Telekomunikacja - laboratorium				Studia zaoczne - inzynierskie	
Nazwa zadania	**Implementacja algorytmu kodowania Huffmana**				
Dzień	**Poniedziałek**	**Godzina**	**14:00**	**Rok akademicki**	**2020/2021**
Imię i Nazwisko	**Wiktor Bechciński, 229840**				
Imię i Nazwisko	**Kamil Budzyn, 229850**				

Opis programu, rozwiązania problemu.

Celem zadania było zaprojektowanie programu pozwalającego na przesłanie zakodowanego pliku tekstowego za pomocą statycznego algorytmu Huffmana poprzez metodę gniazd sieciowych. W celu weryfikacji rozwiązania problemu stworzono dwa programy, jeden szyfrujący wiadomość i wysyłający ją poprzez gniazdo sieciowe a drugi odbierający wiadomość poprzez gniazdo oraz odszyfrowujący ją. Algorytm pozwala na zmniejszenie ilości transferowanych bitów pomiędzy programami.

Najważniejsze elementy kodu programu z opisem.

```
//Utworzenie gniazda otwartego do nasłuchu
serverSocket = socket.socket()
ip = "127.0.0.1"
port = 3554
serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
serverSocket.bind((ip, port))
print("Gniazdo ustawiono na adres", ip, ":", port)

serverSocket.listen()

//Utworzenie gniazda wysyłającego wiadomość
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = "127.0.0.1"
port = 3554
clientSocket.connect((ip, port))
print("Gniazdo polaczono z adresem", ip, ":", port)

//Deklaracja węzłów
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 = ''

//Stworzenie tablic zawierjących litery oraz ich zakodowaną postać

def createNodes(node, arr1, arr2, val=''):
    newVal = val + str(node.huff)
    if node.left:
        createNodes(node.left, arr1,arr2, newVal)
    if node.right:
        createNodes(node.right, arr1,arr2, newVal)
    if not node.left and not node.right:
        arr1.append(node.symbol)
        arr2.append(newVal)

//Zakodowanie wiadomości
def coding(msg):
    nodes = Huffman()
    letters = []
    code = []
    createNodes(nodes[0], letters, code)
    output = ''
    for i in msg:
        for j in range(len(letters)):
            if i == letters[j]:
                output = output + code[j]
    return output

//Odkodowanie wiadomości
def decoding(msg):
    nodes = Huffman()
    current_node = nodes[0]
    output = ''
    for i in msg:
        if int(i) == 0:
```

```
            current_node = current_node.left
        else:
            current_node = current_node.right
        if current_node.left is None and current_node.right is None:
            output += current_node.symbol
            current_node = nodes[0]
    return output

//Utworzenie drzewa Huffmana
def Huffman():
    chars = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
             'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't',
             'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D',
             'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
             'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
             'Y', 'Z', '-', '_', '/', "'", '"', ':', ';', ')',
             '(', '!', '.', '?', ',', chr(10), ' ', '1', '2', '3',
             '4', '5', '6', '7', '8', '9', '0', '=', '@', '*']

    freq = [87, 10, 29, 47, 133, 25, 37, 84, 43, 1, 5, 55, 15, 97, 107, 24, 2, 73, 53, 85, 49, 8, 14, 1, 32, 0, 0, 0, 5, 0, 0, 1,
0, 10, 1, 0, 8, 0, 3, 0, 6, 6, 0, 1, 1, 19, 0, 0, 2, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 1, 10, 24, 0, 39, 0, 265, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0]
    nodes = []
    for x in range(len(chars)):
        nodes.append(Node(freq[x], chars[x]))

    while len(nodes) > 1:
        nodes = sorted(nodes, key=lambda x: x.freq)
        left = nodes[0]
        right = nodes[1]
        left.huff = 0
        right.huff = 1
        newNode = Node(left.freq + right.freq, left.symbol + right.symbol, left, right)
        nodes.remove(left)
        nodes.remove(right)
        nodes.append(newNode)
    return nodes
```

Podsumowanie wnioski.

Program spełnia przedstawione w zadaniu wymagania. Jest w stanie nadać przez gniazdo sieciowe zaszyfrowany plik algorytmem Huffmana oraz następnie odebrac go i odszyfrować.