**Update of the previous version (HuffmanCompressionASCII)**

Replacing PrintWriter in Encode\_InputFromTextFile() method with OutputStream and BufferedReader in Decode\_EncodedFile() method with InputStream

public void Encode\_InputFromTextFile()throws IOException{

BufferedReader br=new BufferedReader(new FileReader(file));

OutputStream outputstream=new FileOutputStream(tofile);

int bitcount=0;

byte write=0;

int ch;boolean start=true;

while((ch=br.read())!=-1){

for(int i=0;i<code[ch].length();i++){

char c=code[ch].charAt(i);

if(TotalBits<BitLength){

outputstream.write((byte)c);

continue;

}

if(start){

write|=c;start=false;

}

else{

write<<=1;

write|=c;

}

bitcount++;

if(bitcount==BitLength){

outputstream.write(write);

write='\0';

start=true;

TotalBits-=BitLength;

bitcount=0;

}

}

}

outputstream.flush();

outputstream.close();

}

public void Decode\_EncodedFile()throws IOException{

InputStream inputstream =new FileInputStream(encode);

PrintWriter pw=new PrintWriter(new FileWriter(decode));

int ch;

Node node=root;

while((ch=inputstream.read())!=-1){

if(TotalBits<BitLength){

if((char)ch==0){

node=node.left;

}else{

node=node.right;

}

if(node.value!='\0'){

pw.print((char)node.value);

node=root;

}

continue;

}

for(int i=(BitLength-1);i>=0;i--){

int c=(ch>>i)&1;

if(c==0){

node=node.left;

}else{

node=node.right;

}

if(node.value!='\0'){

pw.print((char)node.value);

node=root;

}

}TotalBits-=BitLength;

}

pw.flush();pw.close();

}

**IMPLEMENTATION AND RESULTS:**

Text files with varying sizes were compressed using the algorithm.

Data compression ratio = **Uncompressed File / Compressed File**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| `Name of file | Original Size(kb) | Time Taken to compress(sec) | Compressed File Size(kb) | Compression  Ratio | Time Taken to decompress(sec) |
| **small.txt** | **15** | **0.180632726** | **9** | **1.67** | **0.090824928** |
| **medium.txt** | **94** | **0.404426832** | **50** | **1.88** | **0.273155782** |
| **large.txt** | **163** | **0.650212432** | **90** | **1.811** | **0.446221603** |

**OBSERVATIONS:**

1. Time taken for compression and decompression has increased because InputStreamReader and OutputStreamWriter are slower as compared to BufferedReader and PrintWriter respectively.
2. Compression ratio in all the 3 cases has increased because in the previous version I was only writing 15 out of 16 bits of a char but now I am writing all the 8 bits of a byte.