به نام خدا

پروژه تحلیل و طراحی الگوریتم

استاد : دکتر داریوش نجفی

افراد گروه :

امیرحسین داداش زاده - 97463199
امیرحسین صفری - 97463137

الگوریتم **Huffman**

در این الگوریتم از object هایی به نام nodes استفاده شده است که همان نود های الگوریتم هافمن هستند.

این الگوریتم object oriented پیاده سازی شده است و ابتدا نود ها را بر اساس تکرار با چک کردن نود های Edge و ایجاد parent node به آخرین نود رسیده سپس خروجی چاپ میشود.

```
43  class node:
44      def __init__(self, frequency, name, node_left=None, node_right=None):
45
46          self.node_left = node_left
47          self.node_right = node_right
48          self.frequency = frequency
49          self.name = name
50          self.direction = ''
51
```

مقدار دهی اولیه

```
2   def main():
3       #-----------------------------------------------
4       # intialization arrays
5       #-----------------------------------------------
6
7       chars = ['a', 'b', 'c', 'd', 'e', 'f']
8       frequency = [ 1, 6, 11, 8, 6, 4]
9       nodes = [] ##unused nodes
10
11      #input_file = open("input.txt", "r")
12      #for line in input_file:
13      #    elements = line.split()
14      #    frequency.append(elements)
15
16      for x in range(len(chars)):
17          nodes.append(node(frequency[x], chars[x]))
18
19      #print(nodes)
```

تابع چاپ نود (اگر edge باشند:)

```python
def printer(node, value=''):

    value_main = value + str(node.direction)  # directionman for current node

    if(node.node_left):     #if node is not edge
        printer(node.node_left, value_main)
    if(node.node_right):
        printer(node.node_right, value_main)

    if not(node.node_left or node.node_right): # if node is on edge
        print(f"|   {node.name}, {value_main}")


```

گشتن در نود ها:

```python
while len(nodes) > 1:

    nodes = sorted(nodes, key=lambda x: x.frequency)    # sort nodes so we can pick smallest nodes

    #print nodes
    node_left = nodes[0]
    node_left.direction = 0
    node_right = nodes[1]
    node_right.direction = 1

    newNode = node(node_left.frequency+node_right.frequency, node_left.name+node_right.name, node_left, node_right) #merging two nodes

    nodes.remove(node_left)     #removing nodes
    nodes.remove(node_right)    #
    nodes.append(newNode)

print("#####################################")
printer(nodes[0])
```

جواب نهایی:

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
[Running] python -u "d:\Uni\5\Algorithm project\haffman\haffman.py"
#########################################
|   e, 00
|   d, 01
|   c, 10
|   a, 1100
|   f, 1101
|   b, 111

[Done] exited with code=0 in 0.088 seconds
```