الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Department of Computer Science and Engineering

Submitted By:

Student Id:	C181208
Name:	Sameha Hasan
Section:	8AF
Course Code:	CSE-4875
Course Title:	Pattern Recognition and Image Processing sessional
Email:	samehasan25@gmail.com

Submitted To:

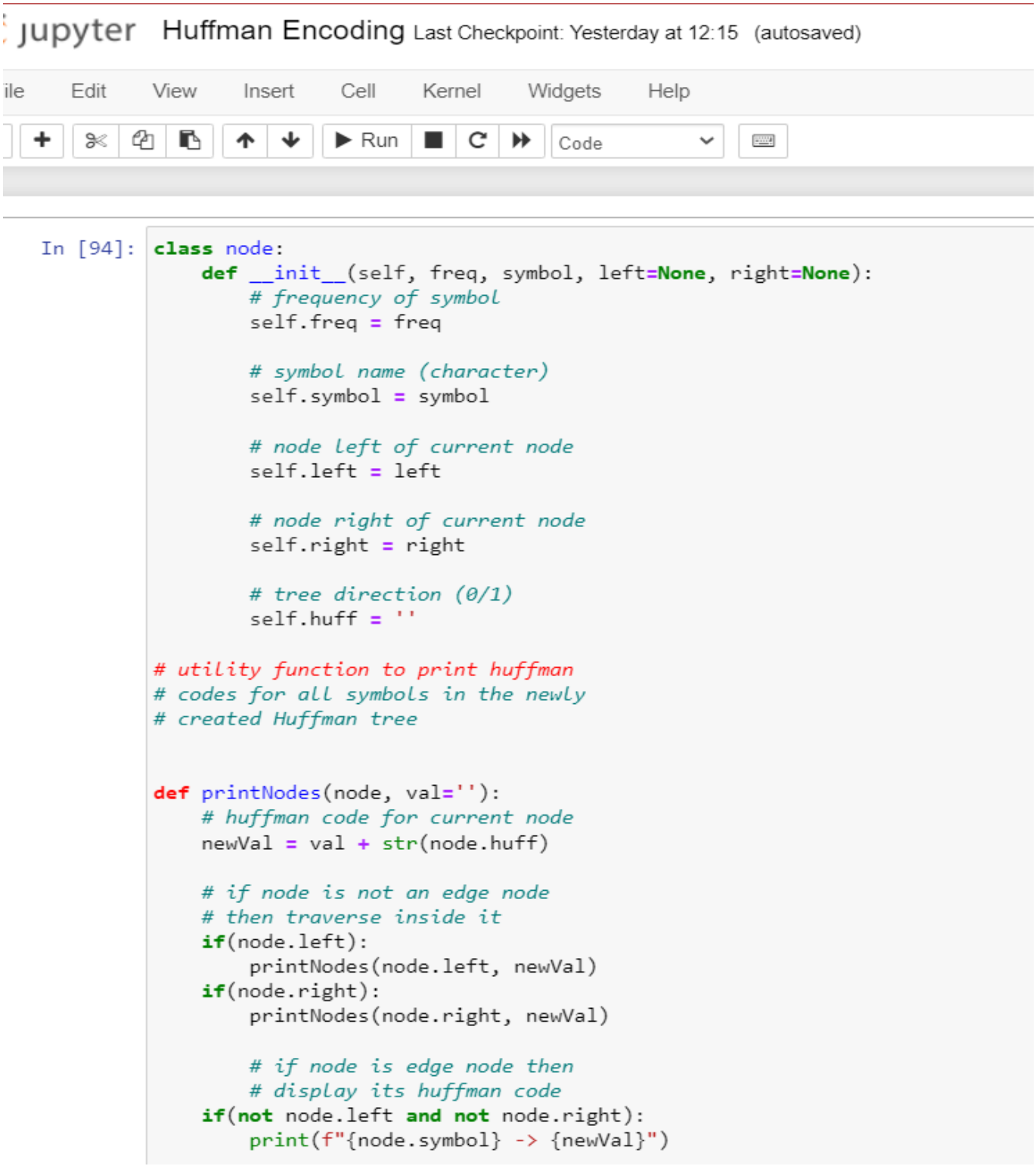
Mr. Mohammad Mahadi Hassan

Associate Professor,

Dept. of CSE , IIUC.

Lab 10

- Huffman coding
- Arithmetic coding
- RLE coding



ıpyter Huffman Encoding Last Checkpoint: Yesterday at 12:15 (unsaved changes)

Edit View Insert Cell Kernel Widgets Help

Run Code

```
# characters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f']
# frequency of characters
freq = [ 5, 9, 12, 13, 16, 45]
# list containing unused nodes
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.symbol, left, right)

    # 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])
```

```
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
```

jupyter Arithmetic coding Last Checkpoint: Yesterday at 12:15 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [2]:
```
def get_dict_from_signal():
    signal_dict = {}
    signal_dict['A'] = (0, 0.2)
    signal_dict['S'] = (0.2, 0.4)
    signal_dict['M'] = (0.4, 0.6)
    signal_dict['E'] = (0.6, 0.8)
    signal_dict['H'] = (0.8, 1)
    return signal_dict
def encoder(signal, signal_dict):
    Low = 0
    High = 1
    for s in signal:
        CodeRange = High - Low
        High = Low + CodeRange * signal_dict[s][1]
        Low = Low + CodeRange * signal_dict[s][0]
    return Low
def decoder(encoded_number, signal_dict, signal_length):
    signal = []
    while signal_length:
        for k, v in signal_dict.items():
            if v[0] <= encoded_number < v[1]:
                signal.append(k)
                range = v[1] - v[0]
                encoded_number -= v[0]
                encoded_number /= range
                break
        signal_length -= 1
    return signal
def main():
    signal_dict = get_dict_from_signal()
    signal = 'SAMEHA'
    ans = encoder(signal, signal_dict)
    print(ans)
    signal_rec = decoder(ans, signal_dict, len(signal))
    print(signal_rec)

main()
```

```
0.22208000000000003
['S', 'A', 'M', 'E', 'H', 'A']
```

jupyter Run Length Encoding Last Checkpoint: Yesterday at 12:16 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:

```
def encode_message(message):
    encoded_string = ""
    i = 0
    while (i <= len(message)-1):
        count = 1
        ch = message[i]
        j = i
        while (j < len(message)-1):
            if (message[j] == message[j + 1]):
                count = count + 1
                j = j + 1
            else:
                break
        encoded_string = encoded_string + str(count) + ch
        i = j + 1
    return encoded_string
def decode_message(our_message):
    decoded_message = ""
    i = 0
    j = 0
 # splitting the encoded message into respective counts
    while (i <= len(our_message) - 1):
        run_count = int(our_message[i])
        run_word = our_message[i + 1]
 # displaying the character multiple times specified by the count
        for j in range(run_count):
 # concatenated with the decoded message
            decoded_message = decoded_message+run_word
            j = j + 1
        i = i + 2
    return decoded_message
def display():
 # the original string
    our_message = "Sameha C181208"
 # pass in the original string
    encoded_message=encode_message(our_message)
 # pass in the decoded string
    decoded_message=decode_message(encoded_message)
    print("Original string: [" + our_message + "]")
    print("Encoded string: [" + encoded_message +"]")
    print("Decoded string: [" + decoded_message +"]")
display()
```

Output:

```
Original string: [Sameha C181208]
Encoded string: [1S1a1m1e1h1a1 1C111811121018]
Decoded string: [Sameha C181208]
```