**实验七：哈夫曼树编码 (2学时)**

**[问题描述]**

利用哈夫曼编码进行通讯可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据进行预先编码。试用算法实现哈夫曼树。

**[实验目的]**

1. 通过哈夫曼树的定义，掌握构造哈夫曼树的意义。
2. 掌握构造哈夫曼树的算法思想。
3. 通过具体构造哈夫曼树，进一步理解构造哈夫曼树编码的意义。

**[实验内容及要求]**

输入相应的n个权值，建立哈夫曼树，并输出哈夫曼编码。

**#include** <limits.h>

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <string.h>

**typedef** ***char*** **\*\****HuffmanCode*;

**typedef** ***struct***

{

***int*** weight, parent,

      LChild, RChild;

} *HTNode*, \**HuffmanTree*;

// 函数声明

***void*** CreatHuffmanTree(*HuffmanTree* &*HT*, ***int*** *w*[], ***int*** *n*);

***void*** select(*HuffmanTree* &*HT*, ***int*** *n*, ***int*** &*s1*, ***int*** &*s2*);

***void*** GetHuffmanCode(*HuffmanTree* *HT*, *HuffmanCode* &*HC*, ***int*** *n*);

***void*** printCode(*HuffmanTree* *HT*, ***char*** \*\**a*, ***int*** *n*);

***void*** PreOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*);

***void*** InOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*);

***void*** PostOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*);

***void*** CreatHuffmanTree(*HuffmanTree* &*HT*, ***int*** *w*[], ***int*** *n*)

{

***int*** s1, s2, i;

*HT* **=** **new** *HTNode*[2 **\*** *n* **-** 1];

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

  {

*HT*[i].weight **=** *w*[i];

*HT*[i].LChild **=** *HT*[i].RChild **=** *HT*[i].parent **=** 0;

  }

**for** (i **=** *n*; i **<** 2 **\*** *n* **-** 1; i**++**)

  {

*HT*[i].LChild **=** *HT*[i].RChild **=** *HT*[i].parent **=** 0;

  }

***int*** m **=** 2 **\*** *n* **-** 1;

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

  {

    select(*HT*, i **-** 1, s1, s2);

*HT*[i].weight **=** *HT*[s1].weight **+** *HT*[s2].weight;

*HT*[i].LChild **=** s1;

*HT*[i].RChild **=** s2;

*HT*[s1].parent **=** i;

*HT*[s2].parent **=** i;

  }

}

***void*** select(*HuffmanTree* &*HT*, ***int*** *n*, ***int*** &*s1*, ***int*** &*s2*)

{

***int*** i;

*s1* **=** *s2* **=** 0;

***int*** min1 **=** INT\_MAX; //最小值

***int*** min2 **=** INT\_MAX; //次小值

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

  {

**if** (*HT*[i].parent **==** 0)

    {

**if** (*HT*[i].weight **<** min1) //如果比最小值小

      {

        min2 **=** min1;

*s2* **=** *s1*;

        min1 **=** *HT*[i].weight;

*s1* **=** i;

      }

**else** **if** ((*HT*[i].weight **>=** min1) **&&** (*HT*[i].weight **<** min2)) //如果大于等于最小值，且小于次小值

      {

        min2 **=** *HT*[i].weight;

*s2* **=** i;

      }

**else** //如果大于次小值，则什么都不做

      {

        ;

      }

    }

  }

}

***void*** GetHuffmanCode(*HuffmanTree* *HT*, *HuffmanCode* &*HC*, ***int*** *n*)

{

***int*** c, start, p;

*HC* **=** **new** ***char*** **\***[*n*];

***char*** **\***cd;

  cd **=** (***char*** **\***)malloc(*n* **\*** **sizeof**(***char***));

  cd[*n* **-** 1] **=** '\0';

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

  {

    start **=** *n* **-** 1;

    c **=** i;

    p **=** *HT*[c].parent;

**while** (p **!=** 0)

    {

**if** (*HT*[p].LChild **==** c)

        cd[**--**start] **=** '0';

**else**

        cd[**--**start] **=** '1';

      c **=** p, p **=** *HT*[c].parent;

    }

*HC*[i] **=** (***char*** **\***)malloc((*n* **-** start) **\*** **sizeof**(***char***));

    strcpy(*HC*[i], **&**cd[start]);

  }

  free(cd);

}

***void*** printCode(*HuffmanTree* *HT*, ***char*** \*\**a*, ***int*** *n*)

{

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

  {

    printf("%2d---", *HT*[i].weight);

    puts(*a*[i]);

  }

}

***void*** PreOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*)

{

  printf("%-3d ", *ht*.weight);

**if** (*ht*.LChild **!=** 0)

    PreOrder(*HT*, *HT*[*ht*.LChild]);

**if** (*ht*.RChild **!=** 0)

    PreOrder(*HT*, *HT*[*ht*.RChild]);

}

***void*** InOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*)

{

**if** (*ht*.LChild **!=** 0)

    InOrder(*HT*, *HT*[*ht*.LChild]);

  printf("%-3d ", *ht*.weight);

**if** (*ht*.RChild **!=** 0)

    InOrder(*HT*, *HT*[*ht*.RChild]);

}

***void*** PostOrder(*HuffmanTree* &*HT*, *HTNode* &*ht*)

{

**if** (*ht*.LChild **!=** 0)

    PostOrder(*HT*, *HT*[*ht*.LChild]);

**if** (*ht*.RChild **!=** 0)

    PostOrder(*HT*, *HT*[*ht*.RChild]);

  printf("%-3d ", *ht*.weight);

}

***int*** main()

{

*HuffmanTree* HT;

*HuffmanCode* HC;

***int*** n;

  printf("输入权值的个数：");

  scanf("%d", **&**n);

***int*** **\***w **=** **new** ***int***[2 **\*** n **-** 1];

  printf("输入相应的%d个权值，建立哈夫曼树：", n);

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

    scanf("%d", **&**w[i]);

  CreatHuffmanTree(HT, w, n);

  GetHuffmanCode(HT, HC, n);

  PostOrder(HT, HT[2 **\*** n **-** 2]);

  printf("\nHuffmanCode：\n");

  printCode(HT, HC, n);

**return** 0;

}

