HUFFMAN CODING

Name:-Rajul Gupta.

Roll no:-307

Batch-C-1

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<math.h>

#include<stdlib.h>

#include<alloc.h>

typedef struct tree

{

int len;

float data;

struct tree \*next;

struct tree \*prev;

struct tree \*l;

struct tree \*h;

char cod[20];

}node;

int n;

float p[20],add;

node \*stk[40];

int top=-1;

node \*new1,\*temp,\*head;

/\* Stack functions \*/

int empty()

{

if(top==-1)

return(1);

return(0);

}

void push(node \*temp)

{

top++;

stk[top]=temp;

}

node\* pop()

{

node \*temp;

if(!empty())

{

temp=stk[top];

top--;

}

return(temp);

}

/\* Sorts the probabilities using bubble sort\*/

void sort()

{

int i,j;

float temp;

for(i=0;i<n;i++)

for(j=i+1;j<n;j++)

{

if(p[i]<p[j])

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

/\* Creates the linked list of above sorted probabilties \*/

void create()

{

int i,flag=0;

new1=NULL;

for(i=0;i<n;i++)

{

new1=(node\*)malloc(sizeof(node));

new1->next=NULL;

new1->prev=NULL;

new1->l=NULL;

new1->h=NULL;

strcpy(new1->cod,"");

new1->len=0;

new1->data=p[i];

if(flag==0)

{

head=new1;

temp=new1;

flag=1;

}

else

{

temp->next=new1;

new1->prev=temp;

temp=new1;

}

}

}

/\*forms the tree \*/

void place(node \*sum)

{

int s=0;

for(temp=head;temp!=NULL;temp=temp->next)

{

if(head->data<=sum->data)

{

sum->next=head;

head->prev=sum;

head=sum;

s=1;

}

else

{

if(temp->next->data<=sum->data)

{

sum->next=temp->next;

temp->next->prev=sum;

temp->next=sum;

sum->prev=temp;

s=1;

}

if(temp->next==NULL)

{

temp->next=sum;

sum->prev=temp;

temp=temp->next;

sum->next=NULL;

s=1;

}

}

if(s==1)

break;

}

}

void tree()

{

node \*sum;

while(head->next!=NULL)

{

for(temp=head;temp->next->next!=NULL;temp=temp->next);

sum=(node\*)malloc(sizeof(node));

sum->next=NULL;

sum->prev=NULL;

strcpy(sum->cod,"");

sum->len=0;

sum->data=temp->data+temp->next->data;

sum->l=temp;

sum->h=temp->next;

if(temp!=head)

{

temp->next=NULL;

temp->prev->next=NULL;

temp->prev=NULL;

place(sum);

}

else

head=sum;

}

}

/\* Coding \*/

void code()

{

node \*x;

add=0;

temp=head;

printf("\n\nCODES ARE AS FOLLOWS........");

printf("\n\nSymbol probability -> Symbol code");

do

{

while(temp!=NULL)

{

push(temp);

x=temp;

temp=temp->l;

if(temp!=NULL)

{

strcpy(temp->cod,x->cod);

strcat(temp->cod,"0");

temp->len=strlen(temp->cod);

}

}

temp=pop();

if(temp->l==NULL && temp->h==NULL)

{

printf("\n\n\t%f\t-> %s ",temp->data,temp->cod);

add=add+((temp->data)\*(temp->len));

}

x=temp;

temp=temp->h;

if(temp!=NULL)

{

strcpy(temp->cod,x->cod);

strcat(temp->cod,"1");

temp->len=strlen(temp->cod);

}

}while(temp!=NULL || !empty());

}

float entropy(float p[])

{

int i;

float sum=0;

for(i=0;i<n;i++)

sum=sum+(-p[i]\*log10(p[i])/log10(2));

return(sum);

}

void main()

{

float e;

int i;

clrscr();

printf("\n\nEnter no. of source samples:");

scanf("%d",&n);

printf("\n\nEnter the probabilities of source samples:\n");

for(i=0;i<n;i++)

{

printf("\nFor p[%d]:\t",i);

scanf("%f",&p[i]);

}

sort();

create();

tree();

code();

e=entropy(p);

printf("\n\nThe entropy is :%f bits",e);

printf("\n\nAverage code length is:%f bits",add);

printf("\n\nThe coding efficiency is:%f %",(e/add\*100));

getch();

}

OUTPUT:

Enter no. of source samples:5

Enter the probabilities of source samples:

For p[0]: 0.4

For p[1]: 0.2

For p[2]: 0.2

For p[3]: 0.1

For p[4]: 0.1

CODES ARE AS FOLLOWS........

Symbol probability -> Symbol code

0.400000 -> 00

0.100000 -> 010

0.100000 -> 011

0.200000 -> 10

0.200000 -> 11

The entropy is :2.121928 bits

Average code length is:2.200000 bits

The coding efficiency is:96.451270 %