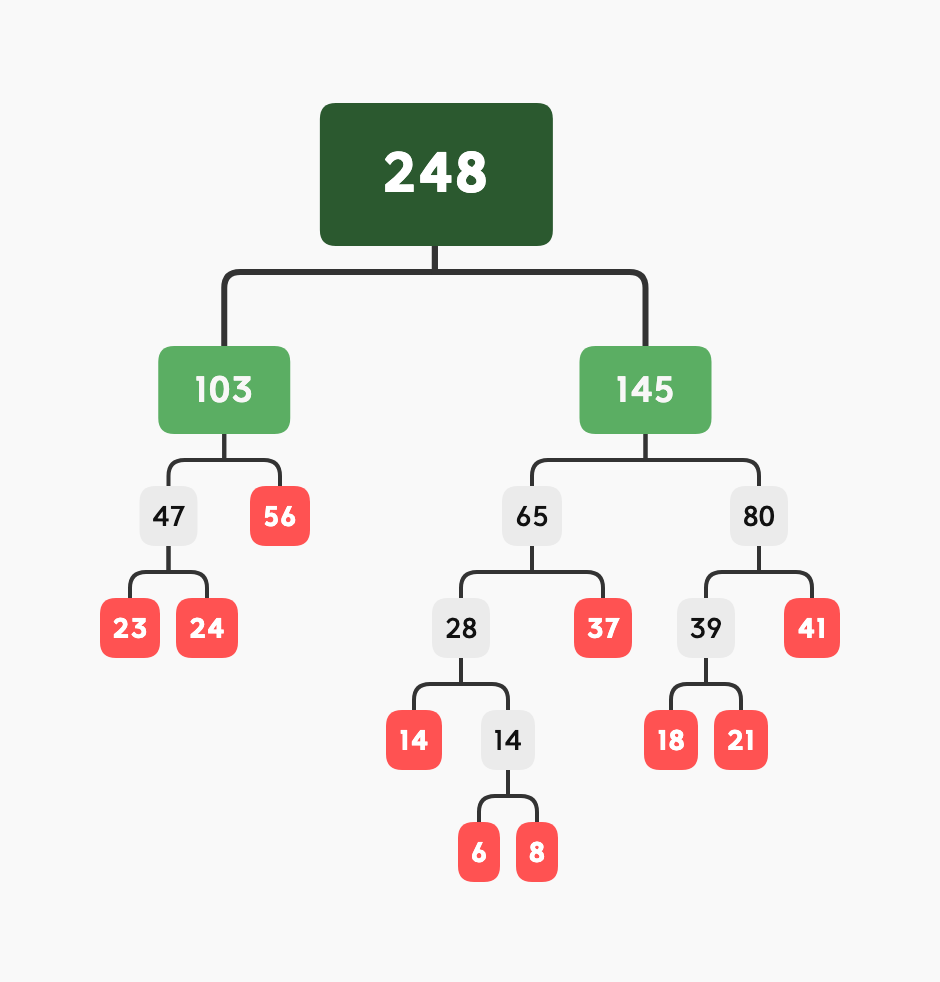
数据结构作业

姓名：叶先志 学号：18040400011 班级：1818039

第14题

哈夫曼树：

从哈夫曼树根结点开始，对左子树分配代码0，右子树分配代码1， 一直到达叶子结点为止，然后将从树根沿每条路径到达叶子结点的代码排列起来，便得到了哈夫曼编码：

|  |  |  |
| --- | --- | --- |
| 符号编号 | 符号出现频率 | 哈夫曼编码 |
| 0 | 23 | 000 |
| 1 | 24 | 001 |
| 2 | 14 | 1000 |
| 3 | 6 | 10010 |
| 4 | 8 | 10011 |
| 5 | 37 | 101 |
| 6 | 56 | 01 |
| 7 | 18 | 1100 |
| 8 | 21 | 1101 |
| 9 | 41 | 111 |

第15题

原二叉排序树：

图片包含 游戏机, 钟表

描述已自动生成

删除结点37后的二叉排序树：

图片包含 游戏机, 钟表

描述已自动生成

第17题

代码：

1. #include<stdio.h>
2. #include<stdlib.h>
3. #define MAX 100
4. **typedef** **char** elemtype;
6. **typedef** **struct** node         //定义二叉树
7. {
8. elemtype ch;
9. **struct** node \*lchild,\*rchild;
10. }Bitree;
12. Bitree \*CreatTree()     //二叉树的建立，广度优先输入
13. {
14. **char** m;
15. Bitree \*S,\*root;
16. root=NULL;
17. Bitree \*Q[MAX];         //指向每个元素的指针构成的队列
18. **int** front=1,rear=0;
19. **while**((m=getchar())!='#')
20. {
21. S=NULL;
22. **if**(m!='@')
23. {
24. S=(Bitree\*)malloc(**sizeof**(Bitree));
25. S->ch=m;
26. S->lchild=NULL;
27. S->rchild=NULL;
28. }
29. rear++;
30. Q[rear]=S;
31. **if**(rear==1)
32. {
33. root=S;
34. }
35. **else**
36. {
37. **if**(S&&Q[front])
38. {
39. **if**(rear%2==0)
40. {
41. Q[front]->lchild=S;
42. }
43. **else**
44. {
45. Q[front]->rchild=S;
46. }
47. }
48. **if**(rear%2==1)
49. {
50. front++;
51. }
52. }
53. }
54. **return** root;
55. }
57. **void** preOrder(Bitree \*root)//非递归的先序遍历算法
58. {
59. Bitree \*Stack[MAX];
60. Bitree \*s;
61. **int** top=-1;
62. **if**(root!=NULL)
63. {
64. top++;
65. Stack[top]=root;
66. **while**(top!=-1)
67. {
68. s=Stack[top];
69. top--;
70. **while**(s)
71. {
72. printf("%c ",s->ch);
73. **if**(s->rchild!=NULL)     //若有右孩子，入栈
74. {
75. top++;
76. Stack[top]=s->rchild;
77. }
78. s=s->lchild;
79. }
80. }
81. }
82. **else**
83. printf("二叉树为空！\n");
84. }
86. **int** main()
87. {
88. Bitree \*root;
89. printf("请输入按广度优先的二叉树，@表示虚结点，#表示结束：\n");
90. root=CreatTree();
91. printf("按非递归的先序遍历算法可以得到：\n");
92. preOrder(root);
93. printf("\n");
94. **return** 0;
95. }

程序效果：

手机屏幕截图

描述已自动生成

输入二叉树为：

图片包含 游戏机, 钟表

描述已自动生成

第21题

代码：

1. #include<stdio.h>
2. #include<stdlib.h>
3. #define MAX 100
4. **typedef** **struct** node         //定义二叉树
5. {
6. **char** ch;
7. **struct** node \*lchild,\*rchild;
8. } Bitree;
10. Bitree \*creatree()          //二叉树的建立，广度优先输入
11. {
12. **char** m;
13. Bitree \*Q[MAX];
14. Bitree \*s,\*root;
15. root=NULL;
16. **int** front=1,rear=0;
17. **while**((m=getchar())!='#')
18. {
19. s=NULL;
20. **if**(m!='@')
21. {
22. s=(Bitree\*)malloc(**sizeof**(Bitree));
23. s->ch=m;
24. s->lchild=NULL;
25. s->rchild=NULL;
26. }
27. rear++;
28. Q[rear]=s;
29. **if**(rear==1)
30. root=s;
31. **else**
32. {
33. **if**(Q[front]&&s)
34. {
35. **if**(rear%2==0)
36. Q[front]->lchild=s;
37. **else**
38. Q[front]->rchild=s;
39. }
40. **if**(rear%2==1)
41. front++;
42. }
43. }
44. **return** root;
45. }
47. **void** Layer(Bitree \*root)    //按广度优先遍历输出,利用队列的先进先出来实现
48. {
49. Bitree \*Q[MAX];
50. Bitree \*s;
51. **int** front=0,rear=1;
52. **if**(root)                    //二叉树不为空时
53. {
54. Q[rear]=root;
55. **while**(front<rear)       //队列从1开始，front指向队头的头元素
56. {
57. front++;
58. printf("%c ",Q[front]->ch);      //出队列
59. s=Q[front];
60. **if**(s->lchild!=NULL)             //入队列
61. {
62. rear++;
63. Q[rear]=s->lchild;
64. }
65. **if**(s->rchild!=NULL)
66. {
67. rear++;
68. Q[rear]=s->rchild;
69. }
70. }
71. printf("\n");
72. }
73. **else**
74. printf("二叉树为空！\n");
75. }
77. **void** exchange(Bitree \*root)      //交换左右子树
78. {
79. Bitree \*s;
80. **if**(root)
81. {
82. s=root->lchild;
83. root->lchild=root->rchild;
84. root->rchild=s;
85. exchange(root->lchild);  //根节点的左子树交换
86. exchange(root->rchild);  //根节点的右子树交换
87. }
88. }
90. **int** main()
91. {
92. Bitree \*root;
93. printf("请输入按广度优先的二叉树，@表示虚结点，#表示结束：\n");
94. root=creatree();
95. printf("交换之前按广度优先遍历可以得到：\n");
96. Layer(root);
97. printf("交换之后按广度优先遍历可以得到：\n");
98. exchange(root);
99. Layer(root);
100. **return** 0;
101. }

程序效果：

手机屏幕截图

描述已自动生成

输入二叉树为：

图片包含 游戏机, 钟表

描述已自动生成

交换后输出的二叉树为：

图片包含 游戏机, 画

描述已自动生成

哈夫曼编码译码程序：

代码：

1. #include <stdio.h>
2. #include <stdlib.h>
3. **int** num\_of\_leaves;      //符号个数
4. **int** num\_of\_nodes;       //节点个数
6. **typedef** **struct**
7. {
8. **char** bits[50];  //编译器不允许用变量来设置数组长度
9. **int** start;
10. **char** ch;
11. }codetype;
12. codetype code[50];
14. **typedef** **struct**
15. {
16. **int** lchild,rchild,parent;
17. **float** weight;
18. **char** ch;
19. }hufmtree;
20. hufmtree \*root;
22. hufmtree \* creathufmtree()
23. {
24. **int** i,j,p1,p2,s1,s2;
25. hufmtree \* root;
26. num\_of\_nodes=2\*num\_of\_leaves-1;
27. root=(hufmtree\*)malloc(num\_of\_nodes\***sizeof**(hufmtree));
28. **for**(i=0;i<num\_of\_nodes;i++)         //初始化
29. {
30. root[i].lchild=root[i].parent=root[i].rchild=root[i].weight=-1;
31. root[i].ch='@';
32. }
34. printf("请连续输入%d个符号（无空格）：\n",num\_of\_leaves);
35. **for**(i=0;i<num\_of\_leaves;i++)
36. scanf("%c",&root[i].ch);
37. getchar();
39. printf("请依次输入每个符号的权值：\n");
40. **for**(i=0;i<num\_of\_leaves;i++)    //输入前n个点的权值
41. scanf("%f",&root[i].weight);
42. getchar();
44. **for**(i=num\_of\_leaves;i<num\_of\_nodes;i++)    //合并
45. {
46. p1=p2=0;
47. s1=s2=1000;
48. **for**(j=0;j<i;j++)    //从第一个节点遍历，寻找权值最小的两个节点
49. {
50. **if**(root[j].parent==-1)
51. {
52. **if**(root[j].weight<s1)
53. {
54. s2=s1;
55. s1=root[j].weight;
56. p2=p1;
57. p1=j;
58. }
59. **else** **if**(root[j].weight<s2)
60. {
61. s2=root[j].weight;
62. p2=j;
63. }
64. }
65. }
66. root[p1].parent=i;
67. root[p2].parent=i;
68. root[i].lchild=p1;
69. root[i].rchild=p2;
70. root[i].weight=root[p1].weight+root[p2].weight;
71. }
72. **return** root;
73. }
75. **void** hufmcode(hufmtree \*root)
76. {
77. **int** i,son,p;
78. codetype temp;
79. **for**(i=0;i<num\_of\_leaves;i++)
80. {
81. temp.start=num\_of\_leaves;
82. temp.ch=root[i].ch;       //保留root[i]的符号
83. son=i;
84. p=root[son].parent;
85. **while**(p!=-1)
86. {
87. temp.start--;
88. **if**(root[p].lchild==son)
89. temp.bits[temp.start]='0';
90. **else**
91. temp.bits[temp.start]='1';
92. son=p;
93. p=root[son].parent;
94. }
95. code[i]=temp;
96. }
97. printf("生成的密码表为：\n");
98. **for**(i=0;i<num\_of\_leaves;i++)
99. {
100. printf("%c \t ",code[i].ch);
101. **for**(**int** j=code[i].start;j<num\_of\_leaves;j++)
102. {
103. printf("%c",code[i].bits[j]);
104. }
105. printf("\n");
106. }
107. }
109. **int** hufmdecode(hufmtree \*root)
110. {
111. **int** i,bit;
112. i=num\_of\_nodes-1;       //从根节点向下遍历
113. printf("请输入二进制代码，每位以空格间隔，最终输入2结束：\n");
114. scanf("%d",&bit);
115. **while**(bit!=2)
116. {
117. **if**(bit==0)
118. i=root[i].lchild;
119. **else** **if**(bit==1)
120. i=root[i].rchild;
121. **else**
122. {
123. printf("请输入0或1，或输入2结束！\n");
124. **return** 0;
125. }
126. **if**(root[i].lchild==-1)      //找到叶子,输出符号
127. {
128. printf("%c",root[i].ch);
129. printf("\n");
130. i=num\_of\_nodes-1;
131. }
132. scanf("%d",&bit);
133. }
134. **return** 0;
135. }
137. **int** main(**int** argc, **const** **char** \* argv[])
138. {
139. printf("请输入符号个数：");
140. scanf("%d",&num\_of\_leaves);
141. getchar();
142. root=creathufmtree();
143. /\*
144. for(int i=0;i<num\_of\_leaves;i++)
145. printf("%c",root[i].ch);
146. printf("\n");
147. \*/
148. hufmcode(root);
149. hufmdecode(root);
150. **return** 0;
151. }

程序效果：

手机屏幕的截图

描述已自动生成

检测第十五题：

手机屏幕的截图

描述已自动生成