Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 51
AAD Practical 13

Institute of Computer Technology
B. Tech Computer Science and Engineering

Sub: Algorithm Analysis and Design

Practical 13

Problem : Huffman coding assigns variable length code words to fixed length input characters based on their frequencies. More frequent characters are assigned shorter code words and less frequent characters are assigned longer code words. All edges along the path to a character contain a code digit. If they are on the left side of the tree, they will be a 0 (zero). If on the right, they'll be a 1 (one). Only the leaves will contain a letter and its frequency count. All other nodes will contain a null instead of a character, and the count of the frequency of all of it and its descendant characters.

1. Construct the Huffman tree for the following data Characters : A B C D E -
   Frequency/ Probability : 0.5 0.35 0.5 0.1 0.4 0.2

2. Which characters will become left children and which will become right children?

Code :

```
import YSL_io


def heapify(q, n, i):

root = i

l = 2*i + 1

r = 2*i + 2
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 51
AAD Practical 13

```
if (l < n) and (q[root] > q[l]):
root = l
if (r < n) and (q[root] > q[r]):
root = r
if root != i:
(q[i], q[root]) = (q[root], q[i])
heapify(q, n, root)
def heapSort(q):
n = len(q)
for i in range(n // 2 - 1, -1, -1):
heapify(q, n, i)
for i in range(n - 1, 0, -1):
(q[i], q[0]) = (q[0], q[i])
heapify(q, i, 0)
def insert(q,prob):
q += [prob]
heapSort(q)
class node:
def __init__(self, prob=None):
self.prob = prob
self.left = None
self.right = None
def huffman(c):
n = len(c)
```

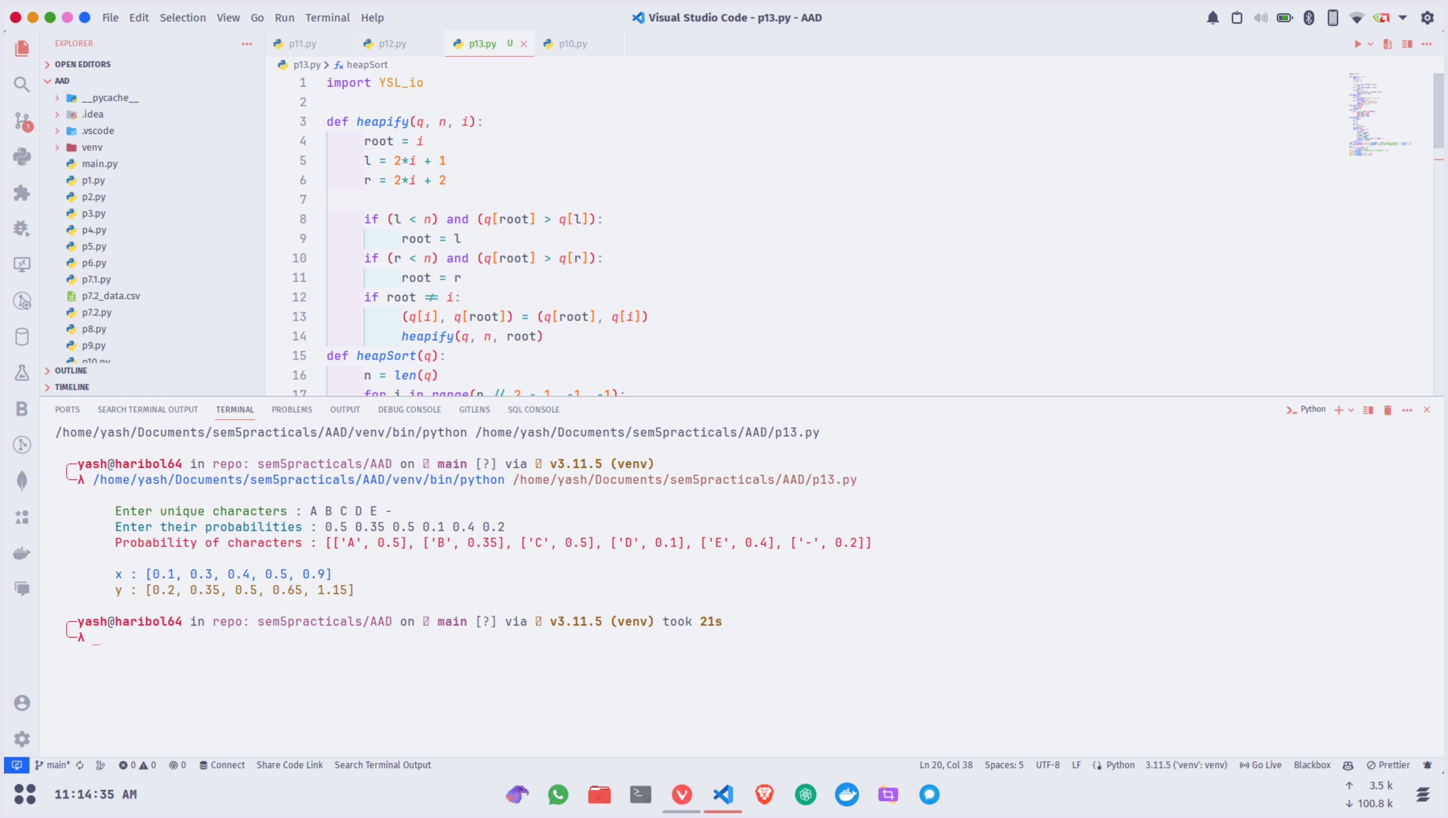
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 51
AAD Practical 13

```
q = []
x = []
y = []
for i in c:
q += [i[1]]
heapSort(q)
for i in range(1, n):
z = node()
z.left = q.pop()
x += [z.left]
z.right = q.pop()
y += [z.right]
z.prob = round(z.left + z.right, 3)
insert(q, z.prob)
return x,y
char = list(map(str, YSL_io.inputGRN("\n\tEnter unique characters :
").split(' ')))
f = list(map(float, YSL_io.inputCYN("\tEnter their probabilities :
").split(' ')))
c = []
for i, j in zip(char, f):
c += [list([i, j])]
YSL_io.printRED(f'\tProbability of characters : {c}')
x, y = huffman(c)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA   Batch - 51
AAD Practical 13

```python
YSL_io.printBLU(f"\n\tx : {x}")
YSL_io.printORNG(f"\ty : {y}")
```

<u>Screenshot :</u>



```python
import YSL_io

def heapify(q, n, i):
     root = i
     l = 2*i + 1
     r = 2*i + 2

     if (l < n) and (q[root] > q[l]):
          root = l
     if (r < n) and (q[root] > q[r]):
          root = r
     if root != i:
          (q[i], q[root]) = (q[root], q[i])
          heapify(q, n, root)
def heapSort(q):
     n = len(q)
     for i in range(n // 2 - 1, -1, -1):
```

```
/home/yash/Documents/sem5practicals/AAD/venv/bin/python /home/yash/Documents/sem5practicals/AAD/p13.py

 ┌yash@haribol64 in repo: sem5practicals/AAD on  main [?] via  v3.11.5 (venv)
 └λ /home/yash/Documents/sem5practicals/AAD/venv/bin/python /home/yash/Documents/sem5practicals/AAD/p13.py

        Enter unique characters : A B C D E -
        Enter their probabilities : 0.5 0.35 0.5 0.1 0.4 0.2
        Probability of characters : [['A', 0.5], ['B', 0.35], ['C', 0.5], ['D', 0.1], ['E', 0.4], ['-', 0.2]]

        x : [0.1, 0.3, 0.4, 0.5, 0.9]
        y : [0.2, 0.35, 0.5, 0.65, 1.15]

 ┌yash@haribol64 in repo: sem5practicals/AAD on  main [?] via  v3.11.5 (venv) took 21s
 └λ _
```