Covid-19 Fasta File Compression

10.04.2020

Fenil Milankumar Parmar
170170116027
Sem : 6 (IT)
Data Compression and Data Retrieval

Overview

Compress the fasta sequence of Covid-19 virus by applying various compression algorithms. I have used the Shannon-Fano compression technique to compress Covid-19 Sequence which is in Fasta file format.

Goals

1. Read and analyse the fast file format.
2. Choose proper compression technique to compress the fasta sequence
3. Achieving the best compression ratio.
4. Retrive the compressed file.

Different compression techniques

- Many techniques can be applied to compress the fasta file.
- For instance,
    - 1)LZ77,LZ78,LZW
    - 2)Huffman-Coding,Shannon-Fano coding
    - 3)Burrows-wheeler encoding, Move-to-front encoding,RLE

Applied Method

1. Take the input as a sequence of characters
2. Apply Shannon-fano encoding to the given string.
3. It will give minimum bits to the character with maximum frequency.
4. The bit sequence is generated using the bits assigned by it.
5. Generate equivalent characters of it in a new file.
6. Store the characters and the table for mapping.
7. Generate equivalent bits and use table to decode the string.

Code

Header FIles and initialization of class

```
#include<iostream>
#include<bits/stdc++.h>
#include<vector>
#include<stdlib.h>
#include<math.h>
#include<fstream>
#include<windows.h>
using namespace std;

int actual;
class OP{
	public:
	int freq;
	char ch;
	string codw;
	void getd(){
		cout<<"Enter Character:";cin>>ch;
		cout<<"Enter Frequency:";cin>>freq;
		this->codw="";
	}
	void getd(char dh,int preq){
		this->ch=dh;
		this->freq=preq;
		this->codw="";
```

```
	}
	void display(){
		cout<<"Character:"<<ch<<"	Frequency:"<<freq<<endl;
	}
	void cop(){
		cout<<"Code of "<<ch<<"="<<codw<<endl;
	}
};
OP pi[127];

class chunks{
	public:
		string b8;
		char d8;
		int i8;
};
chunks cnks[1000000];
```

Shannon-Fano Coding Function

```
void shan(int s,int e)
{
	if(s<e)
	{
		int sum_1=0,sum_2=0;
		int mind=10000;
		int t=0;
		int diff=0;
		int pt;
		int i,j,k,cmp;

		for(cmp=0;cmp<(e-s);cmp++)
		{

			for(k=s;k<=s+cmp;k++)
			{
				sum_1=sum_1+pi[k].freq;
			}
			for(j=s+cmp+1;j<=e;j++)
			{
				sum_2=sum_2+pi[j].freq;
			}
			diff=abs(sum_2-sum_1);
			if(diff<=mind)
			{
				mind=diff;
				pt=cmp+s;
```

```
                    }
                    sum_1=0;
                    sum_2=0;
            }
                    for(i=s;i<=pt;i++)
                    {
                        pi[i].codw=pi[i].codw+"1";
                    }
                    cout<<endl;
                    for(i=pt+1;i<=e;i++)
                    {
                        pi[i].codw=pi[i].codw+"0";
                    }
                    int tmpp=pt;
                    shan(tmpp+1,e);
                    shan(s,tmpp);
        }
}


void rdfile()
{
        ifstream inFile;
        char ch;
        cout<<"Reading file = Covid-19.fasta";//REadinf=g Fasta file
        inFile.open("Covid-19.fasta");
        if (!inFile)
        {
          cout << "The input file could not be opened."<<endl;
```

```
  exit(0);
}
for(int i=0;i<127;i++)
{
        cout<<"Char("<<i<<")="<<char(i)<<endl;
        pi[i].getd(char(i),0);
}
ch = inFile.get();
cout<<"Covid-19.fasta file=";
while (ch != EOF)
{
        cout<<ch;
        pi[int(ch)].freq++;
ch = inFile.get();
}
}

void wtfile(int h)
{
        ifstream inFile;
        ofstream outfile;
        outfile.open("1_Codeword.txt");
        cout<<endl;
        char ch;
        cout<<endl;
        cout<<"Reading file = Covid-19.fasta"<<endl;
        cout<<"Generating 1_Codeword.txt file";
        inFile.open("Covid-19.fasta");
        if (!inFile)
```

```
        {
          cout << "The input file could not be opened."<<endl;
          exit(0);
        }

        ch = inFile.get();
        int i;
        cout<<"1_codeword file=";
        while (ch != EOF)
        {
                i=h;
                cout<<ch;
                while(pi[i].ch!=ch)
                {
                        i++;
                }
                outfile<<pi[i].codw;
        ch = inFile.get();
        }
        outfile.close();
        cout<<"File Generated!"<<endl;
}

int binTOdec(string binaryString)
{
        int value = 0;
        int indexCounter = 0;
        for(int i=binaryString.length()-1;i>=0;i--)
        {
```

```
            if(binaryString[i]=='1'){
                value += pow(2, indexCounter);
            }
    indexCounter++;
        }
        return value;
}
```

Compression Function

```
void dcp()
{
        cout<<endl;
        ifstream inFile;
        ofstream outfile;
        outfile.open("2_compressed.txt");
        char ch;
        cout<<"Reading 1_Codeword.txt file"<<endl;
        cout<<"Generating 2_compressed.txt file.";

        inFile.open("1_codeword.txt");
        if (!inFile)
        {
          cout << "The input file could not be opened."<<endl;
          exit(0);
```

```
        }
        ch = inFile.get();
        int i=0,lmt=0;
        cout<<"\tCHAR\tINT\tBINARY\tCint"<<endl;
        while (ch != EOF)
        {
                cnks[i].b8=cnks[i].b8+ch;
                while(cnks[i].b8.length()!=8)
                {
                        ch = inFile.get();
                        if(ch == EOF)
                        {
                                lmt++;
                                ch = '0';
                        }
                        cnks[i].b8=cnks[i].b8+ch;
                }
                cnks[i].i8 = binTOdec(cnks[i].b8);
                cnks[i].d8 = char(cnks[i].i8);

cout<<"\t"<<cnks[i].d8<<"\t"<<cnks[i].i8<<"\t"<<cnks[i].b8<<"\t"<<int(cnks[i].d8)<<endl;
                outfile<<cnks[i].d8;
                ch = inFile.get();
        i++;
        }
        actual = (i*8)-lmt;
        outfile.close();
        cout<<"File 2_compressed.txt Generated!"<<endl;
}
```

```
void fina()
{
        ifstream inFile;   // input file
        ofstream outfile;       //output file

        outfile.open("3_BinaryOfCompressed.txt");
        cout<<endl;
        char ch;
        cout<<"Reading file = 2_Compressed.txt"<<endl;
        cout<<"Generating 3_BinaryOfCompressed.txt file.";
        inFile.open("2_compressed.txt");
        if (!inFile)
        {
          cout << "The input file could not be opened."<<endl;
          exit(0);
        }
        ch = inFile.get();
        int i=0;
        cout<<"Results In 3_BinaryOfCompressed"<<endl;
        cout<<"Character\tBinary"<<endl;
        while(ch != EOF)
        {
                while (ch != EOF)
                {
                cout <<"\t"<< ch<<"\t"<<int(ch)<<"\t";
                        bitset<8> bin_x(ch);
                        outfile<<bin_x;
                        cout<<"binary ="<<bin_x<<endl;
                        ch = inFile.get();
```

```
                cout<<"CHAR"<<ch<<"\t"<<int(ch)<<"\t\t";
            }

            ch = inFile.get();
        }


        cout<<"Generated 3_BinaryOfCompressed.txt file!"<<endl;
        outfile.close();

}
```

Decompression Function

```
void las(int h)
{
        ifstream inFile;
        ofstream outfile;

//      inFile.open("3_BinaryOfCompressed.txt");
        inFile.open("1_Codeword.txt");
        outfile.open("4_Finalizedfile.txt");
        cout<<"Reading file = 3_BinaryOfCompressed.txt"<<endl;
```

```
cout<<"Generating 4_Finalizedfile.txt file.";
if(!inFile)
{
        cout << "The input file could not be opened."<<endl;
        exit(0);
}
char ch;
ch = inFile.get();
string str="";
cout<<"\nFinal File=";
for(int i=0;i<actual;i++)
{

        str=str+ch;
        for(int k=h;k<127;k++)
        {
                if(pi[k].codw==str)
                {
                        outfile<<pi[k].ch;
                        cout<<pi[k].ch;
                        str="";
                }
        }
        ch = inFile.get();
}
outfile.close();
cout<<endl;
cout<<"Generated 4_Finalizedfile.txt file!"<<endl;
```

```
}
```

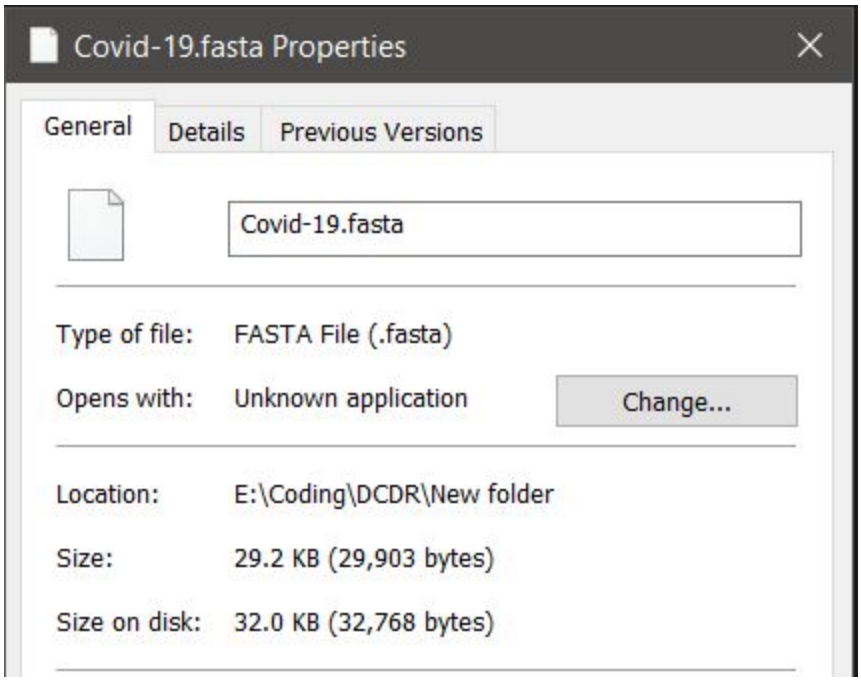
Main Function

```
int main()
{
	int i,j,k,n;
	int sum_1=0,sum_2=0;
	int mind=10000;
	int t=0;
	int diff=0;
	int pt;
	n=127;
	rdfile();

	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-i-1;j++)
		{
			if(pi[j].freq>pi[j+1].freq)
			{
				OP tmp;
				tmp=pi[j];
				pi[j]=pi[j+1];
				pi[j+1]=tmp;
			}
		}
	}
	int h;
```

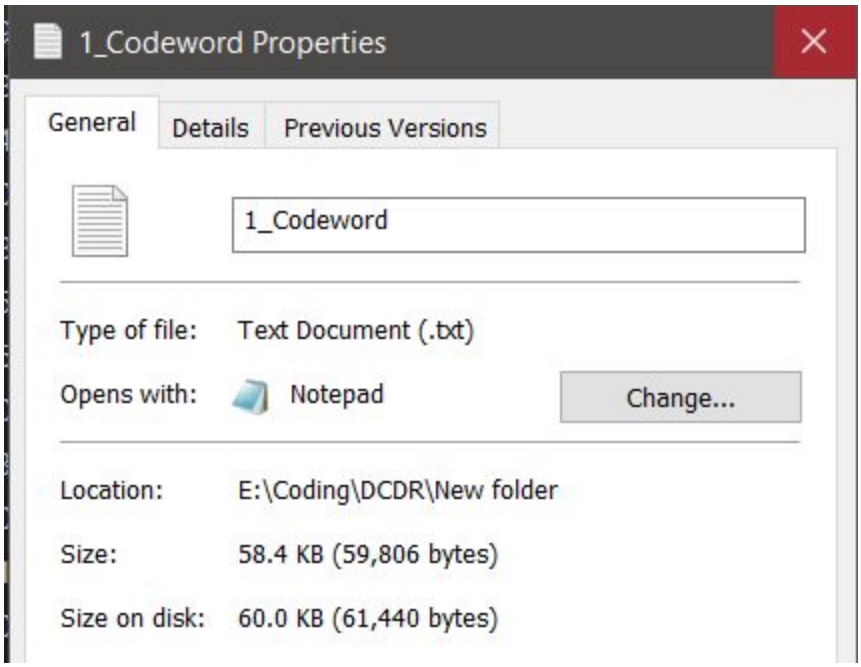
```
for(i=0;i<n;i++)
{
        if(pi[i].freq!=0)
        {
                h=i;
                break;
        }
}

for(i=h;i<n;i++)
{
        cout<<i;
        pi[i].display();
}

shan(h,n-1);
ofstream codfile;//codeword file
codfile.open("codfile.txt");

for(i=h;i<n;i++)
{
        codfile<<pi[i].ch<<" ||  Freq="<<pi[i].freq<<"  ||
Codeword="<<pi[i].codw<<endl;
        pi[i].cop();
}
codfile.close();
wtfile(h);
dcp();
```

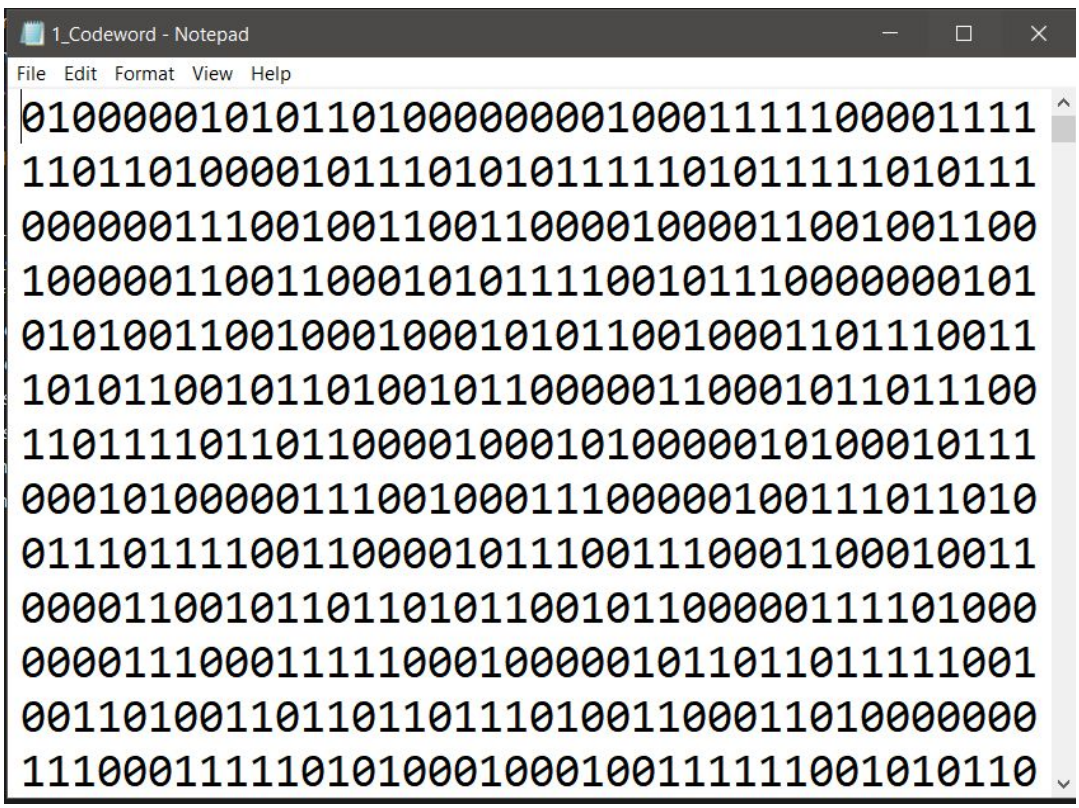
```
cout<<"ACTUAL======================================================"<<actual<<e
ndl;
        fina();
        las(h);
        return 0;
}
```

Input Fasta FIle

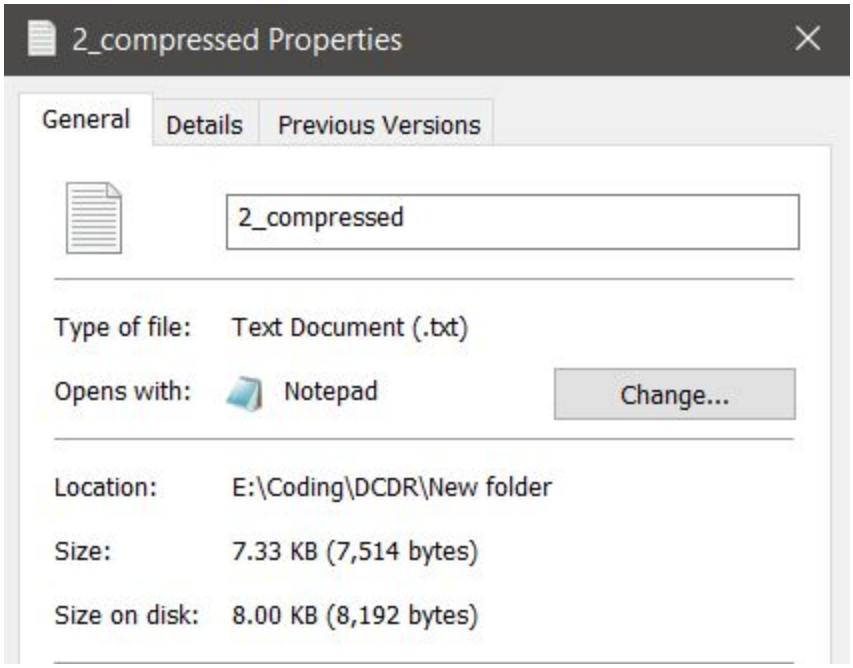


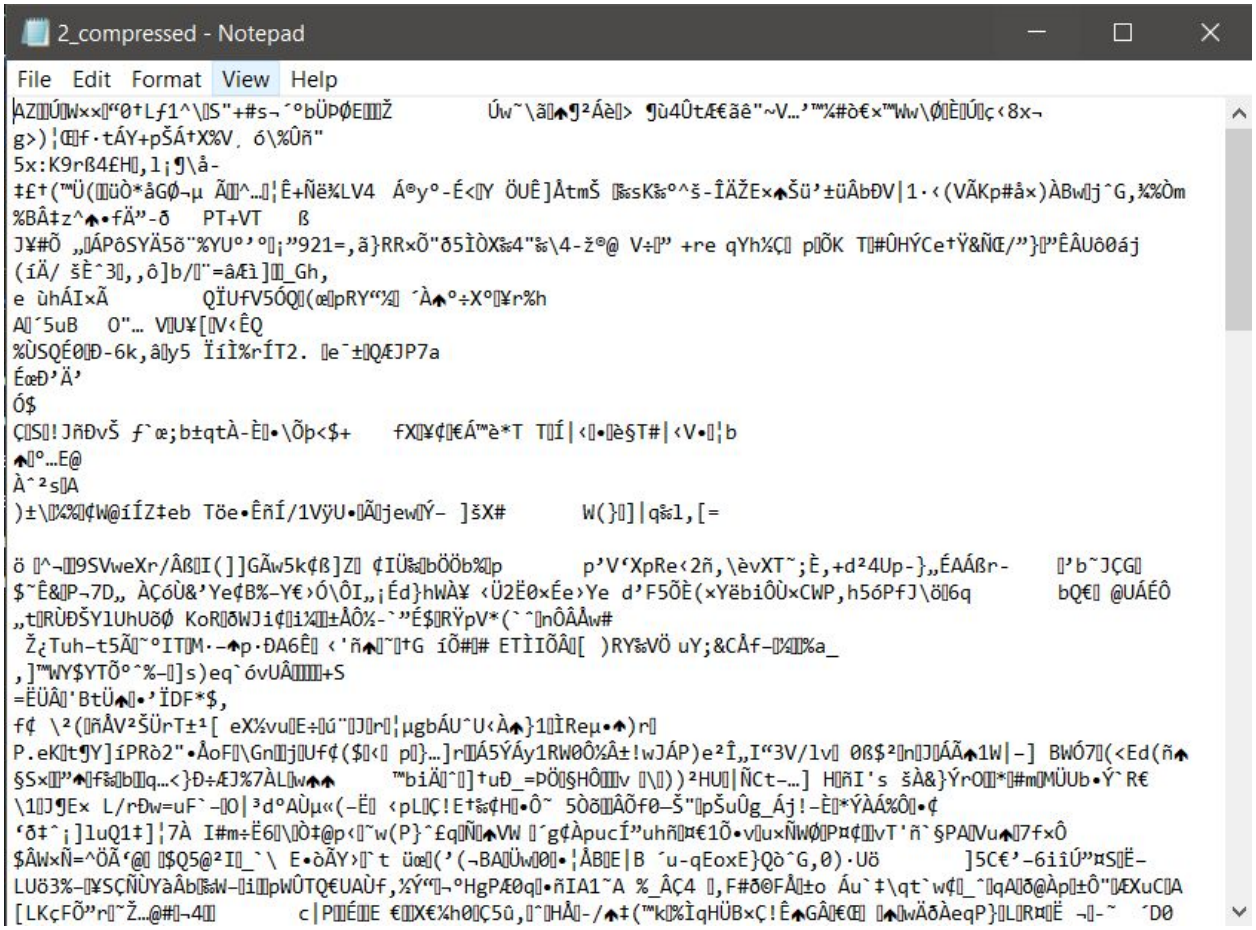
Binary Encoded FIle



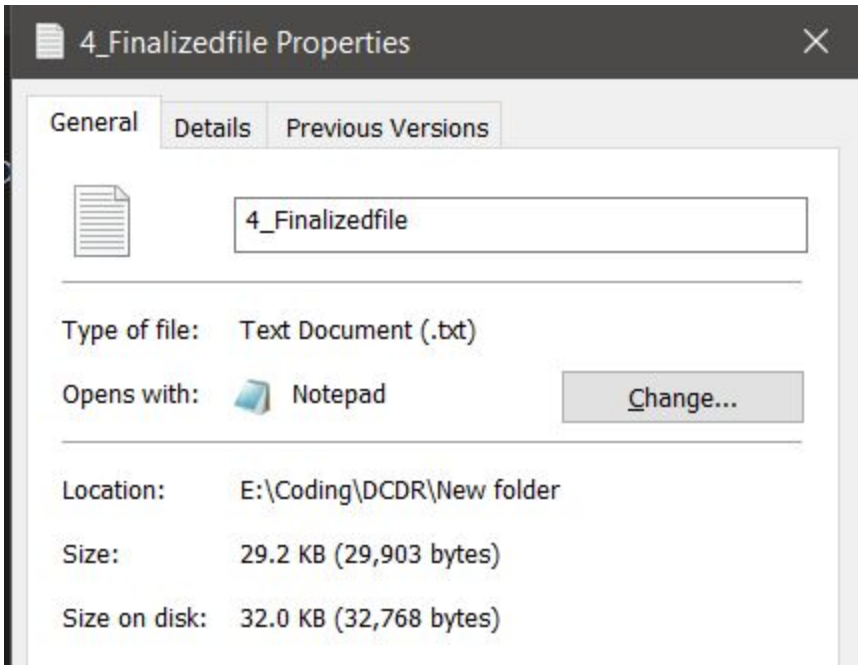


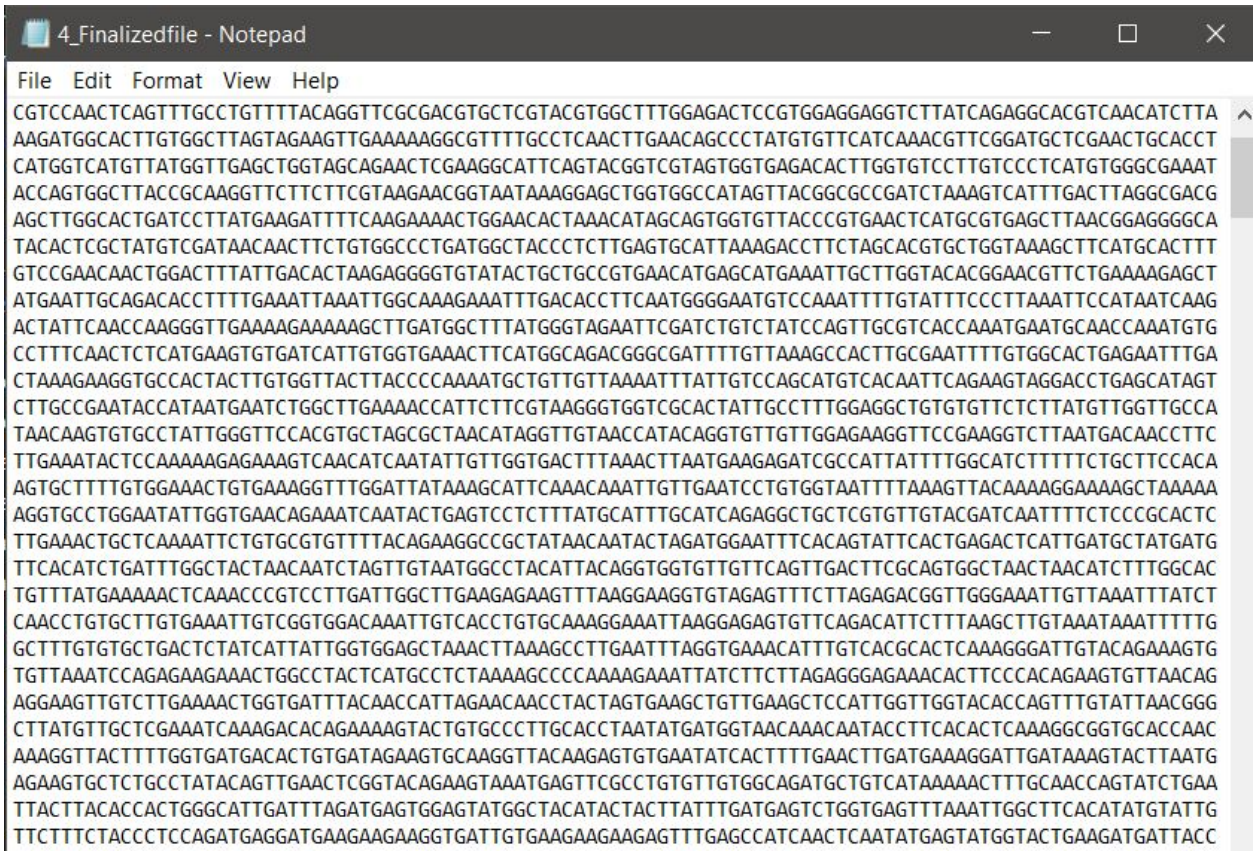
Compressed File





Decoded FIle






```
CGTCCAACTCAGTTTGCCTGTTTTACAGGTTCGCGACGTGCTCGTACGTGGCTTTGGAGACTCCGTGGAGGAGGTCTTATCAGAGGCACGTCAACATCTTA
AAGATGGCACTTGTGGCTTAGTAGAAGTTGAAAAAGGCGTTTTGCCTCAACTTGAACAGCCCTATGTGTTCATCAAACGTTCGGATGCTCGAACTGCACCT
CATGGTCATGTTATGGTTGAGCTGGTAGCAGAACTCGAAGGCATTCAGTACGGTCGTAGTGGTGAGACACTTGGTGTCCTTGTCCCTCATGTGGGCGAAAT
ACCAGTGGCTTACCGCAAGGTTCTTCTTCGTAAGAACGGTAATAAAGGAGCTGGTGGCCATAGTTACGGCGCCGATCTAAAGTCATTTGACTTAGGCGACG
AGCTTGGCACTGATCCTTATGAAGATTTTCAAGAAAACTGGAACACTAAACATAGCAGTGGTGTTACCCGTGAACTCATGCGTGAGCTTAACGGAGGGGCA
TACACTCGCTATGTCGATAACAACTTCTGTGGCCCTGATGGCTACCCTCTTGAGTGCATTAAAGACCTTCTAGCACGTGCTGGTAAAGCTTCATGCACTTT
GTCCGAACAACTGGACTTTATTGACACTAAGAGGGGTGTATACTGCTGCCGTGAACATGAGCATGAAATTGCTTGGTACACGGAACGTTCTGAAAAGAGCT
ATGAATTGCAGACACCTTTTGAAATTAAATTGGCAAAGAAATTTGACACCTTCAATGGGGAATGTCCAAATTTTGTATTTCCCTTAAATTCCATAATCAAG
ACTATTCAACCAAGGGTTGAAAAGAAAAAGCTTGATGGCTTTATGGGTAGAATTCGATCTGTCTATCCAGTTGCGTCACCAAATGAATGCAACCAAATGTG
CCTTTCAACTCTCATGAAGTGTGATCATTGTGGTGAAACTTCATGGCAGACGGGCGATTTTGTTAAAGCCACTTGCGAATTTTGTGGCACTGAGAATTTGA
CTAAAGAAGGTGCCACTACTTGTGGTTACTTACCCCAAAATGCTGTTGTTAAAATTTATTGTCCAGCATGTCACAATTCAGAAGTAGGACCTGAGCATAGT
CTTGCCGAATACCATAATGAATCTGGCTTGAAAACCATTCTTCGTAAGGGTGGTCGCACTATTGCCTTTGGAGGCTGTGTGTTCTCTTATGTTGGTTGCCA
TAACAAGTGTGCCTATTGGGTTCCACGTGCTAGCGCTAACATAGGTTGTAACCATACAGGTGTTGTTGGAGAAGGTTCCGAAGGTCTTAATGACAACCTTC
TTGAAATACTCCAAAAAGAGAAAGTCAACATCAATATTGTTGGTGACTTTAAACTTAATGAAGAGATCGCCATTATTTTGGCATCTTTTTCTGCTTCCACA
AGTGCTTTTGTGGAAACTGTGAAAGGTTTGGATTATAAAGCATTCAAACAAATTGTTGAATCCTGTGGTAATTTTAAAGTTACAAAAGGAAAAGCTAAAAA
AGGTGCCTGGAATATTGGTGAACAGAAATCAATACTGAGTCCTCTTTATGCATTTGCATCAGAGGCTGCTCGTGTTGTACGATCAATTTTCTCCCGCACTC
TTGAAACTGCTCAAAATTCTGTGCGTGTTTTACAGAAGGCCGCTATAACAATACTAGATGGAATTTCACAGTATTCACTGAGACTCATTGATGCTATGATG
TTCACATCTGATTTGGCTACTAACAATCTAGTTGTAATGGCCTACATTACAGGTGGTGTTGTTCAGTTGACTTCGCAGTGGCTAACTAACATCTTTGGCAC
TGTTTATGAAAAACTCAAACCCGTCCTTGATTGGCTTGAAGAGAAGTTTAAGGAAGGTGTAGAGTTTCTTAGAGACGGTTGGGAAATTGTTAAATTTATCT
CAACCTGTGCTTGTGAAATTGTCGGTGGACAAATTGTCACCTGTGCAAAGGAAATTAAGGAGAGTGTTCAGACATTCTTTAAGCTTGTAAATAAATTTTTG
GCTTTGTGTGCTGACTCTATCATTATTGGTGGAGCTAAACTTAAAGCCTTGAATTTAGGTGAAACATTTGTCACGCACTCAAAGGGATTGTACAGAAAGTG
TGTTAAATCCAGAGAAGAAACTGGCCTACTCATGCCTCTAAAAGCCCCAAAAGAAATTATCTTCTTAGAGGGAGAAACACTTCCCACAGAAGTGTTAACAG
AGGAAGTTGTCTTGAAAACTGGTGATTTACAACCATTAGAACAACCTACTAGTGAAGCTGTTGAAGCTCCATTGGTTGGTACACCAGTTTGTATTAACGGG
CTTATGTTGCTCGAAATCAAAGACACAGAAAAGTACTGTGCCCTTGCACCTAATATGATGGTAACAAACAATACCTTCACACTCAAAGGCGGTGCACCAAC
AAAGGTTACTTTTGGTGATGACACTGTGATAGAAGTGCAAGGTTACAAGAGTGTGAATATCACTTTTGAACTTGATGAAAGGATTGATAAAGTACTTAATG
AGAAGTGCTCTGCCTATACAGTTGAACTCGGTACAGAAGTAAATGAGTTCGCCTGTGTTGTGGCAGATGCTGTCATAAAAACTTTGCAACCAGTATCTGAA
TTACTTACACCACTGGGCATTGATTTAGATGAGTGGAGTATGGCTACATACTACTTATTTGATGAGTCTGGTGAGTTTAAATTGGCTTCACATATGTATTG
TTCTTTCTACCCTCCAGATGAGGATGAAGAAGAAGGTGATTGTGAAGAAGAAGAGTTTGAGCCATCAACTCAATATGAGTATGGTACTGAAGATGATTACC
```

Compression Ratio

Compression Ratio = UnCompressed SIze : Compressed Size

Compression Ratio = 30kb : 8kb

= 3.75

Frequency Table

Character	Frequency
C	5492
G	5863
A	8954
T	9594

Codeword Table

Character	Frequency
C	11
G	10
A	01
T	00