VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama, Macche, Santibastwada Road
Belagavi-590018, Karnataka

A
Project Report
On

Shannon-Fano Algorithm for Data Compression.

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering
In
Electronics & Communication Engineering
By

1DS19EC054: Impana Reddy
1DS19EC084: Nainshree Raj

Under the guidance
Of

Prof. Ravindra
Associate Professor, ECE Dept., DSCE, Bengaluru

Department of Electronics & Communication Engineering
(An Autonomous College affiliated to VTU Belgaum, accredited by NBA & NAAC)
Shavige Malleshwara Hills, Kumaraswamy Layout,
Banashankari, Bengaluru-560078, Karnataka, India
2021-22

Mini project Report Declaration

Certified that the UG Mini project entitled, "Shannon-Fano Algorithm for Data Compression"has been submitted as AAT for the subject DIGITAL COMMUNICATION SYSTEM is a Bonafide work that is carried out by myself in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2021-22. We are solely responsible for all the contents that has have been presented in it.

Student sign

Impana Reddy
Nainshree Raj

USN: .1DS19EC054
USN: .1DS19EC084

Date: 07 /01 / 2022

Place: Bengaluru -78

_____

Mini project Guide
Name & Signature
Prof. Ravindra

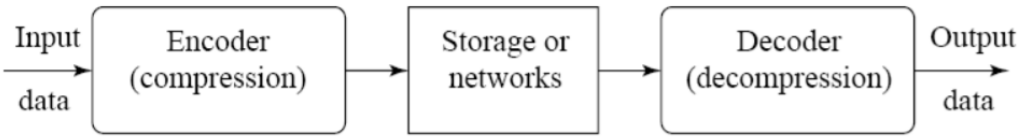
Introduction

The main aim of this report is to Shannon-Fano Algorithm for Data Compression.

Shannon Fano Algorithm is an entropy encoding technique for lossless data compression of multimedia. Named after Claude Shannon and Robert Fano, it assigns a code to each symbol based on their probabilities of occurrence. It is a variable-length encoding scheme, that is, the codes assigned to the symbols will be of varying length.

DATA COMPRESSION AND ITS TYPES

Data Compression, also known as source coding, is the process of encoding or converting data in such a way that it consumes less memory space. Data compression reduces the number of resources required to store and transmit data. Compression: the process of coding that will effectively reduce the total number of bits needed to represent certain information.



WHY COMPRESSION?

Multimedia data are too big ☹ "A picture is worth a thousand words ! "
File Sizes for a One-minute QCIF Video Clip

Frame Rate	Frame Size	Bits / pixel	Bit-rate (bps)	File Size (Bytes)
30 frames/sec	176 x 144 pixels	12	9,123,840	68,428,800

It can be done in two ways- lossless compression and lossy compression. Lossy compression reduces the size of data by removing unnecessary information, while there is no data loss in lossless compression.

HOW DOES SHANNON-FANO ALGORITHM WORK?

The steps of the algorithm are as follows:

1. Create a list of probabilities or frequency counts for the given set of symbols so that the relative frequency of occurrence of each symbol is known.
2. Sort the list of symbols in decreasing order of probability, the most probable ones to the left and least probable to the right.
3. Split the list into two parts, with the total probability of both the parts being as close to each other as possible.
4. Assign the value 0 to the left part and 1 to the right part.
5. Repeat steps 3 and 4 for each part, until all the symbols are split into individual subgroups.

The Shannon codes are considered accurate if the code of each symbol is unique.

HOW DOES IT WORK?

The steps of the algorithm are as follows:

1. Create a list of probabilities or frequency counts for the given set of symbols so that the relative frequency of occurrence of each symbol is known.
2. Sort the list of symbols in decreasing order of probability, the most probable ones to the left and least probable to the right.
3. Split the list into two parts, with the total probability of both the parts being as close to each other as possible.
4. Assign the value 0 to the left part and 1 to the right part.
5. Repeat steps 3 and 4 for each part, until all the symbols are split into individual subgroups.

The Shannon codes are considered accurate if the code of each symbol is unique.

EXAMPLE:

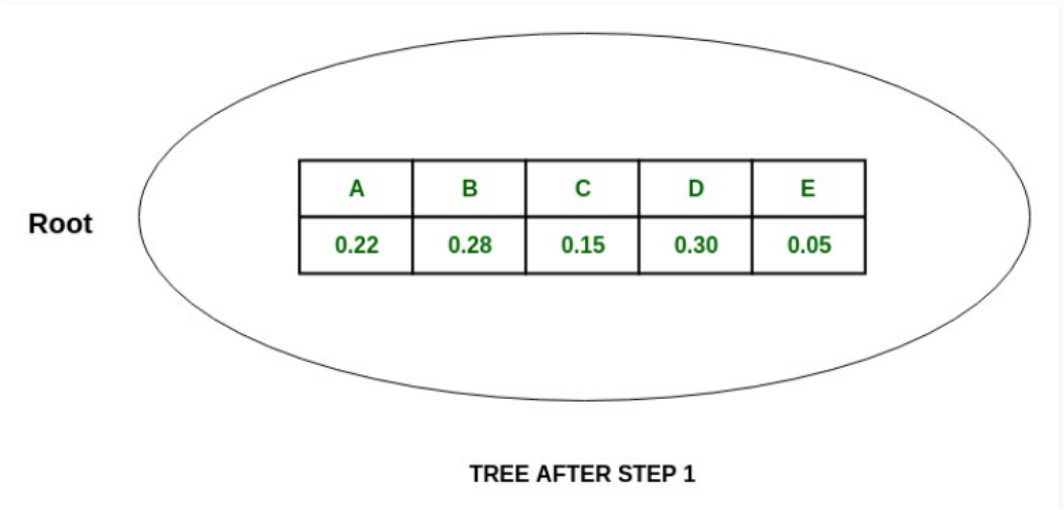
The given task is to construct Shannon codes for the given set of symbols using the Shannon-Fano lossless compression technique.

Step:

SYMBOL	A	B	C	D	E
PROBABILITY OR FRQUENCY	0.22	0.28	0.15	0.30	0.05

THE SYMBOLS AND THEIR PROBABILITY / FREQUENCY ARE TAKEN AS INPUTS.
(In case of Frequency, the values can be any number)

Tree:



TREE AFTER STEP 1

Solution

1. Upon arranging the symbols in decreasing order of probability:

   P(D)+P(B)=0.3+0.28=0.58

   P(A) + P(C) + P(E) = 0.22 + 0.15 + 0.05 = 0.42

   And since the almost equally split the table, the most is divided it the blockquote table isblockquotento

   {D, B} **and** {A, C, E}

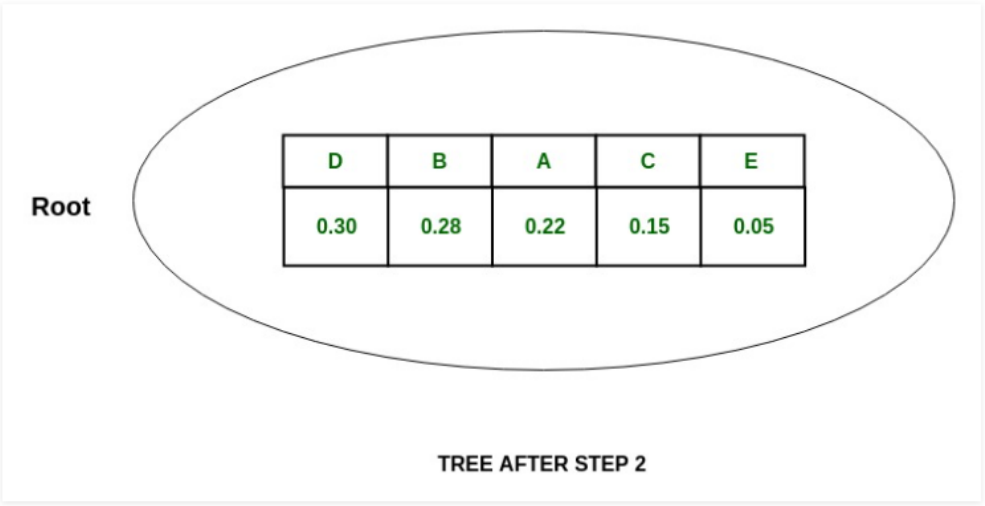
and assign them the values 0 and 1 respectively.

Step2:

SYMBOL	D	B	A	C	E
PROBABILITY OR FRQUENCY	0.30	0.28	0.22	0.15	0.05

INPUTS ARE SORTED ACCORDING TO THEIR PROBABILITY / FREQUENCY (Here they are sorted according to their probability)

Tree:



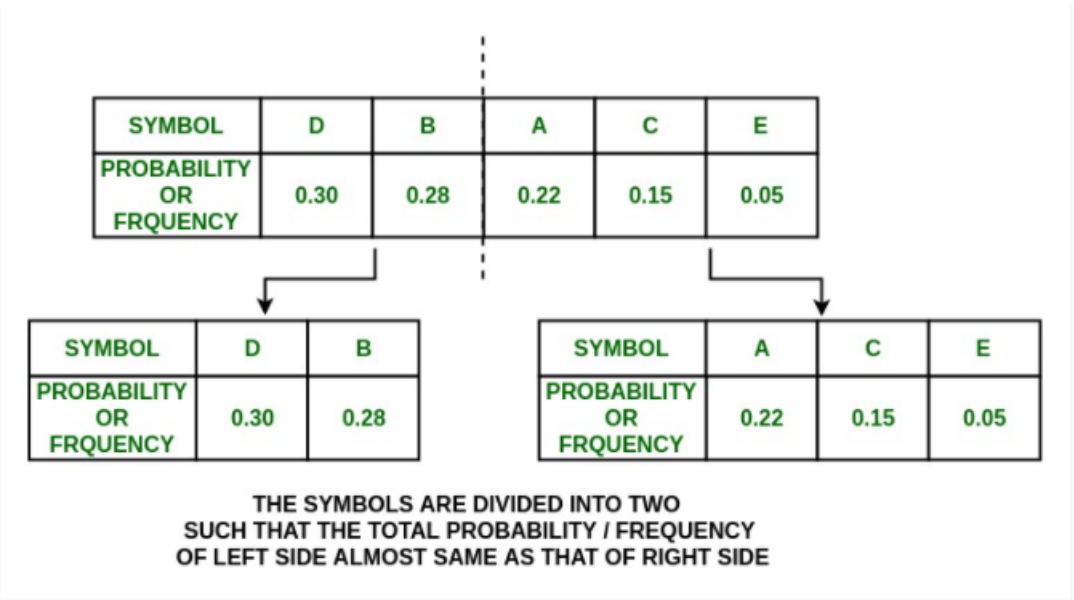
TREE AFTER STEP 2

2. Now, in {D, B} group,

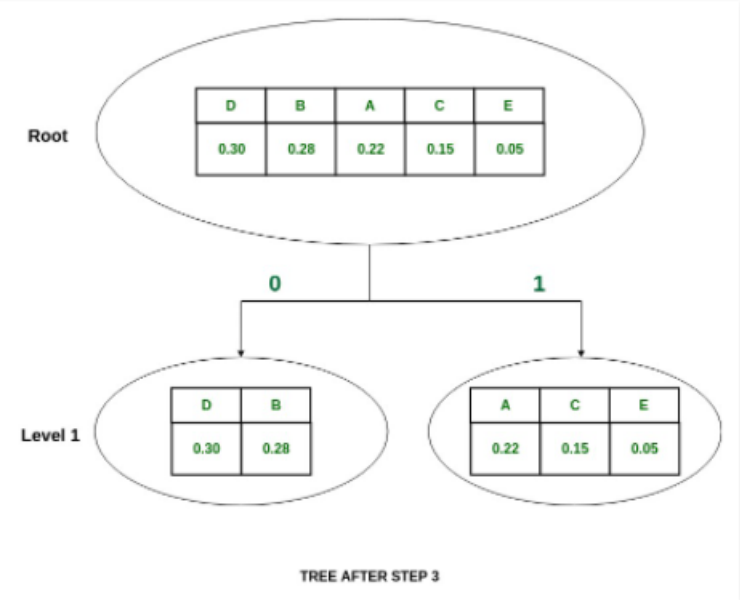
P(D) = 0.30 *and* ***P(B) = 0.28***

which means that **P(D)~P(B)**, so divide {D, B} into {D} and {B} and assign 0 to D and 1 to B.

Step:



Tree:



3. In {A, C, E} group,

P(A) = 0.22 *and* ***P(C) + P(E) = 0.20***

So the group is divided into

{A} *and* ***{C, E}***

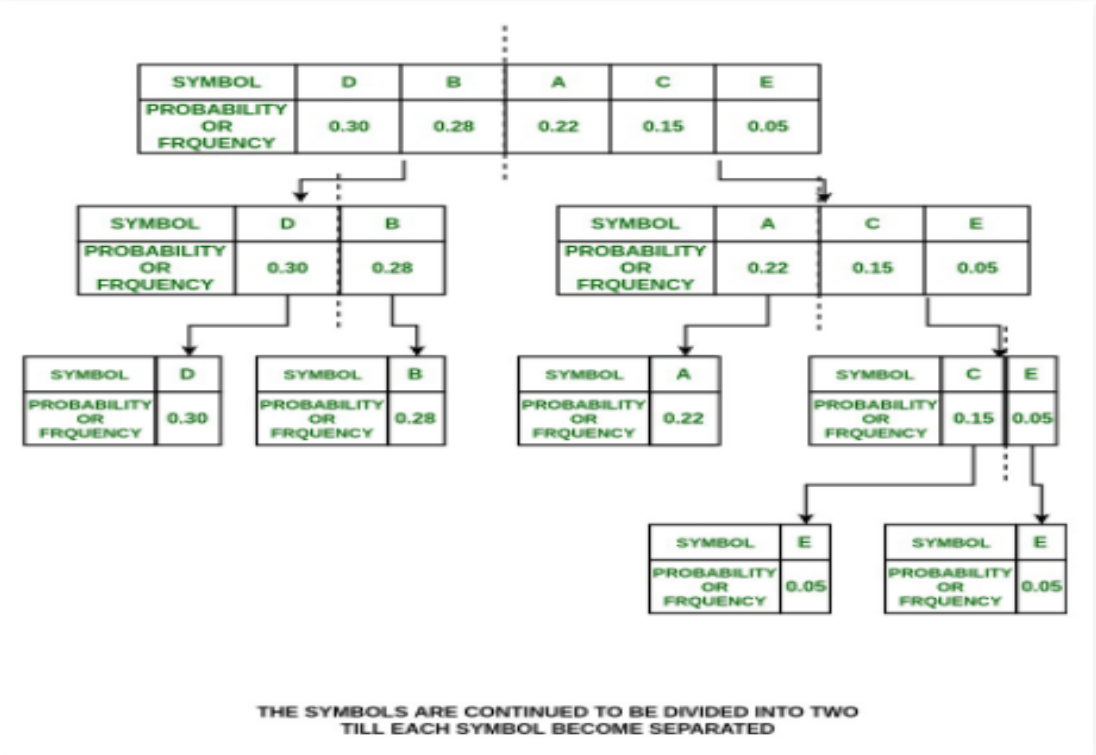
A nd they are assigned values 0 and 1 respectively.

4. In {C, E} group,

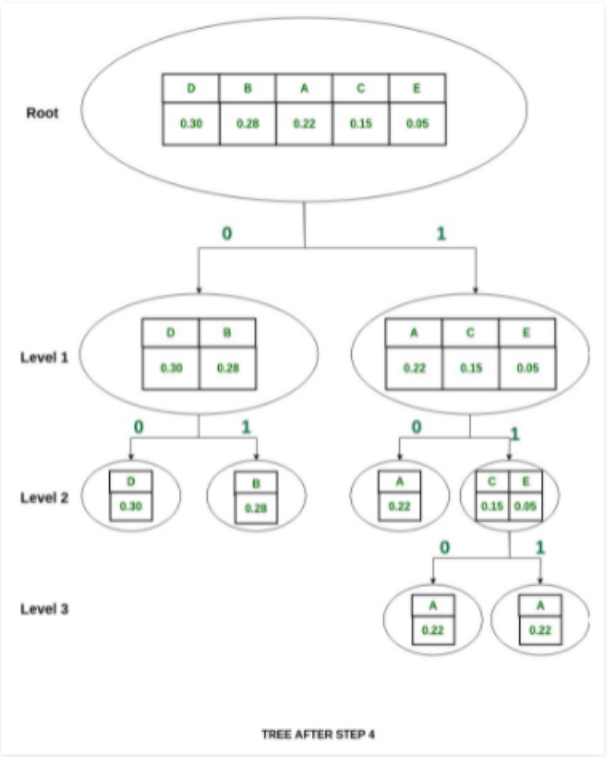
P(C) = 0.15** and **P(E) = 0.05

So divide them into {C} and {E} and assign 0 to {C} and 1 to {E}

Step:



Tree:



TREE AFTER STEP 4

Note: The splitting is now stopped as each symbol is separated now.
The Shannon codes for the set of symbols are:

SYMBOL	A	B	C	D	E
PROBABILITY OR FRQUENCY	0.22	0.28	0.15	0.30	0.05
SHANNON-FANO CODE	00	01	10	110	111

THE SHANNON CODES OF THE SYMBOLS
(based on their traversal from the root node)

As it can be seen, these are all unique and of varying lengths.

Program:

```
// C++ program for Shannon Fano Algorithm
// include header files

#include <bits/stdc++.h>
using namespace std;

// declare structure node
struct node {
   // for storing symbol
   string sym;
   // for storing probability or frequency
   float pro;
   int arr[20];
   int top;
} p[20];

typedef struct node node;

// function to find shannon code

void shannon(int l, int h, node p[])
{
   float pack1 = 0, pack2 = 0, diff1 = 0, diff2 = 0;
   int i, d, k, j;
   if ((l + 1) == h || l == h || l > h)
   {
      if (l == h || l > h)
         return;
      p[h].arr[++(p[h].top)] = 0;
      p[l].arr[++(p[l].top)] = 1;
      return;
   }
   else
   {
      for (i = l; i <= h - 1; i++)
         pack1 = pack1 + p[i].pro;
      pack2 = pack2 + p[h].pro;
      diff1 = pack1 - pack2;
      if (diff1 < 0)
         diff1 = diff1 * -1;
      j = 2;
      while (j != h - l + 1)
```

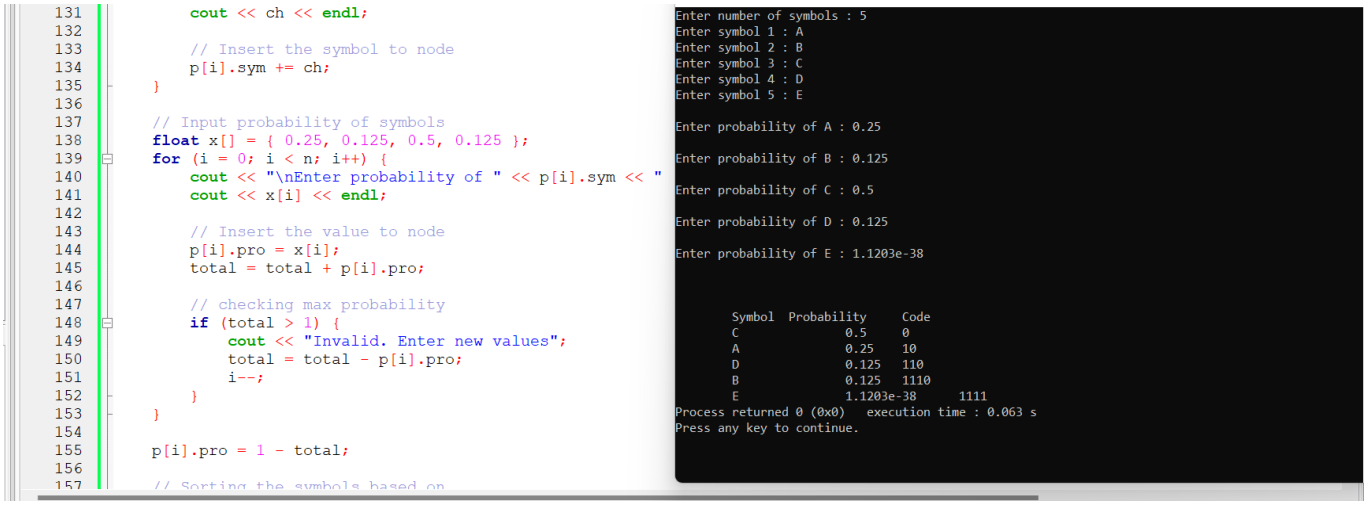
```
        {
            k = h - j;
            pack1 = pack2 = 0;
            for (i = l; i <= k; i++)
                pack1 = pack1 + p[i].pro;

            for (i = h; i > k; i--)
                pack2 = pack2 + p[i].pro;

            diff2 = pack1 - pack2;

            if (diff2 < 0)
                diff2 = diff2 * -1;
            if (diff2 >= diff1)
                break;
            diff1 = diff2;
            j++;
        }
        k++;
        for (i = l; i <= k; i++)
            p[i].arr[++(p[i].top)] = 1;
        for (i = k + 1; i <= h; i++)
            p[i].arr[++(p[i].top)] = 0;

        // Invoke shannon function
        shannon(l, k, p);
        shannon(k + 1, h, p);
    }
}

// Function to sort the symbols
// based on their probability or frequency
void sortByProbability(int n, node p[])
{
    int i, j;
    node temp;
    for (j = 1; j <= n - 1; j++) {
        for (i = 0; i < n - 1; i++) {
            if ((p[i].pro) > (p[i + 1].pro)) {
                temp.pro = p[i].pro;
                temp.sym = p[i].sym;
```

```
            p[i].pro = p[i + 1].pro;
            p[i].sym = p[i + 1].sym;

            p[i + 1].pro = temp.pro;
            p[i + 1].sym = temp.sym;
         }
      }
   }
}

// function to display shannon codes
void display(int n, node p[])
{
   int i, j;
   cout << "\n\n\n\tSymbol\tProbability\tCode";
   for (i = n - 1; i >= 0; i--) {
      cout << "\n\t" << p[i].sym << "\t\t" << p[i].pro << "\t";
      for (j = 0; j <= p[i].top; j++)
         cout << p[i].arr[j];
   }
}

// Driver code
int main()
{
   int n, i, j;
   float total = 0;
   string ch;
   node temp;

   // Input number of symbols
   cout << "Enter number of symbols\t: ";
   n = 5;
   cout << n << endl;

   // Input symbols
   for (i = 0; i < n; i++) {
      cout << "Enter symbol " << i + 1 << " : ";
      ch = (char)(65 + i);
      cout << ch << endl;

      // Insert the symbol to node
```

```
    p[i].sym += ch;
}

  // Input probability of symbols
 //ss=[0.25 0.125 0.5 0.125];
 //ss=[0.25 0.125 0.0625 0.0625 0.0625 0.25 0.0625 0.125];
 //ss=[0.4 0.2 0.12 0.08 0.08 0.08 0.04]
 //ss=[0.4 0.3 0.2 0.1]
 //ss=[0.45 0.15 0.1 0.1 0.08 0.08 0.04]


float x[] = { 0.4, 0.2, 0.12, 0.08, 0.08, 0.08, 0.04 };
for (i = 0; i < n; i++)
{
    cout << "\nEnter probability of " << p[i].sym << " : ";
    cout << x[i] << endl;

    // Insert the value to node
    p[i].pro = x[i];
    total = total + p[i].pro;

    // checking max probability
    if (total > 1) {
        cout << "Invalid. Enter new values";
        total = total - p[i].pro;
        i--;
    }
}

p[i].pro = 1 - total;

// Sorting the symbols based on
// their probability or frequency
sortByProbability(n, p);

for (i = 0; i < n; i++)
    p[i].top = -1;

// Find the shannon code
shannon(0, n - 1, p);

// Display the codes
```

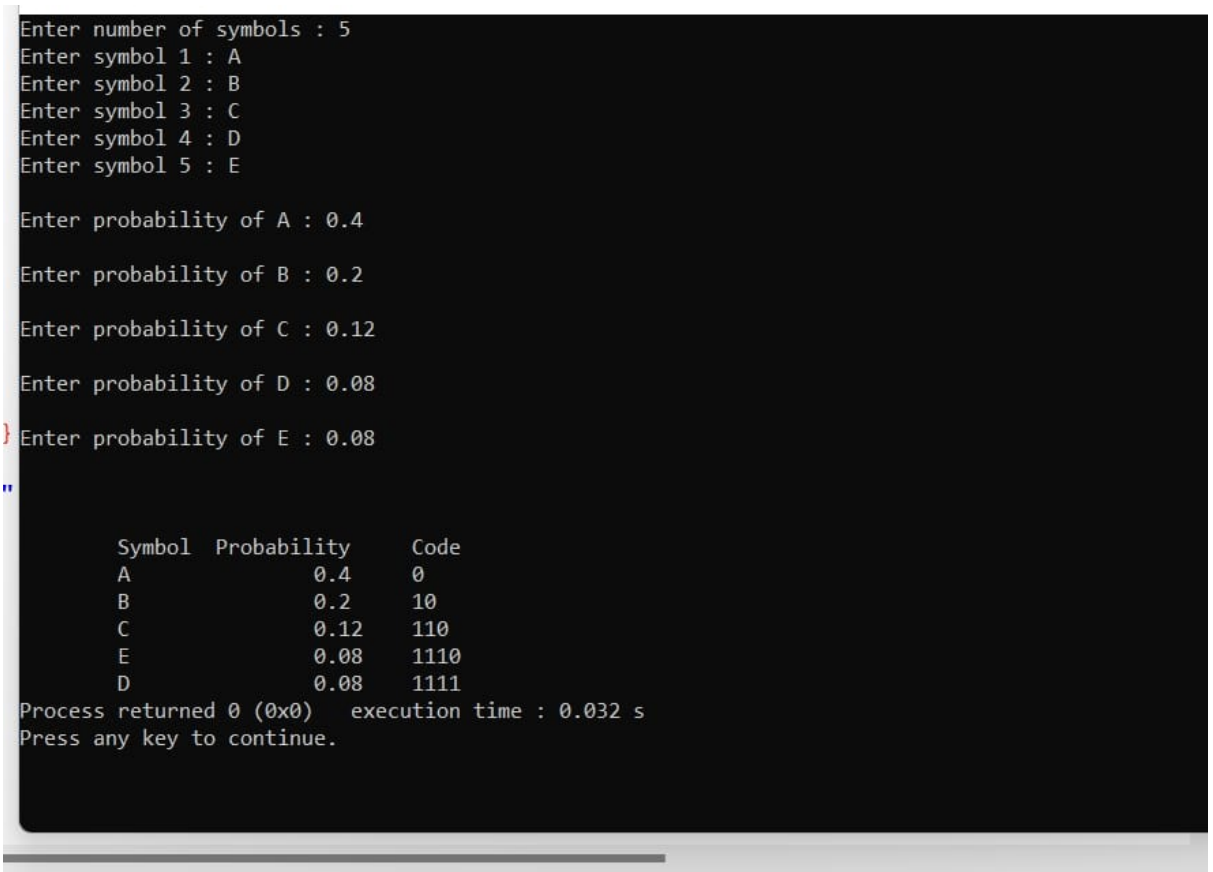
```
    display(n, p);
    return 0;
}
```

Output:



```
        cout << ch << endl;

        // Insert the symbol to node
        p[i].sym += ch;
    }

    // Input probability of symbols
    float x[] = { 0.25, 0.125, 0.5, 0.125 };
    for (i = 0; i < n; i++) {
        cout << "\nEnter probability of " << p[i].sym << "
        cout << x[i] << endl;

        // Insert the value to node
        p[i].pro = x[i];
        total = total + p[i].pro;

        // checking max probability
        if (total > 1) {
            cout << "Invalid. Enter new values";
            total = total - p[i].pro;
            i--;
        }
    }

    p[i].pro = 1 - total;

    // Sorting the symbols based on
```

```
Enter number of symbols : 5
Enter symbol 1 : A
Enter symbol 2 : B
Enter symbol 3 : C
Enter symbol 4 : D
Enter symbol 5 : E

Enter probability of A : 0.25

Enter probability of B : 0.125

Enter probability of C : 0.5

Enter probability of D : 0.125

Enter probability of E : 1.1203e-38



       Symbol  Probability     Code
       C               0.5     0
       A               0.25    10
       D               0.125   110
       B               0.125   1110
       E               1.1203e-38      1111
Process returned 0 (0x0)   execution time : 0.063 s
Press any key to continue.
```



```
Enter number of symbols : 5
Enter symbol 1 : A
Enter symbol 2 : B
Enter symbol 3 : C
Enter symbol 4 : D
Enter symbol 5 : E

Enter probability of A : 0.4

Enter probability of B : 0.2

Enter probability of C : 0.12

Enter probability of D : 0.08

Enter probability of E : 0.08


       Symbol  Probability     Code
       A               0.4     0
       B               0.2     10
       C               0.12    110
       E               0.08    1110
       D               0.08    1111
Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

Application of Shannon-Fano Algorithm for Data Compression.

- They are used in computer memory, modems and embedded processors.
- They are used in Nano Satellites.

Advantages of Shannon-Fano Algorithm for Data Compression.

- They are effectively used to detect and correct errors**.**
- For Shannon Fano coding procedure we do not need to build the entire codebook instead, we simply obtain the code for the tag corresponding to a given sequence. It is entirely feasible to code sequences of length 20 or more.

Disadvantages of Shannon-Fano Algorithm for Data Compression.

1) In Shannon-Fano coding, we cannot be sure about the codes generated. There may be two different codes for the same symbol depending on the way we build our tree.

2) Also, here we have no unique code i.e a code might be a prefix for another code. So in case of errors or loss during data transmission, we have to start from the beginning.

3) Shannon-Fano coding does not guarantee optimal codes.