Ternary Huffman coding

20131292 SeoYongHyeok

Programming language : python3

Algorithm :

```
1  import queue
2
3  class HuffmanNode(object):
4      def __init__(self, left=None, mid=None ,right=None, root=None):
5          self.left = left
6          self.mid = mid
7          self.right = right
8          self.root = root     # Why?  Not needed for anything.
9      def children(self):
10         return((self.left, self.mid, self.right))
11 def create_tree(probabilities):
12     p = queue.PriorityQueue()
13     for value in probabilities:    # 1. Create a leaf node for each symbol
14         p.put(value)               #    and add it to the priority queue
15     while p.qsize() > 2:         # 2. While there is more than two node
16         l, m, r = p.get(), p.get(), p.get()  # 2a. remove three highest nodes
17         node = HuffmanNode(l, m, r) # 2b. create internal node with children
18         p.put((l[0]+m[0]+r[0], node)) # 2c. add new node to queue
19
20     return p.get()               # 3. tree is complete - return root node
21
22 # Recursively walk the tree down to the leaves,
23 #   assigning a code value to each symbol
24 def walk_tree(node, prefix="", code={}):
25     w, n = node
26     if isinstance(n, str):
27         code[n] = prefix
28     else:
29         walk_tree(n.left, prefix + "2")
30         walk_tree(n.mid, prefix + "1")
31         walk_tree(n.right, prefix + "0")
32     return(code)
```

Line 1: using Queue

Line 3~10 :

Make a huffmanNode.

There are three children at the each node(Left, mid, right)

Line 11~20 :

Make a tree with the Huffman nodes. With the 'PriorityQueue'

It make a queue in order.

Line 24~32 :

Return the resulting codeword by 'walk down processing' on tree from root node.

```
34 size = int(input('Enter the number of your codes :'))
35
36 probability = [] # create empty list
37
38 for i in range(0, size): # set up loop to run 5 times
39     print('Please enter the',i+1,'input')
40     number = str(input('The signal number:             ')) # prompt user for number
41     prob = float(input('The probability:               '))
42     tup=(prob,number)
43     probability.append(tup) # append to our_list
44 node = create_tree(probability)
45
46 leng_sum =0
47 var_sum=0
48
49 print('|--------------THE-SOURCE-CODE----------------|')
50 print(' Prob  CodeWord')
51 code = walk_tree(node)
52
53 for i in sorted(probability, reverse=True):
54     print ('{:6.3f}'.format(i[0]), code[i[1]])
55     leng_sum+=i[0]*len(code[i[1]])
56 for i in sorted(probability, reverse=True):
57     var_sum+=i[0]*((len(code[i[1]])-leng_sum)**2)
58 print('|--------------MEASURES-------------------|')
59 print(' The expected Length : ','{:6.2f}'.format(leng_sum))
60 print(' The variance :        ','{:6.2f}'.format(var_sum))
```

Line 34~43:

Input the number of signals and a empty list for probability

Line 44 : Do making a tree

Line 51: Do making a codwords

Line 53~:

Outputs

Sample I/O

```
Ses-Mac-mini-2:c syh4661$ python huffman.py
Enter the number of your codes :5
Please enter the 1 input
The signal number:              1
The probability:                0.5
Please enter the 2 input
The signal number:              2
The probability:                0.3
Please enter the 3 input
The signal number:              3
The probability:                0.1
Please enter the 4 input
The signal number:              4
The probability:                0.05
Please enter the 5 input
The signal number:              5
The probability:                0.05
|---------------THE-SOURCE-CODE-----------------|
 Prob  CodeWord
 0.500 0
 0.300 1
 0.100 20
 0.050 21
 0.050 22
|----------------MEASURES-----------------------|
 The expected Length :    1.20
 The variance :           0.16
```