Neeraj Appai 5073

PAGE NO.

DATE: / /

Algorithm Practicar 09

Aim - Write a program to demonstrate Hoffman Coding

Description

1) What is Huffman coding
-) i)Huffman coding is also known as encoding. It is an algorithm to compress data., basically. It is a baseless data compression algo.
2) This idea is to assign variable lenght ~~and~~ code to input characters
3) The most frequent Character gets the smallest code & the least frequent character gets the biggest code
4) The bitsequence is assigned in such a way that the code assigned to one Character is not the prefix of the code assigned to any other character to prevent ambiguity while encoding

2) Algorithm:
There are mainly 2 parts
i) Built a Huffman ~~from~~ Tree from input characte
ii) Traverse the Huffman Tree and assign codes to characte

i) Input is a set of unique characters along with their frequencies
1. Create a leaf node for each unique character and build a min heap of all nodes
(Min - Heap - used as a priority queue The value of frequency is used to ~~compre~~, compare least

frequency means root node)

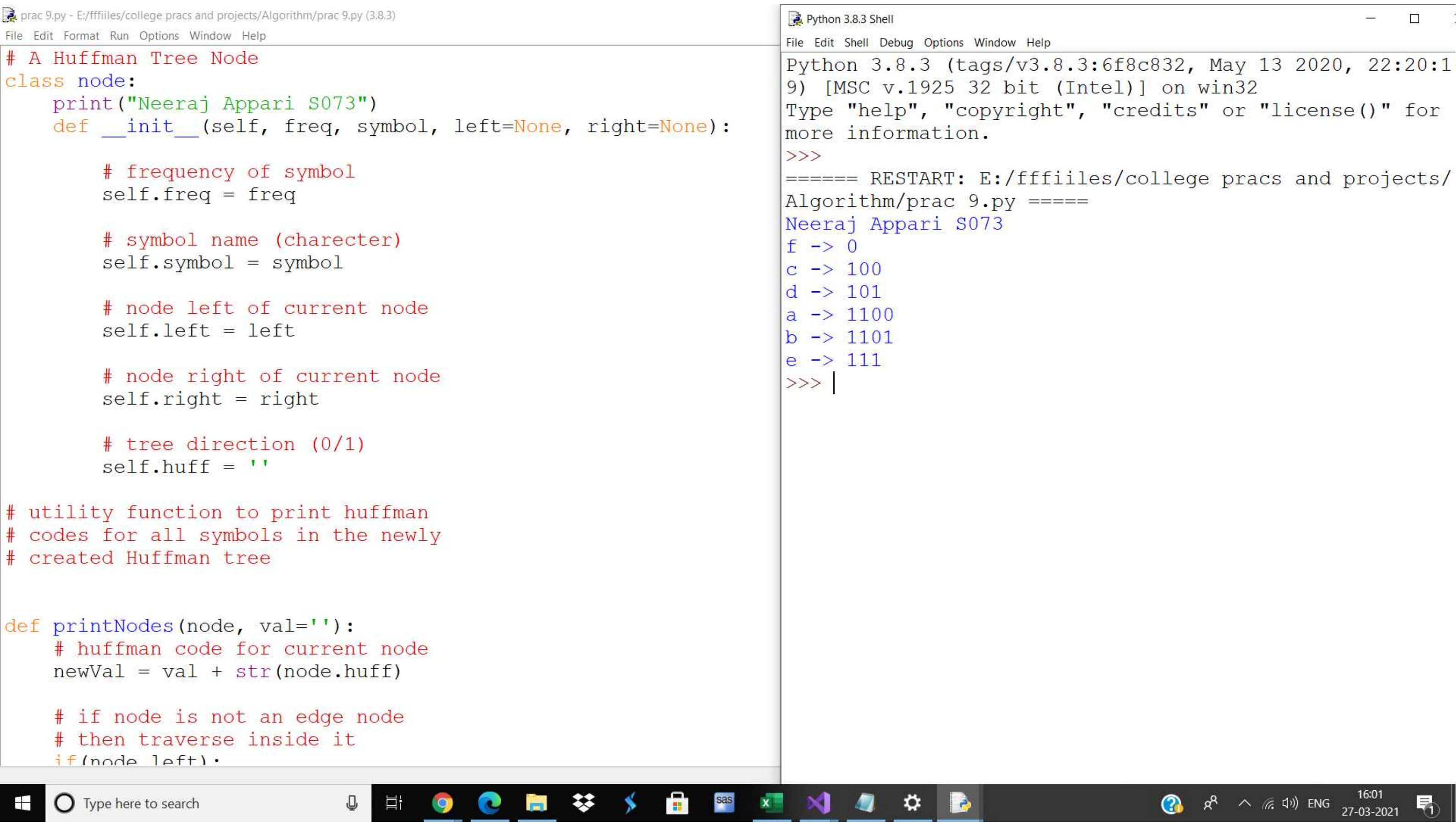
2. Extract 2 nodes with min freq from the min heap

3. Create a new internal node with a frequeny
equal to the sum of the two nodes frequency
Make the first extracted child as its left child
and other as right child
Add this node to min heap

4. Repeat step 2 and 3 until the heap contains only
one node. This remaining node is the root node and
~~at~~ the tree is complete

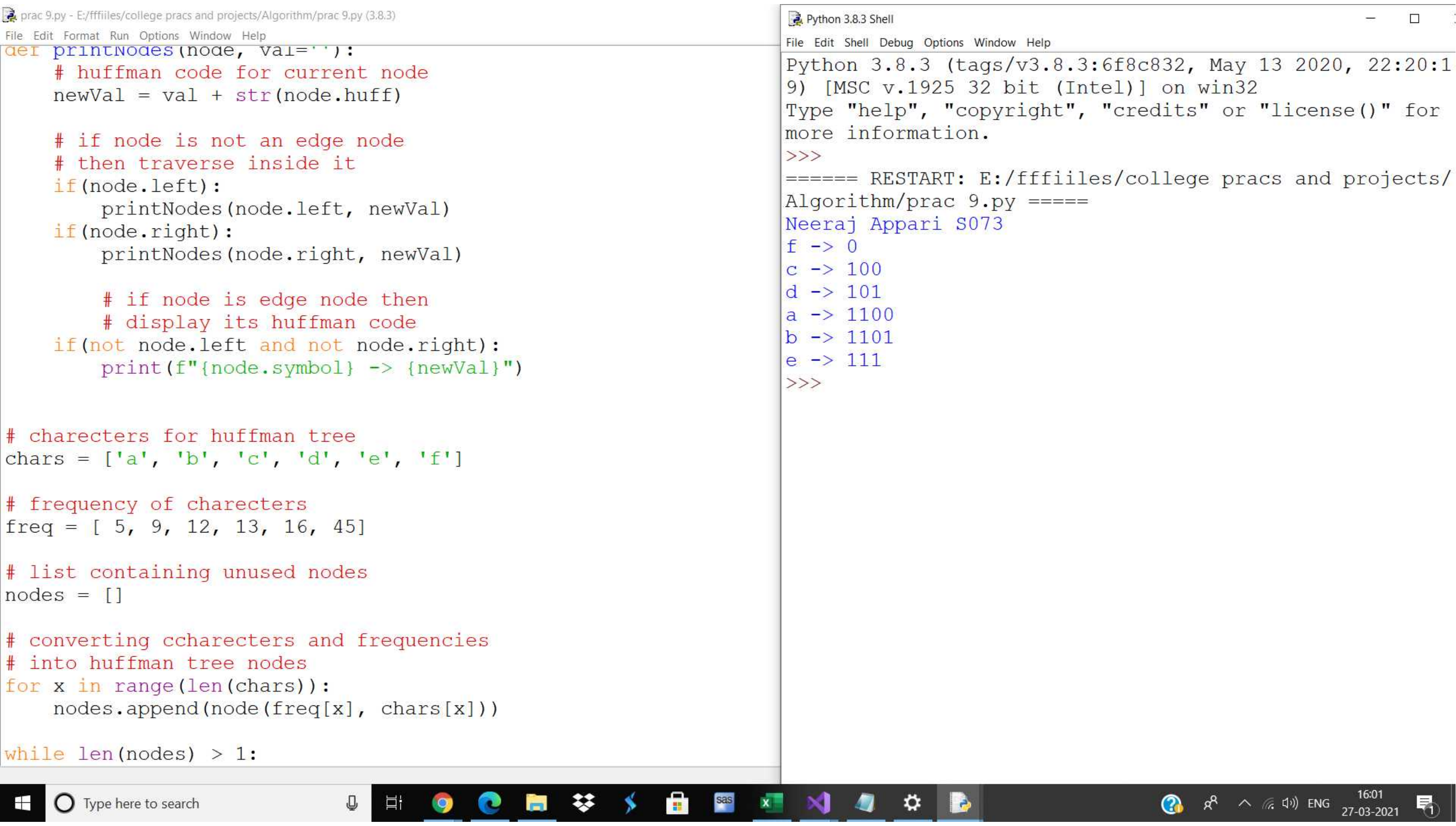
ii) Traverse the tree formed starting from the root
Maintain an auxillary array. While moving to left
child, write 0 to array while moving to
right child, write 1 to array print array when
leaf is encountered

With this we can generate codes for characters
And using them we can compress the data


prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)
File Edit Format Run Options Window Help
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Type here to search
ENG
16:01
27-03-2021

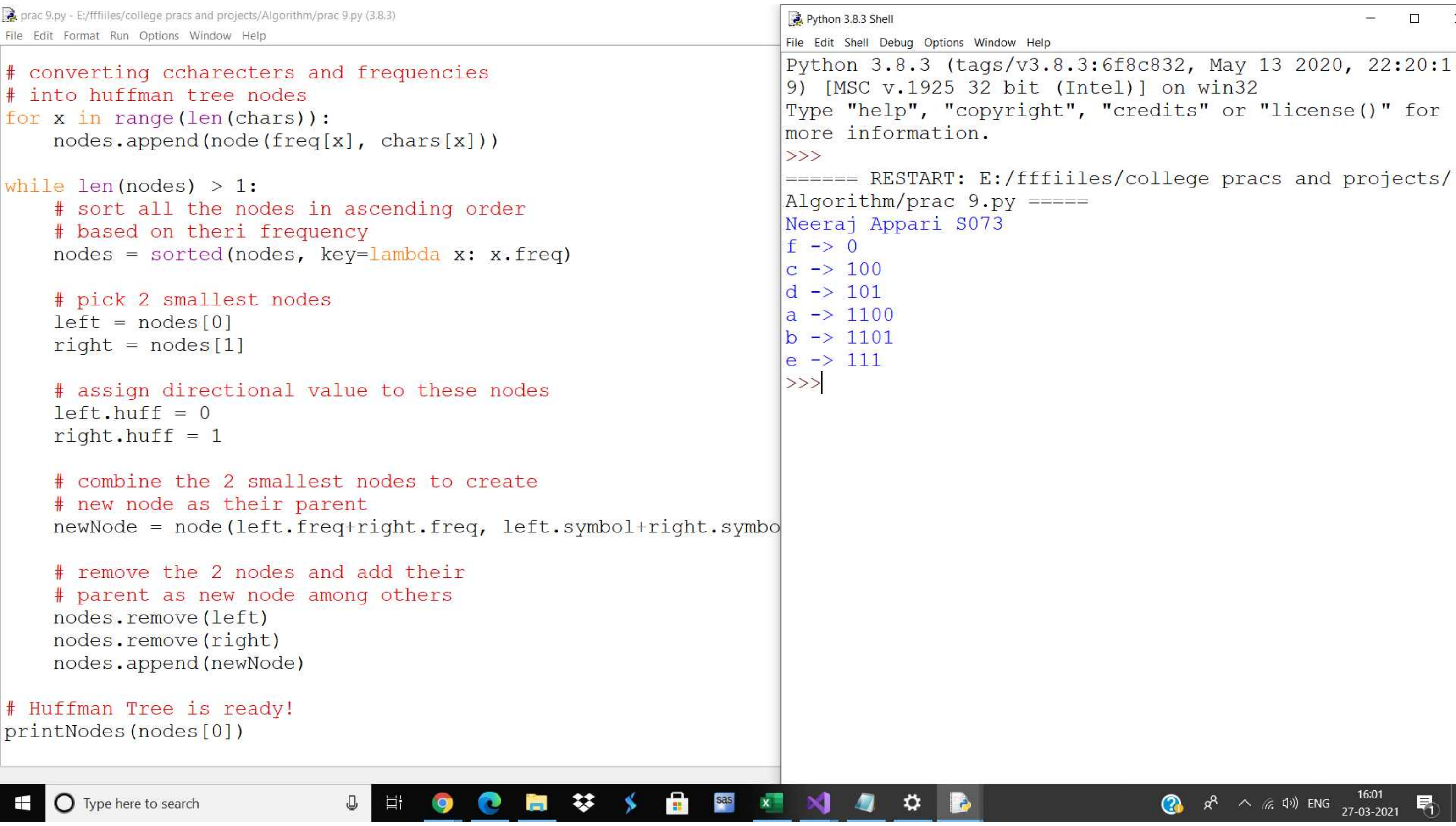
```python
# A Huffman Tree Node
class node:
    print("Neeraj Appari S073")
    def __init__(self, freq, symbol, left=None, right=None):

        # frequency of symbol
        self.freq = freq

        # symbol name (charecter)
        self.symbol = symbol

        # node left of current node
        self.left = left

        # node right of current node
        self.right = right

        # tree direction (0/1)
        self.huff = ''

# utility function to print huffman
# codes for all symbols in the newly
# created Huffman tree


def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node
    # then traverse inside it
    if(node.left):
```

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:1
9) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/
Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```


prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)
File Edit Format Run Options Window Help
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Type here to search
ENG
16:01
27-03-2021

```
def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node
    # then traverse inside it
    if(node.left):
        printNodes(node.left, newVal)
    if(node.right):
        printNodes(node.right, newVal)

        # if node is edge node then
        # display its huffman code
    if(not node.left and not node.right):
        print(f"{node.symbol} -> {newVal}")


# charecters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f']

# frequency of charecters
freq = [ 5, 9, 12, 13, 16, 45]

# list containing unused nodes
nodes = []

# converting ccharecters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
```

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:1
9) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/
Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```


prac 9.py - E:/fffiiles/college pracs and projects/Algorithm/prac 9.py (3.8.3)
File Edit Format Run Options Window Help
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Type here to search
ENG
16:01
27-03-2021

```
# converting ccharecters and frequencies
# into huffman tree nodes
for x in range(len(chars)):
    nodes.append(node(freq[x], chars[x]))

while len(nodes) > 1:
    # sort all the nodes in ascending order
    # based on theri frequency
    nodes = sorted(nodes, key=lambda x: x.freq)

    # pick 2 smallest nodes
    left = nodes[0]
    right = nodes[1]

    # assign directional value to these nodes
    left.huff = 0
    right.huff = 1

    # combine the 2 smallest nodes to create
    # new node as their parent
    newNode = node(left.freq+right.freq, left.symbol+right.symbo

    # remove the 2 nodes and add their
    # parent as new node among others
    nodes.remove(left)
    nodes.remove(right)
    nodes.append(newNode)

# Huffman Tree is ready!
printNodes(nodes[0])
```

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:1
9) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for
more information.
>>>
====== RESTART: E:/fffiiles/college pracs and projects/
Algorithm/prac 9.py =====
Neeraj Appari S073
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
>>>
```