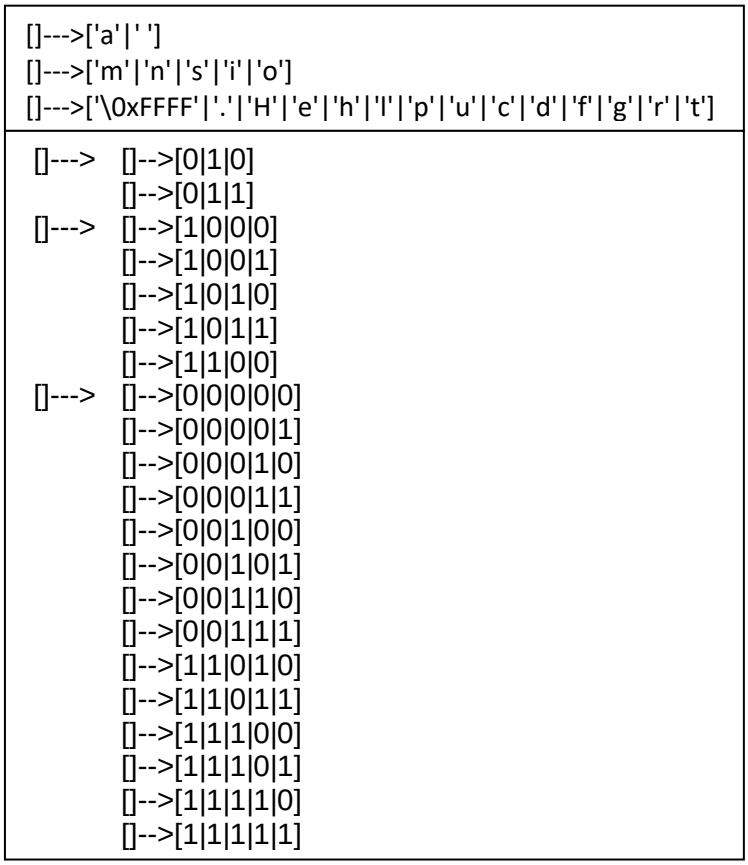
```
dim1
[]--->['a'|' ']
[]--->['m'|'n'|'s'|'i'|'o']
[]--->['\0xFFFF'|'.'|'H'|'e'|'h'|'l'|'p'|'u'|'c'|'d'|'f'|'g'|'r'|'t']
```

  - codes are arranged in sorted order of their length so at a time of decoding we can search for
  - first 3bit if code not found then 4bit then 5bit and so on...

  → [**methods**]
  - **insert(ch : int,code : byte[]):boolean**
    - **findCode(root : TNode,id : int) method** of HuffmanEncode class will call this method while generating codes from given huffman tree
    - so at a time of insertion if ch[dim1] or code[dim1] is null

then it will allocate 2^level bytes of array to corresponding ch[dim1] and code[dim1]

* **FILE STRUCTURE**
    - "txt"
    - CONS.SIG



EXTRA INFORMATION

    - 00010 00111 11100 11100 1000 010 1001 011 11010 1100 11011 1011 1001 11101 011 1011 1010 011 010 011 11011 010 11111 010 011 11010 1100 1000 00110 11110 00011 1010 1010 1011 1100 1001 011 010 00101 11101 1100 11110 1011 11111 00100 1000 00001 **00000**

* /