PROJECT REPORT

on

# Huffman Encoding & Decoding

Subject: Data Structures and Algorithms

Semester: III

Batch: D3

Date: 4th October 2019

Prepared by:

Janak Vaghasiya (CE-122)

and

Jayesh Zinzuvadia (CE-128)

**DEPARTMENT OF COMPUTER ENGINEERING**

**FACULTY OF TECHNOLOGY**

**DHARMSINH DESAI UNIVERSITY, NADIAD**

**2019-20**

## Problem Description

When we want to transfer data lets' say "*Namaste!*" from one computer to another computer, then it cannot directly transfer the data as it is but first, it converts the data into its binary (i.e. encoding), transmits it into the another machine and the other machine will then decode the received data into its original form (i.e. decoding).

Now, to convert the data into binary we may use either ASCII values or assign some suitable codes to the characters for encoding.

**[A]** Using ASCII values for Encoding:-

| Character | ASCII value | Binary |
|---|---|---|
| N | 78 | 01001110 |
| a | 97 | 01100001 |
| m | 109 | 01101101 |
| s | 115 | 01110011 |
| t | 116 | 01110100 |
| e | 101 | 01100101 |
| ! | 33 | 00100001 |

Using this approach the above data (in binary) will be as:-

01001110 01100001 01101101 01100001 01110011 01110100

01100101 00100001

Limitations:-

1) Occupies more bits as in the preceding example, 8 character string requires 64 bits.

2) Useful only for the characters present in ASCII table.

**[B]** Assigning codes (having fixed length) to the characters:-

As there are only 7 characters so we can assign codes of length 3.

| Character | Codes | Binary |
|---|---|---|
| N | 0 | 000 |
| a | 1 | 001 |
| m | 2 | 010 |
| s | 3 | 011 |
| t | 4 | 100 |
| e | 5 | 101 |
| ! | 6 | 110 |

Using this approach the above data (in binary) will be as:-

000 001 010 001 011 100 101 110

Limitation: Occupies 24 bits instead of 64 bits but still it will tend to occupy more space as the characters increases and the corresponding codes length increases.

**[C]** Assigning codes (having variable length) to the characters:-

| Character | Variable length |
|---|---|
| N | 0 |
| a | 1 |
| m | 00 |
| s | 01 |
| T | 10 |
| E | 11 |
| ! | 000 |

Data (in binary) is:-

0 1 00 1 01 10 11 000

Limitation: Occupies 14 bits instead of 24 bits but there is an ambiguity because the variable length codes assigned to ‘m’ is prefix of codes assigned to ‘N’ and ‘a’. So, while decoding the compressed bit stream 0100, it may give different results like: “sm” or “NaNN” or “Nam”.

**Problem:**

Prefix codes problem in assigning the variable length codes to the characters WHERE:

For encoding: Take input from file (in .txt format) and generate output (in .dat format)

For decoding: Take input file (in .dat format) and generate original file (in .txt format)

**Applications:**

1) Useful in data compression.

2) Shortens data-transmission

3) Can be used in conjunction with Cryptography thereby making it difficult for the attackers to crack the code.

## Solution

So, the solution to the above problem is **Huffman Coding**, developed by **David A. Huffman**. Huffman Coding is an encoding algorithm widely used as a lossless data compression technique.

The idea is to assign variable length codes to input characters. Lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

The variable-length codes assigned to input characters are Prefix Codes means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bit stream.

The following general procedure is applied for construction a Huffman tree:
→Search for the two nodes having the lowest frequency, which are not yet assigned to a parent node.
→Couple these nodes together to a new interior node.
→Add both the frequencies and assign this value to the new interior node.
→The procedure has to be repeated until all nodes are combined together in a root node.

**Example**: Construct a Huffman tree by using these nodes.

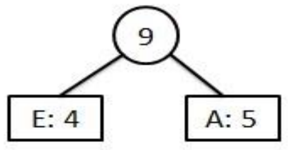
| **Value** | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| **Frequency** | 5 | 25 | 7 | 15 | 4 | 12 |

**Solution:**

**Step 1:** According to the Huffman coding we arrange all the elements (values) in ascending order of the frequencies.

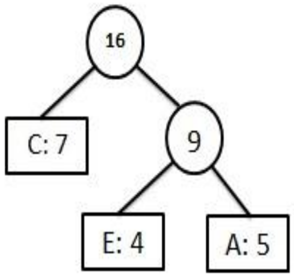
| **Value** | E | A | C | F | D | B |
|---|---|---|---|---|---|---|
| **Frequency** | 4 | 5 | 7 | 12 | 15 | 25 |

**Step 2:** Insert first two elements which have smaller frequency.



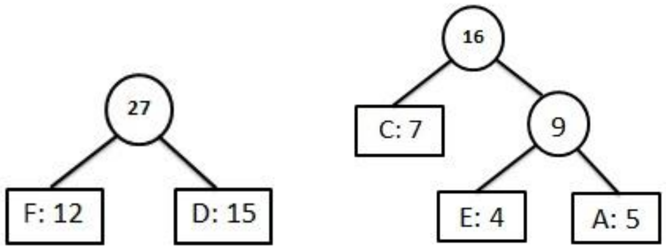
| **Value** | C | EA | F | D | B |
|---|---|---|---|---|---|
| **Frequency** | 7 | 9 | 12 | 15 | 25 |

**Step 3:** Taking next smaller number and insert it at correct place.



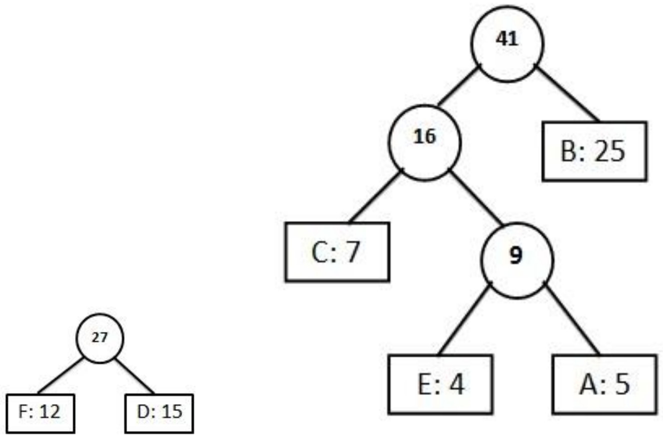
| Value | F | D | CEA | B |
|---|---|---|---|---|
| Frequency | 12 | 15 | 16 | 25 |

**Step 4:** Next elements are F and D so we construct another sub tree for F and D.



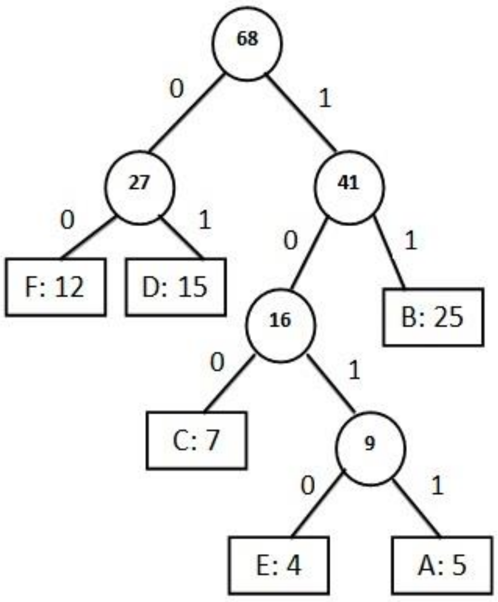
| Value | CEA | B | FD |
|---|---|---|---|
| Frequency | 16 | 25 | 27 |

**Step 5:** Taking next value having smaller frequency then add it with CEA and insert it at correct place.



| Value | FD | CEAB |
|---|---|---|
| Frequency | 27 | 41 |

**Step 6:** We have only two values hence we can combined by adding them.



**Huffman Tree**

| **Value** | FDCEAB |
|---|---|
| **Frequency** | 68 |

**Step 7**: Obtain codes from the Huffman Tree:

| Character | Frequency | Codes |
|---|---|---|
| A | 5 | 1011 |
| B | 25 | 11 |
| C | 7 | 100 |
| D | 15 | 01 |
| E | 4 | 1010 |
| F | 12 | 00 |

Using Huffman encoding we can compress the data using variable length codes obtained from Huffman tree.
Similarly, Huffman decoding is accomplished using these codes.

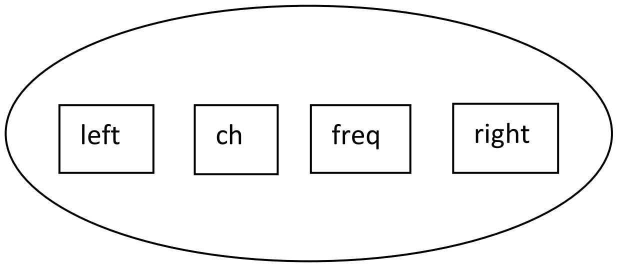
## Design and Algorithm

Class Diagrams:-

[A] **Huffman Encode**:

1) *TNode* class for Bin and Text

| TNode class |
|---|
| ch : int<br>freq : int<br>left : TNode<br>right : TNode |
| TNode (){}<br>+ TNode (int ch, int freq){} |



2) *UserInput* class

| UserInput class |
|---|
| b1 , b2 : Button<br>tb : TextField<br>oldsize , newsize, msg : Label |
| UserInput (){} |
| + actionPerformed(ActionEvent ae) : void<br>+ main (String args[]) : void<br>isText(String fname) : Boolean |

3) *THuffmanEncode* class for Text

| THuffmanEncode class |
|---|
| tree : TNode<br>obj : ExtraInfo<br>- temp[] , nullcode[] , codes[][][] : byte |
| THuffmanEncode (int totalchr , int freq[][] , char ch[][] ){} |
| + main(String fname) : long<br>findCode(TNode curr,int id) : void |

4) *BHuffmanEncode* class for Bin

| BHuffmanEncode class |
|---|
| tree : TNode<br>obj : ExtraInfo<br>- temp[] , nullcode[] , codes[][][] : byte |
| BHuffmanEncode (char ch[] , int freq[] , int totalchr){} |
| + main(String fname) : long<br>findCode(TNode curr,int id) : void |

5) *CONS* interface for Text

| CONS interface |
|---|
| + SIZE1= 512 , SIZE2=128 . SIG=65535 , MINFILESIZE=3 : int |

*CONS* interface for Bin

| CONS interface |
|---|
| SIZE= 256 , SIG=256 , MINFILESIZE=3 : int |

6) *FileIO* class for Bin and Text

| FileIO class |
|---|
| ch[][] : char<br>freq[][] , totalchr : int<br>source : String |
| FileIO(String source){} |
| readFile() : void<br>writeFile(char extra_info_ch[][],byte extra_info_code[][][],byte sigcode[], byte codes[][][]) :long<br>isText(String fname) : Boolean |

7) *FileInput* class for Bin and Text

| FileInput class |
|---|
| // for text only<br>+ scanTextFile(String source,char [][]ch,int [][]freq) : int<br>// for bin only<br>+ scanBinFile(String source,char [][]ch,int [][]freq) : int |

8) *FileOutput* class for Bin and Text

| FileOutput class |
|---|
| // for text only<br>writeCompressedText(char extra_info_ch[][],byte extra_info_code[][][],byte sigcode[],byte codes[][][],String source) : long<br>// for bin only<br>writeCompressedBin(char extra_info_ch[][],byte extra_info_code[][][],byte sigcode[],byte codes[][][],String source) : long |

9) *BitIOQ* class for Bin and Text

| BitIOQ class |
|---|
| - q[] , count : byte<br>front , rear : int |
| BitIOQ(int size){} |
| insertCode(byte ... datas) : Boolean<br>removeByte() : int<br>getCount() : byte |

10) *ExtraInfo* class for Bin and Text

| ExtraInfo class |
|---|
| codes[][][] , sigcode[] : byte<br>ch[][] : char<br>MINLEN=1 : int<br>- endch[] , totalchr , len1 : int |
| ExtraInfo (int level,int totalchr){} |
| insert (int ch,byte ... code) : boolean<br>endEntry() : boolean<br>print() : void |

11) *ListNode* inner class of HuffmanTreeLinkedList

| ListNode class |
|---|
| data : TNode<br>next : ListNode |
| ListNode(){}<br>+ ListNode (TNode data,ListNode next){} |

12) *HuffmanTreeLinkedList* class

| HuffmanTreeLinkedList class |
|---|
| - ListNode inner class<br>fillLL(char[][]ch,int[][]freq,int totalchr) : ListNode<br>makeTree(ListNode head,int totalchr) : TNode<br>printTree(TNode root) : void<br>drawTree(TNode arr[][]) : void<br>printLL(ListNode head) : void<br>getTree(char[][]ch,int[][]freq,int totalchr) : TNode |

[B] **Huffman Decode**:

1) *UserInput* class

| UserInput class |
|---|
| b1 , b2 : Button<br>tb : TextField<br>oldsize , newsize, msg : Label |
| UserInput (){} |
| + actionPerformed(ActionEvent ae) : void<br>+ main (String args[]) : void<br>isDat(String fname) : boolean |

2) De*CodeQueue* class

| DeCodeQueue class |
|---|
| - code[][][] , q[] : byte<br>- chr[][] , ch[] : char<br>- count , front . rear , MINLEN : int |
| DeCodeQueue (int size,byte[][][]code,char chr[][]){} |
| insertByte(byte ... datas) : boolean<br>peep(int idx) : byte<br>removeChar() : int<br>getCount() : int<br>print() : void<br>printQ() : void |

3) *FileIO* class

| FileIO class |
|---|
| decode(String source) : long<br>generateTextFile(ObjectInputStream ois,String extention) : long<br>generateBinFile(ObjectInputStream ois,String extention) : long<br>binCode(int num) : byte[] |

4) *BitIOStack* class

| BitIOStack class |
|---|
| - BitIOStack[] , top . bin[] : byte |
| BitIOStack (int size) {} |
| push(int data) : boolean<br>popArr () : byte[] |

Pseudo Code:-

[A] For Encoding:

**Algo. HuffmanEncode (File f)**

```
1) [For First Pass: Read the file 'f' character by character and count the frequency of characters]
        Call readFile(f)

2) [First, it creates a linked list of tree nodes and then it creates a Huffman Tree by taking two
minimum frequency nodes from the linked list and combining them repeatedly]
        Call fillLL()
        Call makeTree()

3) [Find codes from the Huffman Tree and store them in an array]
        Call findCode()

4) [For Second Pass: Read the file character by character and store its corresponding code into a
new file]
        Call writeFile()

5) [Finished]
        Return
```

**Algo. makeTree(head,totalchr)**

Creates Huffman Tree from the linked list of tree nodes.
First, It selects Two minimum frequency nodes from the linked list of tree nodes.
Second, it combines that Two minimum frequency nodes by creating a Parent node whose left and right reference links to that Two minimum frequency nodes. It stores the sum of the frequencies of that Two nodes as its repeat value.
Third, it sets the remaining links of linked list after the Parent node creation.
Fourth, it inserts the Parent node at the first position of linked list.

```
1) [Initialize]
        tree←NULL

2) [Check whether list is empty or not]
        If totalchr=0 then
                Return tree

3) [Check whether list contains only one node]
        If totalchr=1 then
                tree←head.data
                Return tree
```

```
4) [Constructing Huffman Tree]
        Repeat thru step 10 while (head.next!=NULL)

5) [Initializing fields]
                prvT←head.next
                t←prvT.next

6) [Searching for two nodes having minimum frequency]
                if((head.data.freq)<(head.next.data.freq))
                then
                        min←head
                        smin←head.next
                        prvMin←NULL
                        prvSmin←min
                else
                        min←head.next
                        smin←head
                        prvMin←smin
                        prvSmin←NULL
                endif

                // If the list have only two nodes then come out of the loop
                If (head.next.next=NULL)
                then
                        break

                // Traversing the linked list until min and smin nodes are found
                while(t!=NULL)
                        if(t.data.freq<=min.data.freq)
                        then
                                smin←min
                                prvSmin←prvMin
                                min←t
                                prvMin←prvT
                        else if(t.data.freq<=smin.data.freq)
                        then
                                smin←t
                                prvSmin←prvT
                        endif
                        prvT←t
                        t←t.next

7) [Determines the level of tree]
                if(min.data.ch>=0 && smin.data.ch>=0) then
                        chvalue ← -1
                else if(min.data.ch<smin.data.ch) then
                        chvalue ← min.data.ch-1
                else
                        chvalue ← smin.data.ch-1
                endif
```

8) [Combining two tree nodes to form its Parent Node]

```
        temp ←NEW(TNode)
        temp.ch ←chvalue
        temp.freq ← min.data.freq + smin.data.freq
        temp.left ←min.data
        temp.right ←smin.data
```

9) [Adjusting remaining links in the list]

```
        if(min.next=smin)
        then
                if(prvMin!= NULL) then
                        prvMin.next←smin.next
                else
                        head←smin.next
                endif
        else if(smin.next=min)
        then
                if(prvSmin!= NULL)
                        prvSmin.next←min.next
                else
                        head←min.next
                endif
        else
                if(prvMin!= NULL) then
                        prvMin.next←min.next
                else
                        head←head.next
                endif
                if(prvSmin!= NULL) then
                        prvSmin.next←smin.next
                else
                        head←head.next
                endif
        endif
```

10) [Insert Parent node at the first position of the linked list]

```
        t←NEW(ListNode)
        t.data←temp
        t.next←head
        head←t
```

```
11) [Determines level of tree]
        if(min.data.ch>=0 && smin.data.ch>=0) then
                chvalue ← -1
        else if(min.data.ch<smin.data.ch) then
                chvalue ← min.data.ch-1
        else
                chvalue ← smin.data.ch-1
        endif

12) [Finishing last step for tree construction]
        temp←NEW(TNode)
        temp.ch ←chvalue
        temp.freq←head.data.freq + head.next.data.freq
        temp.left←min.data
        temp.right←smin.data

13) [Assigning root reference to the tree node]
        tree←temp

14) [Finished]
        Return tree
```

**Algo. findCode(curr, id)**

It recursively finds codes of the characters.

```
1) [Proceed left sub tree]
                    if(curr.left!=null)
                    then
                              temp[id] ←0
                              findCode(curr.left , id+1)
                    endif

2) [Proceed right sub tree]
                    if(curr.right!=null)
                    then
                              temp[id] ←1
                              findCode(curr.right , id+1)
                    endif

3) [At leaf nodes]
                    if( curr.left=NULL && curr.right=NULL )
                    then
                              temp1[]←NEW
                              id1←curr.ch/SIZE2
                              id2←curr.ch-(id1* SIZE2)

                              if(code[id1]=NULL) then
                                        code[id1] ←NEW([CONS.SIZE2][])
                              endif

                              // Storing bits of codes
                              for(int i←0 ; i<id ; i←i+1)
                                        temp1[i]←temp[i];

                              // Assigning codes into code[][][]
                              code[id1][id2] ←temp1;

                              // Inserting code obtained from recursion into codes[][][] and ch[][]
                              obj.insert(curr.ch,temp1);

4) [Finished]
          Return
```

[B] For Decoding:

**Algo. Decode(SOURCE_FILE,DESTINATION_FILE)**

```
1)[read extra information of characters and codes]
    EXTRA_INFO_CH[][] ← SOURCE_FILE.readObject();
    EXTRA_INFO_CODE[][][] ← SOURCE_FILE.readObject();
    MAX_CODE_LEN ←findMaxCodeLen(EXTRA_INFO_CODE);

2)[decode sourceFile and generate destinationFile]
    Q ← new DeCodeQueue(MAX_CODE_LEN +(8*2), EXTRA_INFO_CH,
                        EXTRA_INFO_CODE);

    Repeate while(true)
    {
        Repeate while(getCount(Q) < MAX_CODE_LEN)
            CODE←bincode(SOURCE_FILE.readByte())
            insertByte(Q,CODE);
        IF((CH←removeChar(Q))=SIGNAL)
            Write "SIGNAL CODE FOUND"
            Break;
        DESTINATION_FILE.write(CH);
    }
```

**Algo. removeChar(Q)**

```
1)[read data bit by bit from queue until matching character not found]
    CODE,CH←null;
    Repeat while(true)
    {
        CODE=CODE+nextBit(Q)
        CH←Q.findValue(CODE);
        If(CH≠null)
            Return CH;
    }
```

## Source Code

1) HuffmanEncode.java

```
package huffmanencode;
import huffmanencode.text.THuffmanEncode;
import huffmanencode.bin.BHuffmanEncode;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class UserInput extends Frame implements ActionListener
{
	Button b1=new Button("choose file"),b2=new Button("compress");
	TextField tb=new TextField(50);
	Label oldsize=new Label(),newsize=new Label(),msg=new Label();
	/* Creating, Initializing and Describing Events of various Frame objects */
	UserInput()
	{
		super("encode");
		setSize(500,500);
		setVisible(true);
		setLayout(new GridLayout(6,1,0,10));
		Panel arr[]={new Panel(),new Panel(),new Panel()};
		arr[0].add(b1);
		arr[1].add(b2);
		arr[2].add(tb);
		Font f=new Font("Arial",Font.BOLD,15);
		oldsize.setFont(f);		oldsize.setForeground(Color.RED);
		newsize.setFont(f);		newsize.setForeground(Color.RED);
		tb.setEditable(false);
		add(arr[0],0);		add(arr[2],1);
		add(oldsize,2);
		add(newsize,3);
		add(arr[1],4);		add(msg,5);

		b1.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent ae)
			{
				FileDialog fd=new
				FileDialog(UserInput.this,"choosefile",FileDialog.LOAD);
				fd.setVisible(true);
				String fname=fd.getDirectory()+fd.getFile();
				tb.setText(fname);
				oldsize.setText("");
				newsize.setText("");
```

```
                        File f=new File(fname);
                        if(f.exists()&&f.isFile())
                                oldsize.setText(("file size : "+f.length()/1000)+" kb");
                }
        });
        b2.addActionListener(this);

        addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e)
                {dispose();}
        });
}
/* Describes event for 'compress' button */
public void actionPerformed(ActionEvent ae)
{
        String fname=tb.getText();
        File f=new File(fname);
        long size=0;
        newsize.setText("");
        if(!f.exists() || !f.isFile())
        {
                newsize.setText("invalid file");
                return;
        }
        try
        {
                if(isText(fname))
                {size=THuffmanEncode.main(fname);}
                else
                {size=BHuffmanEncode.main(fname);}
        }
        catch(IOException e)
        {
                newsize.setText("can't compress file");
                return;
        }
        newsize.setText("compressedfilesize : "+(size/1000)+" kb");
}
public static void main(String[] args)
{       new UserInput();        }
/* Determines type of file to be compressed*/
static boolean isText(String fname)
{
        String arr[]={"txt","java","py","c"};
        int i=fname.lastIndexOf('.');
        String extention="";
```

```
		if(i>0)
			{extention=fname.substring(i+1);}
		for(String str:arr)
		{
			if(str.equals(extention))
				{return true;}
		}
		return false;
	}
}
```

2) BHuffmanEncode.java

```
package huffmanencode.bin;
import java.io.IOException;
import java.io.FileOutputStream;
import java.io.FileReader;

public class BHuffmanEncode
{
	TNode tree;
	ExtraInfo obj;
	private byte temp[],nullcode[],codes[][]=new byte[CONS.SIZE+1][];
	/*it will initialize tree*/
	BHuffmanEncode(char[]ch,int[]freq,int totalchr)
	{
		tree=HuffmanTreeLinkedList.getTree(ch,freq,totalchr);
		int level=(-1)*tree.ch+1;
		temp=new byte[totalchr];
		obj=new ExtraInfo(level,totalchr);
		this.findCode(tree,0);
		obj.endEntry();
	}
	public static long main(String fname)throws IOException
	{
		BHuffmanEncode encode=null;
		FileIO f=new FileIO(fname); // (source)
		f.readFile();
		encode=new BHuffmanEncode(f.ch,f.freq,f.totalchr);
		f.ch=null;	f.freq=null;
		return
		f.writeFile(encode.obj.ch,encode.obj.codes,encode.obj.sigcode,encode.codes);
	}
	/*recursive method for finding code of character*/
	void findCode(TNode curr,int id)
	{
		if(curr.left!=null){
			temp[id]=0;
			findCode(curr.left,id+1);
		}
		if(curr.right!=null){
			temp[id]=1;
			findCode(curr.right,id+1);
		}
		if(curr.left==null&&curr.right==null){
			byte temp1[]=new byte[id];
			for(int i=0;i<id;i++)
				{	temp1[i]=temp[i];	}
			codes[curr.ch]=temp1;
			obj.insert(curr.ch,temp1);
		}
	}
}
```

3) FileIO.java

```
package huffmanencode.bin;
import java.io.*;
interface CONS
{
	int SIZE=256, SIG=256, MINFILESIZE=3;
}
class FileIO
{
	char[]ch=new char[CONS.SIZE+1];
	int freq[]=new int[CONS.SIZE+1],totalchr;
	private String source;
	FileIO(String source)
	{this.source=source;}
	void readFile()throws IOException
	{
		totalchr=FileInput.scanBinFile(source,ch,freq);
		ch[CONS.SIZE]=(char)CONS.SIG;
		freq[CONS.SIZE]=0;
		totalchr++;
	}
	/*IT WILL GENERATE BINARY FILE*/
	long writeFile(char extra_info_ch[][],byte extra_info_code[][][],byte sigcode[], byte
codes[][]) throws IOException
	{
		return
FileOutput.writeCompressedBin(extra_info_ch,extra_info_code,sigcode,codes,source);
	}
```

```
}
class FileInput
{
	public static int scanBinFile(String source,char []ch,int []freq) throws IOException
	{
		File f=new File(source);
		if(!f.exists() || !f.isFile())
			{throw new IOException("file not found");}
		if(source.contains("compress.dat"))
			{throw new IOException("invalid input file");}
		int c,totalchr=0;
		try(BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(source)))s
		{
			while((c=bis.read() )!= -1)//range of c = 0-255
			{
				if(c==0){freq[0]++;}
				else if(ch[c]==0){
					totalchr++;
					freq[c]++;ch[c]=(char)c;
				}
				else{freq[c]++;}
			}
		}
		if(freq[0]>0){totalchr++;}
		if(totalchr<CONS.MINFILESIZE)
			{throw new IOException("small file not allowed");}
		return totalchr;
	}
}
class FileOutput
{
	/*it will generate binary file*/
	static long writeCompressedBin(char extra_info_ch[][],byte extra_info_code[][][],byte
sigcode[],byte codes[][],String source)throws IOException
	{
		File f=new File(source);
		if(!f.exists() || !f.isFile())
			{throw new IOException("file not found");}
		int i=source.lastIndexOf('.');
		String extention="";
		if(i>0)
			{extention=source.substring(i+1);}
		int c;
		int maxcodelen=extra_info_code[extra_info_ch.length-1][0].length;
		BitIOQ q=new BitIOQ(maxcodelen*3);
```

```
                try(ObjectOutputStream oos=new ObjectOutputStream(new
FileOutputStream("compress.dat")))
                {
                        oos.writeObject(extention);
                        oos.writeChar(CONS.SIG);
                        oos.writeObject(extra_info_ch);
                        oos.writeObject(extra_info_code);
                        try(BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(source)))
                        {
                                out: while(true)
                                {
                                        while(q.getCount()<8)
                                        {
                                                if((c=bis.read())!=-1)
                                                {
                                                        q.insertCode(codes[c]);
                                                }
                                                else{
                                                        /*insert signalcode at end of file*/
                                                        q.insertCode(sigcode);
                                                        while(q.getCount()>0)
                                                        {oos.write(q.removeByte());}
                                                        System.out.println("SIGNAL CODE
ENTERED");

                                                        break out;
                                                }
                                        }
                                        oos.write(q.removeByte());
                                }
                        }
                }
                return new File("compress.dat").length();
        }
}
/*use for creating binary file*/
class BitIOQ
{
        private byte q[],count=0;
        int front=-1,rear=-1;
        BitIOQ(int size)
        {       q=new byte[size];       }

        boolean insertCode(byte ... datas)
        {
                for(byte data : datas)
```

```
			{
				if(rear==q.length-1){rear=0;}
				else {rear+=1;}
				if(front==rear){
					if(rear==0){rear=(byte)(q.length-1);}
					else {rear-=1;}
					return false;
				}
				count++;
				q[rear]=data;
				if(front==-1){front=0;}
			}
			return true;
		}
		int removeByte()
		{
			int value=0,rdx=128;
			byte data;
			for(int i=0;i<8;i++)
			{
				if(front==-1){data=0;}
				else{
					data=q[front];
					count--;
				}
				if(front==rear)
				{front=rear=-1;}
				else if(front==q.length-1){front=0;}
				else {front+=1;}
				value+=data*rdx;
				rdx/=2;
			}
			return value;
		}
		byte getCount(){return count;}
}
```

4) Node.java

```
package huffmanencode.bin;
class TNode
{
		int freq,ch;
		TNode left,right;
		TNode(){}
		public TNode(int ch,int freq)
```

```
        {this.ch=ch;this.freq=freq;}
}
```

5) HuffmanTreeLinkedList.java

```
package huffmanencode.bin;
import java.util.Stack;
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
class HuffmanTreeLinkedList
{
        private static class ListNode
        {
                TNode data;
                ListNode next;
                ListNode(){}
                public ListNode(TNode data,ListNode next)
                {
                        this.data=data;
                        this.next=next;
                }
        }
        /*it will create linklist which contain tree nodes*/
        static ListNode fillLL(char[]ch,int[]freq,int totalchr)
        {
                Stack<ListNode> avail1;
                Stack<TNode> avail2;
                avail1=new Stack<>();
                for(int i=0;i<=totalchr;i++)
                        {avail1.push(new ListNode());}
                avail2=new Stack<>();
                for(int i=0;i<=totalchr;i++)
                        {avail2.push(new TNode());}
                ListNode head=null,temp;
                /*finding code of SIGNAL value for end of file*/
                if(freq[0]>0){
                        temp=avail1.pop();
                        temp.data=avail2.pop();
                        temp.data.freq=freq[0];
                        temp.data.ch=0;
                        temp.next=head;
                        head=temp;
                }
                for(int i=1;i<ch.length;i++)
                {
```

```
                if(ch[i]==0)
                        {continue;}
                temp=avail1.pop();
                temp.data=avail2.pop();
                temp.data.freq=freq[i];
                temp.data.ch=ch[i];
                temp.next=head;
                head=temp;
        }
        return head;
}
/*it will create tree from given linklist*/
static TNode makeTree(ListNode head,int totalchr)
{
        TNode tree=null;
        if(totalchr==0)return tree;
        if(totalchr==1)
        {       tree=head.data;         return tree;    }
        ListNode prvT,prvMin,prvSmin,min=null,smin=null,t;
        int chvalue;
        TNode temp;
        while(head.next!=null)
        {
                prvT=head.next;
                t=prvT.next;
                if((head.data.freq)<(head.next.data.freq)){
                        min=head;smin=head.next;
                        prvMin=null;prvSmin=min;
                }
                else{
                        min=head.next;smin=head;
                        prvMin=smin;prvSmin=null;
                }
                if(head.next.next==null)break;
                while(t!=null)
                {
                        if(t.data.freq<=min.data.freq){
                                smin=min;prvSmin=prvMin;
                                min=t;prvMin=prvT;
                        }
                        else if(t.data.freq<=smin.data.freq){
                                smin=t;
                                prvSmin=prvT;
                        }
                        prvT=t;
                        t=t.next;
```

```
			}
			if(min.data.ch>=0 && smin.data.ch>=0)
				{chvalue=-1;}
			else if(min.data.ch<smin.data.ch)
				{chvalue=min.data.ch-1;}
			else
				{chvalue=smin.data.ch-1;}
			temp=new TNode(chvalue,min.data.freq+smin.data.freq);
			temp.left=min.data;
			temp.right=smin.data;
			if(min.next==smin)
			{
				if(prvMin!=null)
					{prvMin.next=smin.next;}
				else {head=smin.next;}
			}
			else if(smin.next==min)
			{
					if(prvSmin!=null)
						{prvSmin.next=min.next;}
					else {head=min.next;}
			}
			else{
				if(prvMin!=null)
					{prvMin.next=min.next;}
				else
					{head=head.next;}
				if(prvSmin!=null)
					{prvSmin.next=smin.next;}
				else
					{head=head.next;}
			}
			t=new ListNode(temp,head);
			head=t;
		}
		if(min.data.ch>=0 && smin.data.ch>=0)
			{chvalue=-1;}
		else if(min.data.ch<smin.data.ch)
			{chvalue=min.data.ch-1;}
		else
			{chvalue=smin.data.ch-1;}
		temp=new TNode(chvalue,head.data.freq+head.next.data.freq);
		temp.left=min.data;temp.right=smin.data;
		tree=temp;
		return tree;
	}
```

```
static void printTree(TNode root)
{
   if(root==null)
               {return;}
   int level=(-1)*root.ch+1;
   TNode arr[][]=new TNode[level][];
   for(int i=0;i<level;i++)
      {arr[i]=new TNode[(int)Math.pow(2,i)];}

   Queue<TNode> nodeq=new LinkedList<>();
   Queue<Integer> idxq=new LinkedList<>();
   TNode newline=new TNode(-1,-1),prev=null,curr;

   int plevel=1,idx,left,right;
   nodeq.add(root);nodeq.add(newline);
   idxq.add(0);
   arr[0][0]=root;
   while(true)
   {
      curr=nodeq.remove();
      if(curr==newline&&prev==newline){break;}
      prev=curr;
      if(curr==newline)
      {
         nodeq.add(newline);
         plevel++;continue;
      }

      idx=idxq.remove();
      left=idx*2;
      right=left+1;

      if(curr.left!=null){
               nodeq.add(curr.left);
               idxq.add(left);
               arr[plevel][left]=curr.left;
      }
      if(curr.right!=null){
         nodeq.add(curr.right);
         idxq.add(right);
         arr[plevel][right]=curr.right;
      }
   }
               drawTree(arr);
}
```

```
   static void drawTree(TNode arr[][])
        {
                try(BufferedWriter writer=new BufferedWriter(new
FileWriter("TreeRepresentation.txt")))
                {
                        int level=arr.length;
                        System.out.println(level);
                        int constrain1=(int)(Math.pow(2,level)-2)/2,
                                constraint2=constrain1;
                        for(int row=0;row<level;row++)
                        {
                                for(int space=0;space<constrain1;space++)
                                {
                                        for(int i=0; i<3; i++)
                                        {
                                                writer.write(' ');
                                        }
                                }
                                for(int col=0;col<Math.pow(2,row);col++)
                                {
                                        if(arr[row][col]==null)
                                        {
                                                for(int i=0;i<3;i++)
                                                {
                                                        writer.write(' ');
                                                }
                                        }
                                        else
                                        {
                                                String str=String.valueOf(arr[row][col].freq);
                                                if(str.length()<3)
                                                {
                                                        for(int loop=0; loop<str.length(); loop++)
                                                                writer.write(' ');
                                                        if(arr[row][col].left==null &&
arr[row][col].right==null)

                                                        {
                                                                writer.write(arr[row][col].ch);
                                                        }
                                                        else
                                                        {
                                                                writer.write(str);
                                                        }
                                                }
                                                else
                                                {
```

```
                        writer.write(String.valueOf(arr[row][col].freq));
                        }
                    }
                    if(col==Math.pow(2,row)-1)
                        break;
                    for(int i=0;i<constraint2;i++)
                    {
                        for(int j=0; j<3; j++)
                        {
                            writer.write(' ');
                        }
                    }
                }
                constraint2=constrain1;
                constrain1=(constrain1-1)/2;
                writer.newLine();
            }
        }
        catch(IOException io){}
    }
    static void printLL(ListNode head)
    {
        ListNode temp=head;
        int i=0;
        System.out.println();
        while(temp!=null){
            System.out.print("("+temp.data.ch+"|"+temp.data.freq+")");
            temp=temp.next;i++;
        }
        System.out.print("{"+i+"}");
    }
    /*it will return huffman tree*/
    static TNode getTree(char[]ch,int[]freq,int totalchr)
    {
        ListNode head=fillLL(ch,freq,totalchr);
        TNode tree=makeTree(head,totalchr);
        return tree;
    }
}
```

6) ExtraInfo.java

```
package huffmanencode.bin;
class ExtraInfo
{
    byte codes[][][],sigcode[];
```

```
char ch[][];
/*MINIMUM LENGTH OF CODE IS 1 !*/
final int MINLEN=1;
private int endch[],totalchr,len1;
/*endch[i] point to lastelement+1 for each character's code*/
ExtraInfo(int level,int totalchr){
        ch=new char[level-1][];
        endch=new int[level-1];
        codes=new byte[level-1][][];
        this.totalchr=totalchr;
}
boolean insert(int ch,byte ... code)
{
        if(ch==CONS.SIG){sigcode=code;}
        int dim1=code.length-MINLEN;
        if(codes[dim1]==null){
                int value=(int)Math.pow(2,dim1+1);
                int dim2=(value<totalchr)?value:totalchr;
                this.ch[dim1]=new char[dim2];
                codes[dim1]=new byte[dim2][];
                len1++;
        }
        this.ch[dim1][endch[dim1]]=(char)ch;
        codes[dim1][endch[dim1]]=code;
        endch[dim1]++;
        return true;
}
boolean endEntry()
{
        char resizech[][]=new char[len1][];
        byte resizecodes[][][]=new byte[len1][][];
        int countid=0;

        for(int i=0;i<ch.length;i++)
        {
                if(ch[i]==null){continue;}
                resizech[countid]=new char[endch[i]];
                resizecodes[countid]=new byte[endch[i]][];
                for(int j=0;j<endch[i];j++){
                        resizech[countid][j]=ch[i][j];
                        resizecodes[countid][j]=codes[i][j];
                }
                countid++;
        }
        ch=resizech;
        codes=resizecodes;
```

```
		return true;
	}

	void print()
	{
		for(int r=0;r<ch.length;r++){
			for(int c=0;c<ch[r].length;c++){
				System.out.print(ch[r][c]+" ");
				if(codes[r][c]!=null)
				for(int i=0;i<codes[r][c].length;i++){
					System.out.print(codes[r][c][i]);
				}
				System.out.println();
			}
		}
	}
}
```

7) HuffmanDecode.java

```
package huffmandecode;
import java.io.*;
import java.awt.*;
import java.awt.event.*;

class UserInput extends Frame implements ActionListener
{
	Button b1=new Button("choose file"),b2=new Button("decompress");
	TextField tb=new TextField(50);
	Label oldsize=new Label(),newsize=new Label(),msg=new Label();
	UserInput()
	{
		super("decode");
		setSize(500,500);
		setVisible(true);
		setLayout(new GridLayout(6,1,0,10));
		Panel arr[]={new Panel(),new Panel(),new Panel()};
		arr[0].add(b1);
		arr[1].add(b2);
		arr[2].add(tb);
		Font f=new Font("Arial",Font.BOLD,15);
		oldsize.setFont(f);oldsize.setForeground(Color.RED);
		newsize.setFont(f);newsize.setForeground(Color.RED);
		tb.setEditable(false);
		add(arr[0],0);add(arr[2],1);
		add(oldsize,2);
```

```
                add(newsize,3);
                add(arr[1],4);add(msg,5);
                b1.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent ae)
                        {
                                FileDialog fd=new
FileDialog(UserInput.this,"choosefile",FileDialog.LOAD);
                                fd.setVisible(true);
                                String fname=fd.getDirectory()+fd.getFile();
                                tb.setText(fname);
                                oldsize.setText("");
                                newsize.setText("");
                                File f=new File(fname);
                                if(f.exists()&&f.isFile())
                                {oldsize.setText(("file size : "+f.length()/1000)+" kb");}
                        }
                });
                b2.addActionListener(this);

                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e)
                        {dispose();}
                });
        }

        public void actionPerformed(ActionEvent ae)
        {
                String fname=tb.getText();
                File f=new File(fname);
                long size=0;
                newsize.setText("");
                if(!f.exists() || !f.isFile())
                        {newsize.setText("invalid file");return;}

                try
                {
                        if(isDat(fname))
                        {size=FileIO.decode(fname);}
                        else
                        {newsize.setText("only .dat files are valid");return;}
                }
                catch(Exception e)
                {newsize.setText("can't decompress file");return;}
                newsize.setText("decompressedfilesize : "+(size/1000)+" kb");
        }
```

```
	public static void main(String[] args)
	{new UserInput();}

	static boolean isDat(String fname)
	{
		int i=fname.lastIndexOf('.');
		String extention="";
		if(i>0){extention=fname.substring(i+1);}

		if(extention.equals("dat"))
		{return true;}
		return false;
	}
}
```

8) FileIO.java

```
package huffmandecode;
import java.io.*;

class FileIO
{
	static long decode(String source) throws IOException,ClassNotFoundException
	{
		File f=new File(source);
		if(!f.exists()||!f.isFile())
			{throw new IOException("file not found");}

		int i=source.lastIndexOf('.');
		String extention="";
		if(i>0){extention=source.substring(i+1);}
		if(!extention.equals("dat"))
			{throw new IOException("invalid file for decoding");}


		try(ObjectInputStream ois=new ObjectInputStream(new
FileInputStream(source)))
		{
			extention=(String)ois.readObject();
			String arr[]={"txt","java","py","c"};
			for(String str:arr){
				if(str.equals(extention))
				{
					return generateTextFile(ois,extention);
				}
				}
				return generateBinFile(ois,extention);
			}

		}

		static long generateTextFile(ObjectInputStream ois,String extention)throws 
IOException,ClassNotFoundException
		{
			try(BufferedWriter bw=new BufferedWriter(new 
FileWriter("output."+extention)))
			{
				int SIG=ois.readChar();
				char [][]extra_info_ch=(char[][])ois.readObject();
				byte [][][]extra_info_code=(byte[][][])ois.readObject();

				int maxcodelen=extra_info_code[extra_info_code.length-1][0].length;
				CodeQueue q=new 
CodeQueue(maxcodelen+(8*2),extra_info_code,extra_info_ch);

				int ch;
				out : while(true)
				{
					int temp;
					while(q.getCount()<maxcodelen)
					{
						temp=ois.readByte();
						if(temp<0){temp+=256;}
						q.enQueue(binCode(temp));//printing bin code
					}
					if((ch=q.deQueue())==SIG)
					{
						System.out.println("SIGNAL CODE FOUND");
						break;
					}
					bw.write(ch);
				}
				return new File("output."+extention).length();
			}
		}

		static long generateBinFile(ObjectInputStream ois,String extention)throws 
IOException,ClassNotFoundException
		{
			try(BufferedOutputStream bis=new BufferedOutputStream(new 
FileOutputStream("output."+extention)))
```

```
		{
			int SIG=ois.readChar();
			char [][]extra_info_ch=(char[][])ois.readObject();
			byte [][][]extra_info_code=(byte[][][])ois.readObject();

			int maxcodelen=extra_info_code[extra_info_code.length-1][0].length;
			CodeQueue q=new
CodeQueue(maxcodelen+(8*2),extra_info_code,extra_info_ch);
			int ch;
			out : while(true)
			{
				int temp;
				while(q.getCount()<maxcodelen)
				{
					temp=ois.readByte();
					if(temp<0){temp+=256;}
					q.enQueue(binCode(temp));//printing bin code
				}
				if((ch=q.deQueue())==SIG)
				{
					System.out.println("SIGNAL CODE FOUND");
					break;
				}
				bis.write(ch);
			}
			System.out.println("*");
			return new File("output."+extention).length();
		}
	}

	/*it will return binary of given num in 8 bit*/
	static byte[] binCode(int num)
	{
		BitIOStack stk=new BitIOStack(8);
		while(num!=0)
		{
			stk.push(num%2);
			num=num/2;
		}
		byte code[] =stk.popArr();
		return code;
	}
}

class BitIOStack
{
	private byte BitIOStack[],top=-1,bin[];

	BitIOStack(int size)
	{BitIOStack=new byte[size];}

	boolean push(int data)
	{
		if(top==BitIOStack.length-1)return false;
		BitIOStack[++top]=(byte)data;
		return true;
	}
	byte[] popArr()
	{
		while(push(0));
		bin=new byte[BitIOStack.length];
		for(int i=BitIOStack.length-1,j=0;i>=0&&j<BitIOStack.length;i--,j++)
		{
			bin[j]=BitIOStack[i ];
		}
		return bin;
	}
}
```

9) CodeQueue.java

```
package huffmandecode;
class CodeQueue
{
	private byte code[][][],q[];
	private char chr[][],ch;
	private int count,front=-1,rear=-1;
	private final int MINLEN;
	/*create queue of given size and get code[][][] and chr[][]*/
	CodeQueue(int size,byte[][][]code,char chr[][])
	{
		this.code=code;this.chr=chr;
		MINLEN=code[0][0].length;
		q=new byte[size];
	}

	/*insert given byte[] into CircularQueue*/
	boolean enQueue(byte ... datas)
	{
		for(byte data : datas)
		{
			if(rear==q.length-1){rear=0;}
		else {rear+=1;}
		if(front==rear){
			if(rear==0){rear=(q.length-1);}
			else {rear-=1;}
			return false;
		}
		count++;
		q[rear]=data;
		if(front==-1){front=0;}
	}
	return true;
}
/*to get element present at given position from front*/
byte peep(int idx)
{
	if(idx>=count){return -128;}
	int pos=front+idx;
	if(pos>=q.length)
		{pos=pos-q.length;}
	return q[pos];
}
/* first it will check for 3 bit then 4 bit then 5 bit
 * and so on.....
 * if matched code found then it will return currosponding
 * char other wise it will return -1
 */
int deQueue()
{
	if(count<MINLEN){return -1;}
	int len=0;

	for(int idx=0;idx<chr.length;idx++)
	{
		/*finding matched code*/
		for(int id=0;id<chr[idx].length;id++)
		{
			len=0;
			for(int i=0;i<code[idx][id].length;i++){
				if(code[idx][id][i]==peep(i))
					{len++;}
				else break;
			}
			/*if matched code found then deQueue those many bits
			 *from CircularQueue*/
			if(len==code[idx][id].length)
			{
```

```
                    for(int i=0;i<code[idx][id].length;i++)
                    {
                        if(rear==front)
                        {front=rear=-1;}
                        else if(front==q.length-1){front=0;}
                        else{front++;}
                        count--;
                    }
                    return chr[idx][id];
                }
            }
        }
        return -1;
    }
    /*return number of element present inside queue*/
    int getCount()
    {return count;}
    void print()
    {
        for(char[]arr:chr){
            for(char c:arr)
                {System.out.print("("+(int)c+")");}
            System.out.println();
        }
        for(byte[][]byt:code){
            for(byte[]by:byt){
                for(byte b:by)
                    {System.out.print(b);}
                System.out.print(" ");
            }
            System.out.println();
        }
    }
    void printQ()
    {
        int visit=0;
        for(int i=front;visit<count;visit++,i++)
        {
            System.out.print(q[i]);
            if(i==q.length-1)i=-1;
        }
    }
}
```

## Testing

### [A] For Encoding:

**Input:**
Huffman coding is a data compression algorithm.

**Output:**
00010 00111 11100 11100 1000 010 1001 011 11010 1100 11011 1011 1001 11101 011 1011 1010 011 010 011 11011 010 11111 010 011 11010 1100 1000 00110 11110 00011 1010 1010 1011 1100 1001 011 010 00101 11101 1100 11110 1011 11111 00100 1000 00001 **00000**

### [B] For Decoding:

**Input:**
00010 00111 11100 11100 1000 010 1001 011 11010 1100 11011 1011 1001 11101 011 1011 1010 011 010 011 11011 010 11111 010 011 11010 1100 1000 00110 11110 00011 1010 1010 1011 1100 1001 011 010 00101 11101 1100 11110 1011 11111 00100 1000 00001 **00000**

**Output:**
Huffman coding is a data compression algorithm.

- ⇨ For comparing the contents of the input file and output file, use the following command:

```
Compare-Object -ReferenceObject $(Get-Content -Path C:\Test\Testfile1.txt)

-DifferenceObject $(Get-Content -Path C:\Test\Testfile2.txt)
```