哈夫曼树

算法

BiTree CreateHufferman(int num[], int n) //构建哈夫曼树

{

BiTree \*tmp,p;

int k1,k2;

int i,j;

tmp = (BiTree \*)malloc(sizeof(BiTree) \* n); //tmp数组用来存储数值

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

{

tmp[i] = (BiTree)malloc(sizeof(BiTNode));

if (!tmp[i])

{

printf("内存分配失败");

exit(0);

}

tmp[i]->weight = num[i];

tmp[i]->lchild = tmp[i]->rchild = NULL; //叶子节点无左右孩子

}

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

{

k1 = -1;

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

{

if (tmp[j]!=NULL && k1==-1)

{

k1 = j;

continue;

}

if (tmp[j]!=NULL)

{

k2 = j;

break;

}

}

for (j=1; j<n; j++) //此段循环找出tmp数组中权值最小和次小的下标

{

if (tmp[j]!=NULL)

{

if (tmp[j]->weight < tmp[k1]->weight)

{

k2 = k1;

k1 = j; //kl指向最小下标

}

else if (tmp[j]->weight < tmp[k2]->weight)

{

k2 = j; // k2指向次小下标

}

}

}

//合并k1,k2

p = (BiTNode \*)malloc(sizeof(BiTNode));

if (!p)

exit(0);

p->weight = tmp[k1]->weight + tmp[k2]->weight;

p->lchild = tmp[k1];

p->rchild = tmp[k2];

tmp[k1] = p; // 新结点的放到k1的位置

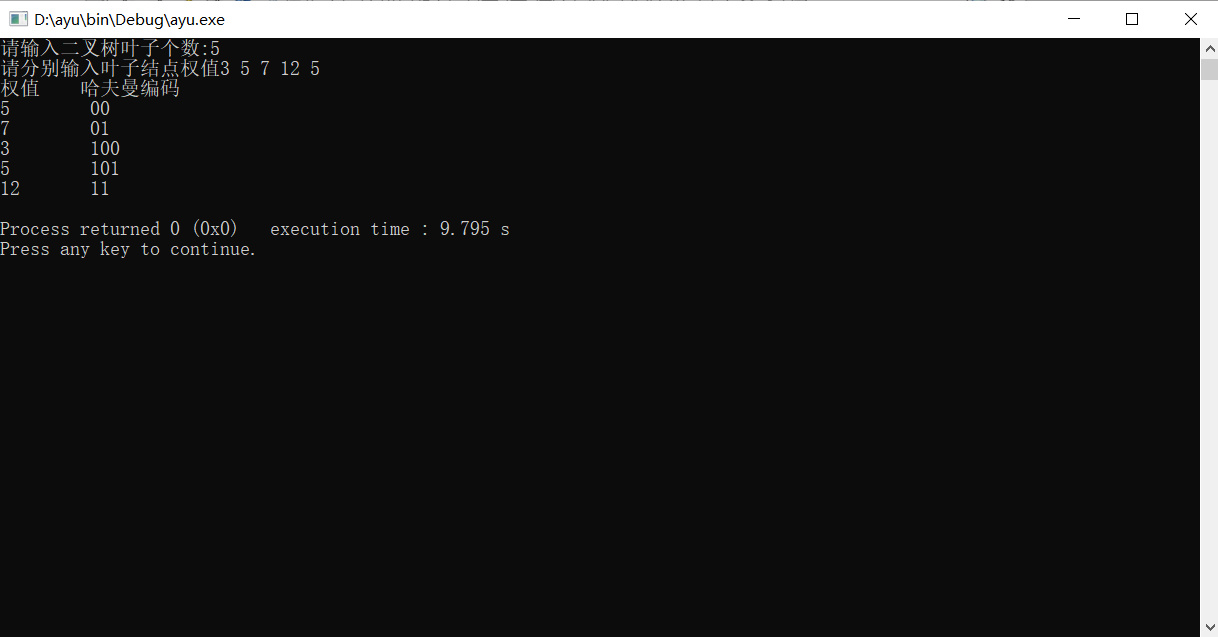
tmp[k2] = NULL; // k2位置为空

}

return p; //返回根

}

运行截图



源代码

#include<stdio.h>

#include<stdlib.h>

typedef struct node

{

int weight;

struct node \*lchild,\*rchild;

} BiTNode, \*BiTree;

BiTree CreateHufferman(int num[], int n) //构建哈夫曼树

{

BiTree \*tmp,p;

int k1,k2;

int i,j;

tmp = (BiTree \*)malloc(sizeof(BiTree) \* n); //tmp数组用来存储数值

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

{

tmp[i] = (BiTree)malloc(sizeof(BiTNode));

if (!tmp[i])

{

printf("内存分配失败");

exit(0);

}

tmp[i]->weight = num[i];

tmp[i]->lchild = tmp[i]->rchild = NULL; //叶子节点无左右孩子

}

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

{

k1 = -1;

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

{

if (tmp[j]!=NULL && k1==-1)

{

k1 = j;

continue;

}

if (tmp[j]!=NULL)

{

k2 = j;

break;

}

}

for (j=1; j<n; j++) //此段循环找出tmp数组中权值最小和次小的下标

{

if (tmp[j]!=NULL)

{

if (tmp[j]->weight < tmp[k1]->weight)

{

k2 = k1;

k1 = j; //kl指向最小下标

}

else if (tmp[j]->weight < tmp[k2]->weight)

{

k2 = j; // k2指向次小下标

}

}

}

//合并k1,k2

p = (BiTNode \*)malloc(sizeof(BiTNode));

if (!p)

exit(0);

p->weight = tmp[k1]->weight + tmp[k2]->weight;

p->lchild = tmp[k1];

p->rchild = tmp[k2];

tmp[k1] = p; // 新结点的放到k1的位置

tmp[k2] = NULL; // k2位置为空

}

return p; //返回根

}

void GetCode(BiTree T, int pathLen)

{

static char tmp[199];

int i;

if (T->lchild==NULL && T->rchild==NULL)

{

printf("\n%d\t", T->weight );

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

{

printf("%c", tmp[i]);

}

return ;

}

//往左走

tmp[pathLen] = '0';

GetCode(T->lchild, pathLen+1);

//往右走

tmp[pathLen] = '1';

GetCode(T->rchild, pathLen+1);

}

int main()

{

int n,num[100];

BiTree HfmTree;

int i;

printf("请输入二叉树叶子个数:");

scanf("%d",&n);

printf("请分别输入叶子结点权值");

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

{

scanf("%d", &num[i]);

}

HfmTree = CreateHufferman(num, n);

printf("权值\t哈夫曼编码");

GetCode(HfmTree, 1);

printf("\n");

return 0;

}