Algorithm-10

—— Huffman-Codes

A. Problem Description

Huffman coding is based on the frequency of occurance of a data item (pixel in images). The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a ***Code Book*** which may be constructed for each image or a set of images. In all cases the code book plus encoded data must be transmitted to enable decoding.

B. Description of algorithm

```
HUFFMAN(C)

    n = |C|

    INITIALIZE Q = BUILD-MIN-HEAP(C)

    for i=1 to n-1

      allocate a new node z

      z.left = x ← EXTRACT-MIN(Q)

      z.right = y ← EXTRACT-MIN(Q)

      z.freq ← x.freq + y.freq

      INSERT(Q, z)

  return EXTRACT-MIN(Q)
```

C. Time Complexity

Step "**INITIALIZE** Q = **BUILD-MIN-HEAP**(*C*)" ➔ *O*(n)

Step "for" ➔ *O*(n)

Step "**EXTRACT-MIN**(Q)" ➔ *O*(nlogn)

Step "**INSERT**(Q, z)" ➔ *O*(nlogn)

Therefore, the total time is

T = *O*(nlogn)

D. Code[Python]

```
#!/usr/bin/python
# Filename: Huffman_Codes.py

class node:
  def __init__(self, flag = '', value = -1, left=-1, right = -1, prefix =
-1):
    self.flag = flag
    self.value = value
    self.left = left
    self.right = right
    self.prefix = prefix

def Huffman_Codes(array):
  low = 0
  high = (len(array) - 1) / 2

  while low < high:
    sort(array, low, high)
    high += 1
    array[high].value = array[low].value + array[low + 1].value
    array[high].left = low
    array[high].right = low + 1
    array[low].prefix = 0
    array[low + 1].prefix = 1
    low += 2

def Traceback(array, index, a):
  if array[index].left != -1:
    a.append(0)
    Traceback(array, array[index].left, a)
    a.pop()
```

```
        a.append(1)
        Traceback(array, array[index].right, a)
        a.pop()
      else:
            print str(a) + ' ==> ' + array[index].flag + '(' +
str(array[index].value) + ')'

    def sort(array, a, b):
      for i in range(a, b):
        k = i
        for j in range(i + 1, b + 1):
          if array[j].value < array[k].value:
            k = j
        if k != i:
          temp = node()
          temp = array[i]
          array[i] = array[k]
          array[k] = temp
```