In [5]:
```
# Title :- Implement Huffman Encoding using a greedy strategy.
```

```
In [8]: import heapq

        class node:

         def __init__(self,freq,symbol,left=None,right=None):

             self.freq=freq
             self.symbol=symbol
             self.left=left
             self.right=right
             self.huff= ''

         def __lt__(self,nxt):
             return self.freq < nxt.freq


        def printnodes(node,val=''):

             newval=val+str(node.huff)

             if node.left:
                 printnodes(node.left,newval)

             if node.right:
                 printnodes(node.right,newval)


             if not node.left and not node.right:
                 print(f"{node.symbol} -> {newval}")

        if __name__=="__main__":

             chars = ['a', 'b', 'c', 'd', 'e', 'f']
             freq = [ 5, 9, 12, 13, 16, 45]
             nodes=[]


             for i in range(len(chars)):
                 heapq.heappush(nodes, node(freq[i],chars[i]))

             while len(nodes)>1:

                 left=heapq.heappop(nodes)
                 right=heapq.heappop(nodes)
                 left.huff = 0
                 right.huff = 1


                 newnode = node(left.freq + right.freq , left.symbol + right.symbol ,


                 heapq.heappush(nodes, newnode)
```

```
printnodes(nodes[0])
```

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