A Lab Mini Project Report

On

**IMPLEMENTATION OF CHAT BOX**

**USING HUFFMAN ALGORITHM**

Semester: IV

Section: CSE-A

**COURSE:** Design and Analysis of Algorithms Lab

**BY**

**T. VYSHNAVI LAHARI**

**(1602-18-733-061)**

**T. HEMANTH REDDY**

**(1602-18-733-022)**



UNDER THE GUIDANCE OF

**V. Punna Rao**

**ASSISTANT PROFESSOR**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VASAVI COLLEGE OF ENGINEERING(AUTONOMOUS)**

**AFFLIATED TO OSMANIA UNIVERSITY**

**IBRAHIMBAGH**

**HYDERABAD-500031**

**ACKNOWLEDGEMENT**

*We would like to express our heartfelt gratitude to V. Punna Rao, our project guide, for his valuable guidance and constant support, along with his capable instructions and persistent encouragement.*

*We are grateful to our Head of Department, Dr. T. Adilakshmi, for her steady support and the provision of every resource required for the completion of this project.*

*We would like to take this opportunity to thank our Principal, Dr. S. V. Ramana, as well as the management of the institute, for having designed an excellent learning atmosphere.*

*We are thankful to and fortunate enough to get constant encouragement, support and guidance which helped us in successfully completing our project work.*

**Design and Analysis of Algorithms Lab**

**Mini Project**

**on**

**IMPLEMENTATION OF CHAT BOX**

**USING HUFFMAN ALGORITHM**

**ABSTRACT**

In this project, we will be implementing chat box in java. For this, we will be using socket programming and Huffman algorithm to encode the message. To ensure safety and to increase usability we will be using Huffman algorithm.

We will be implementing Huffman through Greedy approach to reduce Time Complexity. In this project, message will be encoded at the sender and decoded at the receiver. Using the basic knowledge of networking in java, we will be implementing “Chat Box” by establishing client server environment.

**TABLE OF CONTENTS**

1. Introduction 1

1.1 Huffman

1.2 Java Socket Programming

2. System Design 2

3. System Analysis 6

4. Code 7

5. Output 26

6. Conclusion and Future Scope 27

7. References 28

**LIST OF FIGURES**

Fig 2.1: String to be sent 2

Fig 2.2: Frequencies 3

Fig 2.3: Sorted Frequencies 3

Fig 2.4: Huffman tree(Level 1) 4

Fig 2.5: Huffman tree(Level 2) 4

Fig 2.6: Huffman tree(Level 3) 4

Fig 2.7: Huffman tree(Level 4) 5

Fig 5.1: Output 1 26

Fig 5.2: Output 2 26

1. **INTRODUCTION**

**Huffman**

Huffman Coding is a technique of compressing data so as to reduce its size without losing any of the details. It was first developed by David Huffman. Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.

Codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream.  
Let there be four characters a, b, c and d, and their corresponding variable length codes be 00, 01, 0 and 1. This coding leads to ambiguity because code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be “cccd” or “ccb” or “acd” or “ab”. Hence we use Huffman algorithm to avoid this ambiguity.

**Java Socket Programming**

Networking is a major branch of programming that is vital to connecting users through devices. As such many programming languages have multiple ways to form connections users and servers or between peers. Java is one of the first languages many programmers learn, and one of the interesting ways java can handle network connections is through the use of Java Sockets.

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other’s network location (IP Address) and TCP port. The java.net.Socket class represents a Socket.

Socket socket = new Socket(“127.0.0.1”, 5000)

* First argument – **IP address of Server**. ( 127.0.0.1  is the IP address of localhost, where code will run on single stand-alone machine).
* Second argument – **TCP Port**. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

1. **SYSTEM DESIGN**

In our project, we are using Huffman algorithm for encoding messages.

There are mainly two major parts in Huffman Coding  
**1)**  Build a Huffman Tree from input characters.  
**2)**  Traverse the Huffman Tree and assign codes to characters.

**Steps to build Huffman Tree**Input is an array of unique characters along with their frequency of occurrences and output is Huffman Tree.

**1)**  Create a leaf node for each unique character and build a min heap of all leaf nodes (Min Heap is used as a priority queue. The value of frequency field is used to compare two nodes in min heap. Initially, the least frequent character is at root)

**2)** Extract two nodes with the minimum frequency from the min heap.

**3)**  Create a new internal node with a frequency equal to the sum of the two nodes frequencies. Make the first extracted node as its left child and the other extracted node as its right child. Add this node to the min heap.

**4)**  Repeat steps#2 and #3 until the heap contains only one node. The remaining node is the root node and the tree is complete.

Suppose the string below is to be sent over a network.

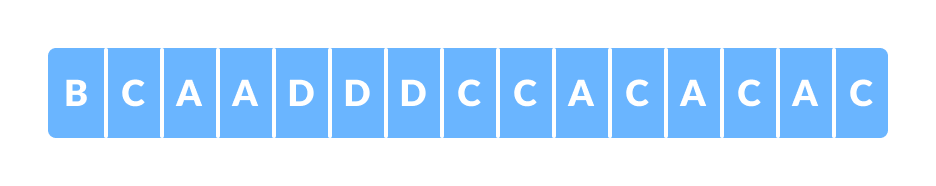


Fig 2.1 String to be sent

Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of 8 \* 15 = 120 bits are required to send this string.

Using the Huffman Coding technique, we can compress the string to a smaller size.

Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.

Once the data is encoded, it has to be decoded. Decoding is done using the same tree.

Huffman Coding prevents any ambiguity in the decoding process using the concept of **prefix code** ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.

Huffman coding is done with the help of the following steps.

1. Calculate the frequency of each character in the string.

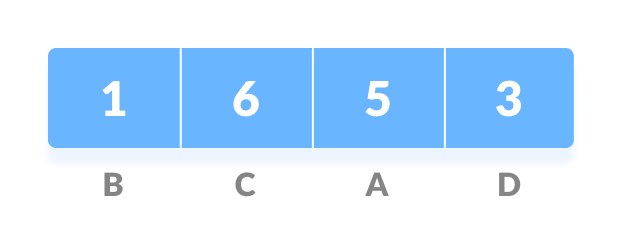


Fig 2.2 Frequencies

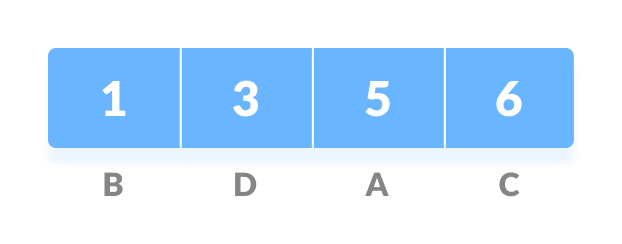
1. Sort the characters in increasing order of the frequency. These are stored in a priority queue   
    

Fig 2.3 Sorted Frequencies

1. Make each unique character as a leaf node.
2. Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies

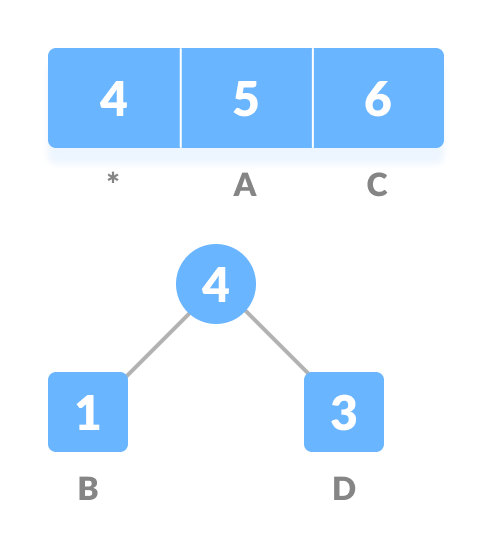


Fig 2.4 Huffman tree(Level 1)

1. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (\* denote the internal nodes in the figure above).
2. Insert node z into the tree.
3. Repeat steps 3 to 5 for all the characters.

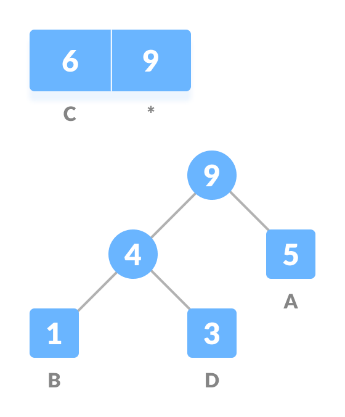


Fig 2.5 Huffman tree(Level 2)

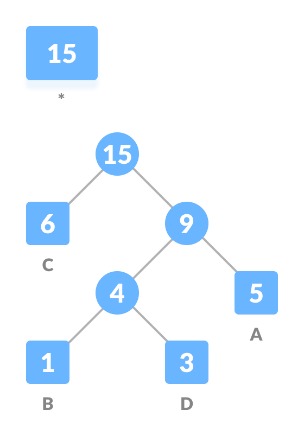


Fig 2.6 Huffman tree(Level 3)

8) For each non-leaf node, assign 0 to the left edge and 1 to the right edge.

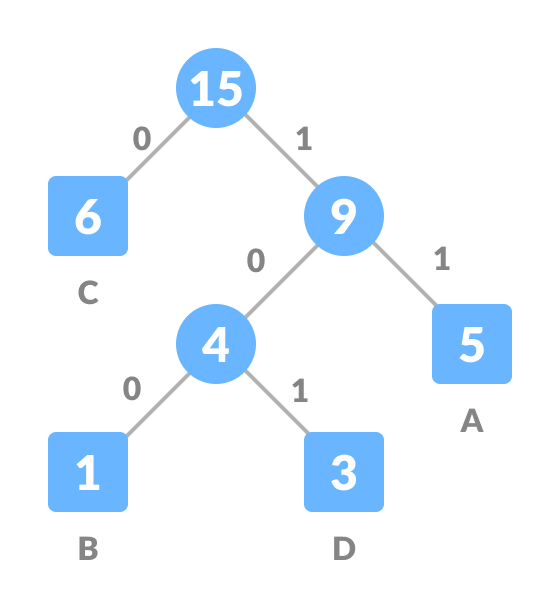


Fig 2.7 Huffman tree(Level 4)

1. **SYSTEM ANALYSIS**

Considering we have n characters to be coded,

* A divide-and-conquer approach might have us asking which characters should appear in the left and right subtrees and trying to build the tree from the top down. As with the optimal binary search tree, this will lead to to an exponential time algorithm.
* A greedy approach places our **n** characters in **n** sub-trees and starts by combining the two least weight nodes into a tree which is assigned the sum of the two leaf node weights as the weight for its root node.

**Time complexity of Huffman using Greedy approach:**

The time complexity of the Huffman algorithm is **O(nlogn)**. Using a heap to store the weight of each tree, each iteration requires **O(logn)** time to determine the cheapest weight and insert the new weight. There are **O(n)** iterations, one for each item.

1. **CODE**

**Package : server**

**Class : serverclass.java**

package server;

import java.awt.Color;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JButton;

import javax.swing.JLabel;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.HashMap;

import java.util.PriorityQueue;

import javax.swing.JTextArea;

import server.node;

import server.sortbyfreq;

public class serverclass {

private static JFrame frame;

private static JTextArea textArea;

private static JTextArea textPane;

static ServerSocket ss;

static Socket s;

static DataInputStream dis;

static DataOutputStream dos;

static String msgin="";

private static HashMap<String,String> enm = new HashMap<String,String>();//for storing ch-1010110

public static String comsg="";

private static void printcodes(node root, String str)throws Exception {

if(root.left == null && root.right == null )

{

String da=""+root.data;

enm.put(da,str);

comsg=comsg+str+":"+da+"$";

return ;

}

printcodes(root.left,str+"0");

printcodes(root.right,str+"1");

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

serverclass window = new serverclass();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

try {

ss = new ServerSocket(2261);

s = ss.accept();

dos = new DataOutputStream(s.getOutputStream());

dis = new DataInputStream(s.getInputStream());

String hemmsg="";

String demmsg="";

int i,j;

String dnm="";

String decodemessage="";

while(msgin!="$$$$")

{

HashMap<String,String> ednm = new HashMap<String,String>();//for storing 1010110-ch

msgin = dis.readUTF();

if(msgin.equals("$$$$"))

frame.setVisible(false);

for(i=0;i<msgin.length();i++)

{

if(msgin.charAt(i)=='@')

{

i++;

break;

}

hemmsg=hemmsg+msgin.charAt(i);

}

for(j=i;j<msgin.length();j++)

{

if(msgin.charAt(j)==':')

{

j++;

ednm.put(demmsg,""+msgin.charAt(j));

demmsg="";

j=j+2;

if(j==msgin.length())

{

break;

}

}

demmsg=demmsg+msgin.charAt(j);

}

for(i=0;i<hemmsg.length();i++)

{

dnm=dnm+hemmsg.charAt(i);

if(ednm.containsKey(dnm))

{

decodemessage=decodemessage+ednm.get(dnm);

dnm="";

}

}

textArea.setText(decodemessage);

textPane.setText(textPane.getText()+"\n"+"rev : "+hemmsg);

hemmsg="";

decodemessage="";

dnm="";

ednm=null;

}

} catch (Exception e) {

}

}

public serverclass() {

initialize();

}

private void initialize() {

frame = new JFrame();

frame.setBounds(100, 100, 600, 520);

frame.getContentPane().setBackground(Color.DARK\_GRAY);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

textPane = new JTextArea();//am

textPane.setBounds(12, 41, 545, 197);

frame.getContentPane().add(textPane);

JTextArea textPane\_1 = new JTextArea();//tm

textPane\_1.setBounds(12, 399, 452, 61);

frame.getContentPane().add(textPane\_1);

JButton btnSend = new JButton("Send");

btnSend.setBounds(473, 410, 97, 25);

frame.getContentPane().add(btnSend);

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

String message =textPane\_1.getText();

String encodemessage="";

int[] fre = new int[message.length()];

int i,j,n=0;

char str[] = message.toCharArray();

for(i=0;i<message.length();i++)

{

fre[i]=1;

for(j=i+1;j<message.length();j++)

{

if(str[i]==str[j])

{

fre[i]++;

str[j]='0';

}

}

}

for(i=0;i<fre.length;i++)

{

if( str[i]!='0')

{

n=n+1;

}

}

PriorityQueue<node> msg = new PriorityQueue<node>(n,new sortbyfreq());

for(i=0;i<fre.length;i++)

{

if( str[i]!='0')

{

node h=new node(fre[i],str[i]);

msg.add(h);

}

}

node root=null;//encoding

while(msg.size()>1)

{

node p = msg.peek();

msg.poll();//remove last

node q = msg.peek();

msg.poll();//remove second last

node temp = new node(p.frequency+q.frequency);

temp.left=p;

temp.right=q;

root=temp;

msg.add(temp);//add sum 2

}

comsg="";

printcodes(root,"");

for(i=0;i<message.length();i++)

{

String da=""+message.charAt(i);

encodemessage=encodemessage+enm.get(da);

}

textPane.setText(textPane.getText()+"\n"+"sen : "+encodemessage);

encodemessage=encodemessage+"@"+comsg;

dos.writeUTF(encodemessage);

dos.flush();

textPane\_1.setText(null);

} catch (Exception e1) {

}

}

});

JLabel lblTypeMessage = new JLabel("Type Message");

lblTypeMessage.setForeground(Color.WHITE);

lblTypeMessage.setFont(new Font("Tempus Sans ITC", Font.BOLD | Font.ITALIC, 20));

lblTypeMessage.setBounds(12, 361, 152, 25);

frame.getContentPane().add(lblTypeMessage);

JLabel lblBeforeMessages = new JLabel("Before Message");

lblBeforeMessages.setForeground(Color.white);

lblBeforeMessages.setFont(new Font("Tahoma", Font.BOLD, 13));

lblBeforeMessages.setBounds(12, 12, 112, 16);

frame.getContentPane().add(lblBeforeMessages);

textArea = new JTextArea();//bm

textArea.setBounds(12, 305, 545, 38);

frame.getContentPane().add(textArea);

JLabel lblRecivedMessage = new JLabel("Received Message");

lblRecivedMessage.setForeground(Color.white);

lblRecivedMessage.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 18));

lblRecivedMessage.setBounds(12, 262, 184, 30);

frame.getContentPane().add(lblRecivedMessage);

}

}

**Package : server**

**Class : node.java**

package server;

public class node {

public int frequency;

public char data;

public node left=null;

public node right=null;

public node(int f,char ch) {

frequency = f;

data = ch;

}

public node(int f)

{

frequency = f;

data = '$';

}

}

**Package : server**

**Class : sortbyfrequency.java**

package server;

import java.util.Comparator;

import server.node;

public class sortbyfreq implements Comparator <node>{

public int compare(node a, node b) {

return a.frequency-b.frequency;

}

}

**Package : client**

**Class : clientclass.java**

package client;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JLabel;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.net.Socket;

import java.util.HashMap;

import java.util.PriorityQueue;

import javax.swing.JTextField;

import server.node;

import server.sortbyfreq;

import javax.swing.JButton;

import javax.swing.JTextArea;

import java.awt.Color;

public class clientclass {

private JFrame frame;

private static JTextField textField;

private static JTextField textField\_1;

private static JTextArea textArea\_1;

private static JTextArea textArea;

static Socket S;

static DataInputStream dis;

static DataOutputStream dos;

static String msgin="";

private static HashMap<String,String> enm = new HashMap<String,String>();//for storing ch-1010110

static String comsg="";

private static void printcodes(node root, String str)throws Exception {

if(root.left == null && root.right == null )

{

String da=""+root.data;

enm.put(da,str);

//ednm.put(str, da);

comsg=comsg+str+":"+da+"$";

return ;

}

printcodes(root.left,str+"0");

printcodes(root.right,str+"1");

}

public static void main(String[] args) throws Exception

{

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

clientclass window = new clientclass();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

try {

S = new Socket("localhost",2261);//(textField.getText(),port);

dos = new DataOutputStream(S.getOutputStream());

dis = new DataInputStream(S.getInputStream());

String hemmsg="";

String demmsg="";

int i,j;

String dnm="";

String decodemessage="";

while(msgin!="$$$$")

{

HashMap<String,String> ednm = new HashMap<String,String>();//for storing 1010110-ch

msgin = dis.readUTF();

for(i=0;i<msgin.length();i++)

{

if(msgin.charAt(i)=='@')

{

i++;

break;

}

hemmsg=hemmsg+msgin.charAt(i);

}

textArea.setText(textArea.getText()+"\n"+"rev : "+hemmsg);

for(j=i;j<msgin.length();j++)

{

if(msgin.charAt(j)==':')

{

j++;

ednm.put(demmsg,""+msgin.charAt(j));

demmsg="";

j=j+2;

if(j==msgin.length())

{

break;

}

}

demmsg=demmsg+msgin.charAt(j);

}

for(i=0;i<hemmsg.length();i++)

{

dnm=dnm+hemmsg.charAt(i);

if(ednm.containsKey(dnm))

{

decodemessage=decodemessage+ednm.get(dnm);

dnm="";

}

}

textArea\_1.setText(decodemessage);

hemmsg="";

decodemessage="";

dnm="";

}

} catch (Exception e1) {

//not handle

}

}

public clientclass() {

initialize();

}

private void initialize() {

frame = new JFrame();

frame.getContentPane().setBackground(Color.DARK\_GRAY);

frame.setBounds(100, 100, 600, 550);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

JLabel lblIpAddress = new JLabel("IP Address");

lblIpAddress.setForeground(Color.white);

lblIpAddress.setFont(new Font("Tahoma", Font.BOLD, 18));

lblIpAddress.setBounds(23, 29, 109, 22);

frame.getContentPane().add(lblIpAddress);

JLabel lblPortNumber = new JLabel("Port Number");

lblPortNumber.setForeground(Color.white);

lblPortNumber.setFont(new Font("Tahoma", Font.BOLD, 18));

lblPortNumber.setBounds(229, 34, 131, 17);

frame.getContentPane().add(lblPortNumber);

textField = new JTextField();

textField.setBounds(12, 64, 173, 22);

frame.getContentPane().add(textField);

textField.setText("IP ADDRESS");

textField.setColumns(10);

textField\_1 = new JTextField();

textField\_1.setBounds(229, 64, 142, 22);

frame.getContentPane().add(textField\_1);

textField\_1.setText(" - $ - \* - $ -");

textField\_1.setColumns(10);

JButton btnDisconnect = new JButton("Disconnect");

btnDisconnect.setBounds(446, 75, 97, 25);

frame.getContentPane().add(btnDisconnect);

btnDisconnect.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

dos = new DataOutputStream(S.getOutputStream());

dos.writeUTF("$$$$");

dos.flush();

S.close();

frame.setVisible(false);

} catch (Exception e1) {

}

}

});

JLabel lblBeforeMessages = new JLabel("Before Message");

lblBeforeMessages.setForeground(Color.white);

lblBeforeMessages.setFont(new Font("Tahoma", Font.BOLD, 18));

lblBeforeMessages.setBounds(22, 111, 163, 22);

frame.getContentPane().add(lblBeforeMessages);

textArea = new JTextArea();

textArea.setBounds(23, 146, 530, 145);

frame.getContentPane().add(textArea);

JLabel lblRecivedMessage = new JLabel("Received Message");

lblRecivedMessage.setForeground(Color.white);

lblRecivedMessage.setFont(new Font("Tahoma", Font.BOLD, 18));

lblRecivedMessage.setBounds(23, 316, 190, 22);

frame.getContentPane().add(lblRecivedMessage);

textArea\_1 = new JTextArea();

textArea\_1.setBounds(23, 351, 530, 46);

frame.getContentPane().add(textArea\_1);

JLabel lblTypeMessage = new JLabel("Type Message");

lblTypeMessage.setForeground(Color.white);

lblTypeMessage.setFont(new Font("Tahoma", Font.BOLD, 18));

lblTypeMessage.setBounds(23, 410, 162, 22);

frame.getContentPane().add(lblTypeMessage);

JTextArea textArea\_2 = new JTextArea();

textArea\_2.setBounds(23, 445, 442, 45);

frame.getContentPane().add(textArea\_2);

JButton btnSend = new JButton("Send");

btnSend.setBounds(477, 444, 97, 25);

frame.getContentPane().add(btnSend);

btnSend.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

String message =textArea\_2.getText();

String encodemessage="";

int[] fre = new int[message.length()];

int i,j,n=0;

char str[] = message.toCharArray();

for(i=0;i<message.length();i++)

{

fre[i]=1;

for(j=i+1;j<message.length();j++)

{

if(str[i]==str[j])

{

fre[i]++;

str[j]='0';

}

}

}

for(i=0;i<fre.length;i++)

{

if( str[i]!='0')

{

n=n+1;

}

}

PriorityQueue<node> msg = new PriorityQueue<node>(n,new sortbyfreq());

for(i=0;i<fre.length;i++)

{

if( str[i]!='0')

{

node h=new node(fre[i],str[i]);

msg.add(h);

}

}

node root=null;//encoding

while(msg.size()>1)

{

node p = msg.peek();

msg.poll();//remove last

node q = msg.peek();

msg.poll();//remove second last

node temp = new node(p.frequency+q.frequency);

temp.left=p;

temp.right=q;

root=temp;

msg.add(temp);//add sum 2

}

comsg="";

printcodes(root,"");

for(i=0;i<message.length();i++)

{

String da=""+message.charAt(i);

encodemessage=encodemessage+enm.get(da);

}

textArea.setText(textArea.getText()+"\n"+"sen : "+encodemessage);

encodemessage=encodemessage+"@"+comsg;

dos.writeUTF(encodemessage);

dos.flush();

textArea\_2.setText(null);

} catch (Exception e1) {

}

}

});

}

}

1. **OUTPUT**

Here, In first frame message “Hello” is typed and after pressing ‘send’ button, message is received in next frame. In the process of transferring message from one frame to other message is encoded at sender (first frame) and decoded (second frame) at receiver.

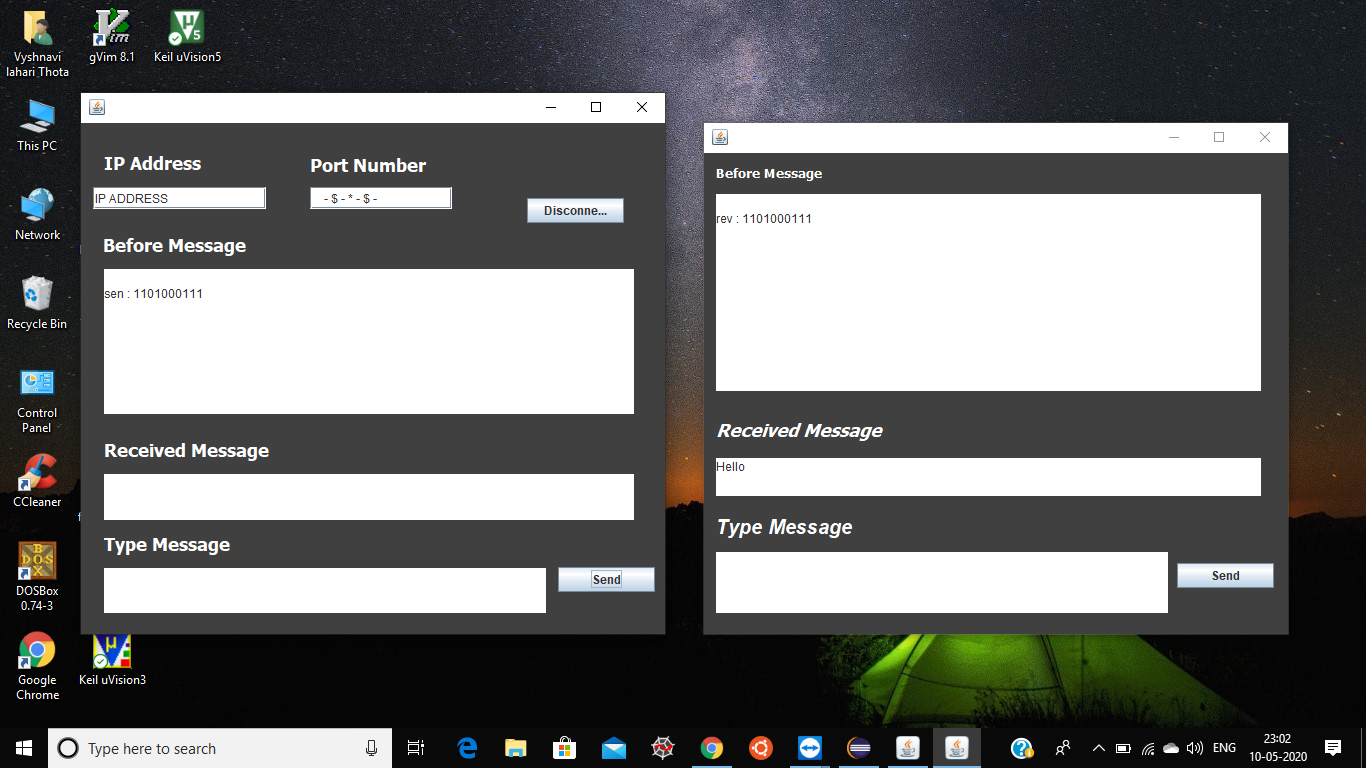


Fig 5.1 Output 1

Here, In second frame message “Hi” is typed and after pressing ‘send’ button, message is received at first frame. In the process of transferring message from one frame to other message is encoded at sender (second frame) and decoded (first frame) at receiver.

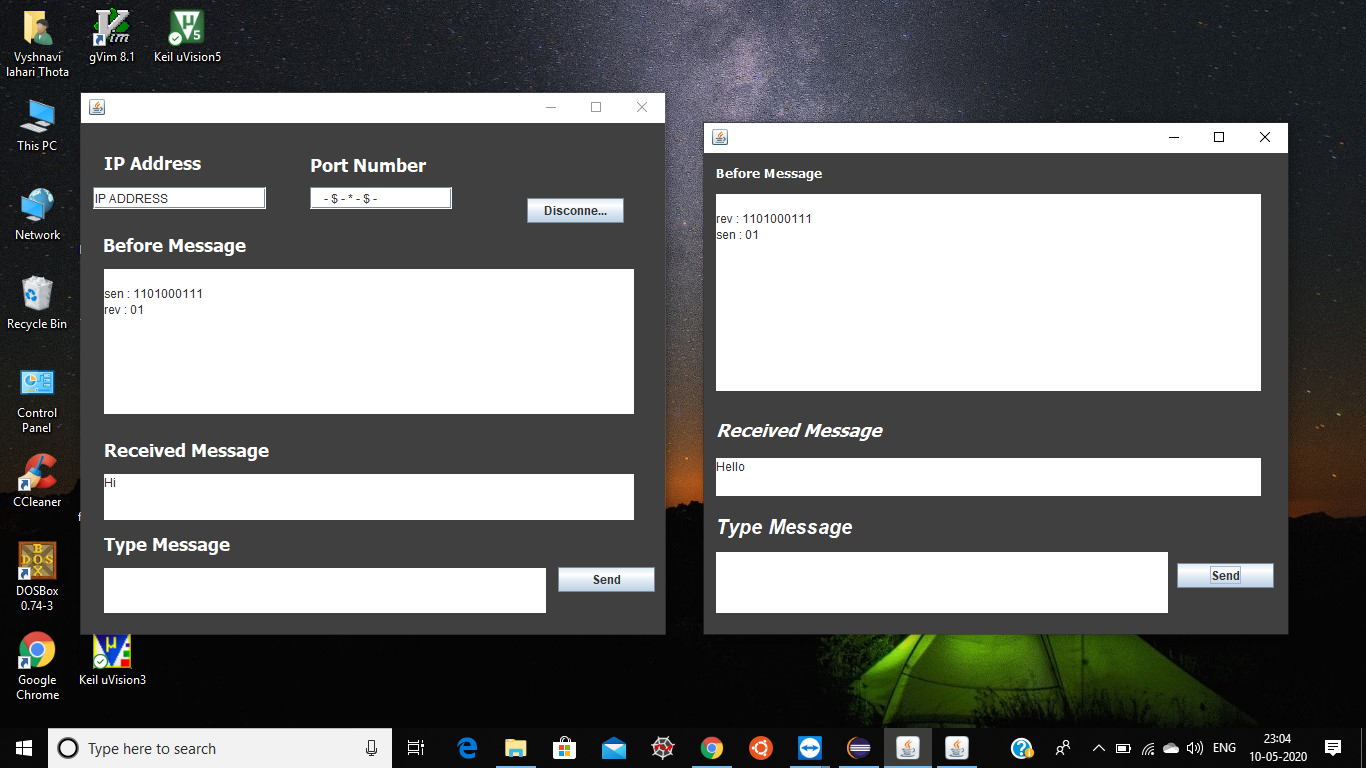


Fig 5.2 Output 2

1. **CONCLUSION AND FUTURE SCOPE**

Through this project, we have implemented a mini chat application using Java for GUI and socket programming. We have used Huffman algorithm to encode and decode messages.

This project can be further developed by including options to share files, videos etc and also by encrypting messages along with encoding to ensure confidentiality.

1. **REFERENCES**

* <https://www.cs.auckland.ac.nz/software/AlgAnim/huffman.html>
* <https://www.geeksforgeeks.org/socket-programming-in-java/>
* <https://www.programiz.com/dsa/huffman-coding>