哈夫曼树的构建及哈夫曼编码及译码

班级：电信1808 姓名：蔡浩晖 学号：U201813453 完成日期：2019年4月26日星期五

1. 需求分析：
2. 输入形式及输入范围：输入的N值不大于Maxsize，并且一起输入N个字符及其权重（出现频率）
3. 输出形式：首先输出各个结点的权重，双亲编号及左孩子与右孩子编号
4. 程序所达到的功能：对一组字符进行哈夫曼编码，并且可以对一串字符进行译码
5. 测试数据：27

pa[0].ch = ' ',w[0] = 186;

pa[1].ch = 'A',w[1] = 64;

pa[2].ch = 'B',w[2] = 13;

pa[3].ch = 'C',w[3] = 22;

pa[4].ch = 'D',w[4] = 32;

pa[5].ch = 'E',w[5] = 103;

pa[6].ch = 'F',w[6] = 21

pa[7].ch = 'G',w[7] = 15;

pa[8].ch = 'H',w[8] = 47;

pa[9].ch = 'I',w[9] = 57;

pa[10].ch = 'J',w[10] = 1;

pa[11].ch = 'K',w[11] = 5;

pa[12].ch = 'L',w[12] = 32;

pa[13].ch = 'M',w[13] = 20;

pa[14].ch = 'N',w[14] = 57;

pa[15].ch = 'O',w[15] = 63;

pa[16].ch = 'P',w[16] = 15;

pa[17].ch = 'Q',w[17] = 1;

pa[18].ch = 'R',w[18] = 48;

pa[19].ch = 'S',w[19] = 51;

pa[20].ch = 'T',w[20] = 80;

pa[21].ch = 'U',w[21] = 23;

pa[22].ch = 'V',w[22] = 8;

pa[23].ch = 'W',w[23] = 18;

pa[24].ch = 'X',w[24] = 1;

pa[25].ch = 'Y',w[25] = 16;

pa[26].ch = 'Z',w[26] = 1;

输出：186 52 0 0

64 44 0 0

13 31 0 0

22 36 0 0

32 38 0 0

103 47 0 0

21 35 0 0

15 32 0 0

47 40 0 0

57 42 0 0

1 27 0 0

5 29 0 0

32 38 0 0

20 34 0 0

57 43 0 0

63 43 0 0

15 33 0 0

1 27 0 0

48 41 0 0

51 41 0 0

80 45 0 0

23 36 0 0

8 30 0 0

18 34 0 0

1 28 0 0

16 33 0 0

1 28 0 0

2 29 10 17

2 30 24 26

7 31 27 11

10 32 28 22

20 35 29 2

25 37 30 7

31 37 16 25

38 39 23 13

41 39 31 6

45 40 3 21

56 42 32 33

64 44 4 12

79 45 34 35

92 46 36 8

99 46 18 19

113 47 37 9

120 48 14 15

128 48 1 38

159 49 39 20

191 50 40 41

216 50 5 42

248 51 43 44

345 51 45 0

407 52 46 47

593 0 48 49

593 0 50 0

1

A010

B100101

C00000

D0110

E010

F10011

G011001

H0001

I0111

J10010000

K1001001

L0111

M10001

N000

O001

P011010

Q10010001

R0010

S0011

T101

U00001

V0110001

W10000

X01100000

Y011011

Z01100001

1. 概要设计
2. 抽象数据类型

Hftree(pHftree t)

T = (pHftree)malloc(sizeof(Node));

T->lchild = NULL;

T->rchild = NULL;

1. 主函数流程：

先输入数据存入相应的数组，再调用HFtree函数，生成哈夫曼树及编码。

1. 详细设计

void select(int \*s1, int \*s2, pHFtree Hf, int n)

//选择此时权重最小的两个结点

int \_min = 9999;

int i,L1=0,L2=0;

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

{

if(Hf[i].weight<\_min&&Hf[i].parent==0)

{

\_min = Hf[i].weight;

L1 = i;

}

}

\*s1 = L1;

\_min = 9999;

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

{

if(Hf[i].weight<\_min&&(i!=\*s1)&&Hf[i].parent==0)

{

\_min = Hf[i].weight;

L2 = i;

}

}

\*s2 = L2;

}

void createHftree(pHFtree Hf, int \*w, int n)

//建立哈夫曼树并编码

int i;

int s1,s2;

int m = 2\*n - 1;

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

{

Hf[i].weight = w[i];

Hf[i].parent = 0;

Hf[i].Lchild = 0;

Hf[i].Rchild = 0;

}

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

{

Hf[i].weight = 0;

Hf[i].parent = 0;

Hf[i].Lchild = 0;

Hf[i].Rchild = 0;

}

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

{

select(&s1,&s2,Hf,i-1);

Hf[i].weight = Hf[s1].weight+Hf[s2].weight;

Hf[s1].parent = i;

Hf[s2].parent = i;

Hf[i].Lchild = s1;

Hf[i].Rchild = s2;

}

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

{

printf("%d %d %d %d\n",Hf[i].weight,Hf[i].parent,Hf[i].Lchild,Hf[i].Rchild);

}

}

void showHfcode(pHFtree Hf,int n,struct word \* pa)

//显示各个结点的信息

void showHfcode(pHFtree Hf,int n,struct word \* pa)

{

int start;

int i,j,k,t;

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

{

start = 19;

k = i;

t = Hf[k].parent;

while(t!=0)

{

if(Hf[t].Lchild == k)

pa[i].code[start--] = '0';

else

pa[i].code[start--] = '1';

k = t;

t = Hf[t].parent;

}

pa[i].codelength = 19-start;

}

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

{

printf("%c",pa[i].ch);

for(j=20-pa[i].codelength;j<20;j++)

printf("%c",pa[i].code[j]);

printf("\n");

}

}

void Decoding(char c,int n,struct word\*pa)

//对输入的字符串进行译码

void Decoding(char c,int n,struct word\*pa)

{

int j;

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

{

if(c ==pa[i].ch)

{

for(int j=20-pa[i].codelength;j<20;j++)

printf("%c",pa[i].code[j]);

}

}

}

1. 调试分析
2. 编码时从后往前录入，输出code数组中的编码时较困难，此时引入codelength变量记录编码长度
3. 该主程序的时间复杂度是O（n^2）,空间复杂度为O（n）
4. 测试结果

输入：

pa[0].ch = ' ',w[0] = 186;

pa[1].ch = 'A',w[1] = 64;

pa[2].ch = 'B',w[2] = 13;

pa[3].ch = 'C',w[3] = 22;

pa[4].ch = 'D',w[4] = 32;

pa[5].ch = 'E',w[5] = 103;

pa[6].ch = 'F',w[6] = 21

pa[7].ch = 'G',w[7] = 15;

pa[8].ch = 'H',w[8] = 47;

pa[9].ch = 'I',w[9] = 57;

pa[10].ch = 'J',w[10] = 1;

pa[11].ch = 'K',w[11] = 5;

pa[12].ch = 'L',w[12] = 32;

pa[13].ch = 'M',w[13] = 20;

pa[14].ch = 'N',w[14] = 57;

pa[15].ch = 'O',w[15] = 63;

pa[16].ch = 'P',w[16] = 15;

pa[17].ch = 'Q',w[17] = 1;

pa[18].ch = 'R',w[18] = 48;

pa[19].ch = 'S',w[19] = 51;

pa[20].ch = 'T',w[20] = 80;

pa[21].ch = 'U',w[21] = 23;

pa[22].ch = 'V',w[22] = 8;

pa[23].ch = 'W',w[23] = 18;

pa[24].ch = 'X',w[24] = 1;

pa[25].ch = 'Y',w[25] = 16;

pa[26].ch = 'Z',w[26] = 1;

输出

186 52 0 0

64 44 0 0

13 31 0 0

22 36 0 0

32 38 0 0

103 47 0 0

21 35 0 0

15 32 0 0

47 40 0 0

57 42 0 0

1 27 0 0

5 29 0 0

32 38 0 0

20 34 0 0

57 43 0 0

63 43 0 0

15 33 0 0

1 27 0 0

48 41 0 0

51 41 0 0

80 45 0 0

23 36 0 0

8 30 0 0

18 34 0 0

1 28 0 0

16 33 0 0

1 28 0 0

2 29 10 17

2 30 24 26

7 31 27 11

10 32 28 22

20 35 29 2

25 37 30 7

31 37 16 25

38 39 23 13

41 39 31 6

45 40 3 21

56 42 32 33

64 44 4 12

79 45 34 35

92 46 36 8

99 46 18 19

113 47 37 9

120 48 14 15

128 48 1 38

159 49 39 20

191 50 40 41

216 50 5 42

248 51 43 44

345 51 45 0

407 52 46 47

593 0 48 49

593 0 50 0

1

A010

B100101

C00000

D0110

E010

F10011

G011001

H0001

I0111

J10010000

K1001001

L0111

M10001

N000

O001

P011010

Q10010001

R0010

S0011

T101

U00001

V0110001

W10000

X01100000

Y011011

Z01100001

输入ASDFGHJ

输出：0100011011010011011001000110010000

1. 附录

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct

{

int weight;

int parent;

int Lchild;

int Rchild;

}HFtree , \*pHFtree;

struct word

{

char ch;

char code[20];

int codelength;

};

void select(int \*s1, int \*s2, pHFtree Hf, int n)

{

int \_min = 9999;

int i,L1=0,L2=0;

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

{

if(Hf[i].weight<\_min&&Hf[i].parent==0)

{

\_min = Hf[i].weight;

L1 = i;

}

}

\*s1 = L1;

\_min = 9999;

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

{

if(Hf[i].weight<\_min&&(i!=\*s1)&&Hf[i].parent==0)

{

\_min = Hf[i].weight;

L2 = i;

}

}

\*s2 = L2;

}

void createHftree(pHFtree Hf, int \*w, int n)

{

int i;

int s1,s2;

int m = 2\*n - 1;

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

{

Hf[i].weight = w[i];

Hf[i].parent = 0;

Hf[i].Lchild = 0;

Hf[i].Rchild = 0;

}

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

{

Hf[i].weight = 0;

Hf[i].parent = 0;

Hf[i].Lchild = 0;

Hf[i].Rchild = 0;

}

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

{

select(&s1,&s2,Hf,i-1);

Hf[i].weight = Hf[s1].weight+Hf[s2].weight;

Hf[s1].parent = i;

Hf[s2].parent = i;

Hf[i].Lchild = s1;

Hf[i].Rchild = s2;

}

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

{

printf("%d %d %d %d\n",Hf[i].weight,Hf[i].parent,Hf[i].Lchild,Hf[i].Rchild);

}

}

void showHfcode(pHFtree Hf,int n,struct word \* pa)

{

int start;

int i,j,k,t;

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

{

start = 19;

k = i;

t = Hf[k].parent;

while(t!=0)

{

if(Hf[t].Lchild == k)

pa[i].code[start--] = '0';

else

pa[i].code[start--] = '1';

k = t;

t = Hf[t].parent;

}

pa[i].codelength = 19-start;

}

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

{

printf("%c",pa[i].ch);

for(j=20-pa[i].codelength;j<20;j++)

printf("%c",pa[i].code[j]);

printf("\n");

}

}

void Decoding(char c,int n,struct word\*pa)

{

int j;

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

{

if(c ==pa[i].ch)

{

for(int j=20-pa[i].codelength;j<20;j++)

printf("%c",pa[i].code[j]);

}

}

}

int main(void)

{

int n=27;

char str[20];

struct word \* pa = (struct word \*)malloc(n\*sizeof(word));

int m = 2\*n-1;

pHFtree Hf = (pHFtree)malloc(m\*sizeof(HFtree));

int \*w = (int\*)malloc(n\*sizeof(int));

pa[0].ch = ' ',w[0] = 186;

pa[1].ch = 'A',w[1] = 64;

pa[2].ch = 'B',w[2] = 13;

pa[3].ch = 'C',w[3] = 22;

pa[4].ch = 'D',w[4] = 32;

pa[5].ch = 'E',w[5] = 103;

pa[6].ch = 'F',w[6] = 21;

pa[7].ch = 'G',w[7] = 15;

pa[8].ch = 'H',w[8] = 47;

pa[9].ch = 'I',w[9] = 57;

pa[10].ch = 'J',w[10] = 1;

pa[11].ch = 'K',w[11] = 5;

pa[12].ch = 'L',w[12] = 32;

pa[13].ch = 'M',w[13] = 20;

pa[14].ch = 'N',w[14] = 57;

pa[15].ch = 'O',w[15] = 63;

pa[16].ch = 'P',w[16] = 15;

pa[17].ch = 'Q',w[17] = 1;

pa[18].ch = 'R',w[18] = 48;

pa[19].ch = 'S',w[19] = 51;

pa[20].ch = 'T',w[20] = 80;

pa[21].ch = 'U',w[21] = 23;

pa[22].ch = 'V',w[22] = 8;

pa[23].ch = 'W',w[23] = 18;

pa[24].ch = 'X',w[24] = 1;

pa[25].ch = 'Y',w[25] = 16;

pa[26].ch = 'Z',w[26] = 1;

createHftree(Hf,w,n);

showHfcode(Hf,n,pa);

printf("Input the sentence:\n");

scanf("%s",str);

int N = strlen(str);

for(int i=0;i<N;i++)

{

Decoding(str[i],n,pa);

}

}