

🕒 当前作业

» [22级第六次作业 \(查找与排序\)](#)

» [22级第五次作业 \(树\)](#)

» [2022级 \(信息大类\) 数据结构综合作业 \(正确性和性能\)](#)

» [2022级 \(信息大类\) 数据结构综合作业 \(可扩展性\)](#)

🕒 历史作业

» [22级第四次作业 \(栈和队\)](#)

» [22级第三次作业 \(线性表\)](#)

» [22级第二次作业](#)

» [22级第一次作业](#)

» [21级第七次作业 \(图\)](#)

» [21级第六次作业 \(查找与排序\)](#)

» [21级第五次作业 \(树\)](#)

» [2021级 \(信息大类\) 数据结构综合作业 \(正确性和性能\)](#)

» [2021级 \(信息大类\) 数据结构综合作业 \(可扩展性\)](#)

» [21级第四次作业 \(栈和队\)](#)

» [21级第三次作业](#)

21级第五次作业 (树)

作业时间： 2022-04-28 18:00:00 至 2022-07-01 02:00:00

第五次作业主要考查对树和二叉树知识的掌握情况，请用相关知识完成本次作业。

✎ 选择题

1.

首次提交时间:2022-05-05 08:57:20 最后一次提交时间:2022-05-05 08:57:20

在一棵度为4的树T中，若有20个度为4的结点，10个度为3的结点，1个度为2的结点，10个度为1的结点，则树T的叶节点个数是B\_\_\_\_\_。

A. 41 B. 82 C. 113 D. 122
2.

首次提交时间:2022-04-28 20:12:55 最后一次提交时间:2022-04-28 20:12:55

一个满二叉树有m个树枝，n个结点，其深度为h，则 D\_\_\_\_\_。

A.  $n = h + m$  B.  $h + m = 2n$  C.  $m = h - 1$  D.  $n = 2^h - 1$
3.

首次提交时间:2022-04-28 20:13:28 最后一次提交时间:2022-05-05 09:00:21

若二叉树的前序序列与后序序列的次序正好相反，则该二叉树一定是D\_\_\_\_\_的二叉树。

A. 空或仅有一个结点

B. 其分支结点无左子树

C. 其分支结点无右子树

D. 其分支结点的度都为1
4.

首次提交时间:2022-04-28 20:13:33 最后一次提交时间:2022-04-28 20:13:33

在二叉查找树中进行查找的效率与A\_\_\_\_\_有关。

A. 二叉查找树的深度

B. 二叉查找树的结点个数

C. 被查找结点的度

D. 二叉查找树的存储结构
5.

首次提交时间:2022-05-05 08:54:41 最后一次提交时间:2022-05-05 08:54:42

将森林F转换为对应的二叉树T，F中叶结点的个数等于C\_\_\_\_\_：

A. T中叶结点的个数

B. T中度为1的结点个数

C. T中左孩子指针为空的结点个数

D. T中右孩子指针为空的结点个数

6. 首次提交时间:2022-04-28 20:18:41 最后一次提交时间:2022-05-05 09:00:57

当一棵有n 个结点的二叉树按层次从上到下，同层次从左到右将数据存放在一维数组  $A[1..n]$  中时，数组中第i 个结点的左孩子为 D。

- A.  $A[2i]$  ( $2i \leq n$ )
- B.  $A[2i+1]$  ( $2i+1 \leq n$ )
- C.  $A[i/2]$
- D. 无法确定

7. 首次提交时间:2022-05-04 13:28:28 最后一次提交时间:2022-05-04 13:28:29

已知一算术表达式的中缀形式为  $A+B*C-D/E$ ，后缀形式为  $ABC*+DE/-$ ，其前缀形式为(D)。

- A.  $-A+B*C/DE$
- B.  $-A+B*CD/E$
- C.  $-+*ABC/DE$
- D.  $-+A*BC/DE$

8. 首次提交时间:2022-05-04 13:29:24 最后一次提交时间:2022-05-04 13:29:24

5个字符有如下4种编码方案。其中，不是前缀编码的是 B：

- A. 0, 10, 110, 1111
- B. 11, 10, 001, 101, 0001
- C. 00, 010, 0110, 1000
- D. b, c, aa, ac, aba, abb, abc

9. 首次提交时间:2022-05-05 08:34:03 最后一次提交时间:2022-05-05 08:34:04

由带权为3, 9, 6, 2, 5的五个叶子结点构成一颗哈夫曼树，则带权路径长度为 B

- A. 54
- B. 55
- C. 65
- D. 25

10. 首次提交时间:2022-05-05 08:17:33 最后一次提交时间:2022-05-05 08:17:34

有11个叶结点的哈夫曼树共有 B 个结点。

- A. 22
- B. 21
- C. 20
- D. 19

1.

首次提交时间:2022-05-05 08:20:07    最后一次提交时间:2022-05-05 08:20:45

已提交

对具有n个结点的完全二叉树按照层次从上到下，每一层从左到右的次序对所有结点进行编号，编号为i的结点的双亲结点的编号为    $\lfloor i/2 \rfloor$   ，左孩子的编号为    $2*i$    右孩子的编号为    $2*i+1$   。（从1开始编号，用[x]表示对x向下取整）
2.

首次提交时间:2022-05-05 08:21:25    最后一次提交时间:2022-05-05 08:21:33

已提交

度为k的树中，第i层最多有    $k^{i-1}$   个结点（ $i \geq 1$ ）（2的幂指数可表示为形如  $2^{(n+1)}$  形式）
3.

首次提交时间:2022-05-05 08:22:27    最后一次提交时间:2022-05-05 08:22:30

已提交

若一棵满二叉树有2047个结点，则该二叉树中叶结点的个数为   1024  。
4.

首次提交时间:2022-05-05 08:24:07    最后一次提交时间:2022-05-05 08:24:13

已提交

已知某完全二叉树采用顺序存储结构，结点的存放次序为A, B, C, D, E, F, G, H, I, J, 则该二叉树的后序序列为   HIDJEBFGCA  。（答案中不要加入空格及其他符号，格式如ABCDE）
5.

首次提交时间:2022-05-05 08:24:29    最后一次提交时间:2022-05-05 08:25:04

已提交

若具有n个结点的二叉树采用二叉链表存储结构，则该链表中有    $2*n$   个指针域，其中    $n-1$   个指针域用于链接孩子结点，    $n+1$   个指针域空闲存放着NULL。
6.

首次提交时间:2022-05-05 08:26:21    最后一次提交时间:2022-05-05 08:26:25

已提交

已知二叉树的前序遍历序列为ABDCEFG,中序遍历序列是DBCAFEG,则其后序遍历序列为   DCBFGEA  。（答案中不要加空格及其他符号）
7.

首次提交时间:2022-05-05 08:26:55    最后一次提交时间:2022-05-05 08:58:59

已提交

在顺序存储的二叉树中，编号为i 和j 的两个结点处在同一层的条件是    $\lfloor \log i \rfloor == \lfloor \log j \rfloor$   。（利用[]表示向下取整， $\log_2 X$ 可表示为： $\log x$ ）
8.

首次提交时间:2022-05-05 08:48:37    最后一次提交时间:2022-05-05 08:49:11

已提交

如果A,B,C,D的值分别为2, 3, 4, 5, 试计算下列前缀表达式的值。  
(1) + - × A B C D     答:   7    
(2) - × A + B C D     答:   9
9.

首次提交时间:2022-05-05 08:44:52    最后一次提交时间:2022-05-05 08:44:52

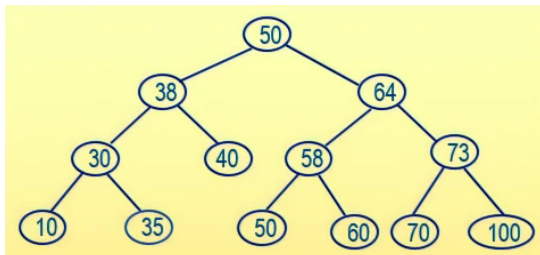
已提交


采用逐点插入法建立序列（54,28,16,34,73,62,95,60,26,43）的二叉查找树后，查找数据元素62共进行   3   次元素间的比较。
10.

首次提交时间:2022-05-05 08:31:22    最后一次提交时间:2022-05-05 08:36:48

已提交

若以{4, 5, 6, 7, 8}作为叶子结点的权值构造哈夫曼树，则其带权路径长度是   69  。

#	题目	分值	批阅信息												
1.	<a href="#">树叶节点遍历 (树-基础题)</a>	20.00	<a href="#">下载源文件</a>												
<div>【问题描述】</div> <p>从标准输入中输入一组整数，在输入过程中按照左子结点值小于根结点值、右子结点值大于等于根结点值的方式构造一棵二叉查找树，然后从左至右输出所有树中叶结点的值及高度（根结点的高度为1）。例如，若按照以下顺序输入一组整数：50、38、30、64、58、40、10、73、70、50、60、100、35，则生成下面的二叉查找树：</p> <div></div> <p>从左到右的叶子结点包括：10、35、40、50、60、70、100，叶结点40的高度为3，其它叶结点的高度都为4。</p> <div>【输入形式】</div> <p>先从标准输入读取整数的个数，然后从下一行开始输入各个整数，整数之间以一个空格分隔。</p> <div>【输出形式】</div> <p>按照从左到右的顺序分行输出叶结点的值及高度，值和高度之间以一个空格分隔。</p> <div>【样例输入】</div> <pre>13 50 38 30 64 58 40 10 73 70 50 60 100 35</pre> <div>【样例输出】</div> <pre>10 4 35 4 40 3 50 4 60 4 70 4 100 4</pre> <div>【样例说明】</div> <p>按照从左到右的顺序输出叶结点（即没有子树的结点）的值和高度，每行输出一个。</p> <div>【评分标准】</div> <p>该题要求输出所有叶结点的值和高度，提交程序名为：bst.c</p>															
<div>得分20.00 最后一次提交时间:2022-04-28 20:05:18</div> <div>共有测试数据:5 平均占用内存:1.400K 平均CPU时间:0.00649S 平均墙钟时间:0.00650S</div> <table><tr><th>测试数据</th><th>评判结果</th></tr><tr><td>测试数据1</td><td>完全正确</td></tr><tr><td>测试数据2</td><td>完全正确</td></tr><tr><td>测试数据3</td><td>完全正确</td></tr><tr><td>测试数据4</td><td>完全正确</td></tr><tr><td>测试数据5</td><td>完全正确</td></tr></table> <div>详细</div>				测试数据	评判结果	测试数据1	完全正确	测试数据2	完全正确	测试数据3	完全正确	测试数据4	完全正确	测试数据5	完全正确
测试数据	评判结果														
测试数据1	完全正确														
测试数据2	完全正确														
测试数据3	完全正确														
测试数据4	完全正确														
测试数据5	完全正确														

#	题目	分值	批阅信息												
2.	<a href="#">词频统计 (树实现)</a>	20.00	<a href="#">下载源文件</a>												
<div><div><div>【问题描述】</div><div>编写程序统计一个英文文本文件中每个单词的出现次数（词频统计），并将统计结果按单词字典序输出到屏幕上。</div><div>要求：程序应用二叉排序树（BST）来存储和统计读入的单词。</div><div>注：在此单词为仅由字母组成的字符序列。包含大写字母的单词应将大写字母转换为小写字母后统计。在生成二叉排序树不做平衡处理。</div><div>【输入形式】</div><div>打开当前目录下文件<b>article.txt</b>，从中读取英文单词进行词频统计。</div><div>【输出形式】</div><div>程序应首先输出二叉排序树中根节点、根节点的右节点及根节点的右节点的右节点上的单词（即<b>root</b>、<b>root-&gt;right</b>、<b>root-&gt;right-&gt;right</b>节点上的单词），单词中间有一个空格分隔，最后一个单词后没有空格，直接为回车（若单词个数不足三个，则按实际数目输出）。</div><div>程序将单词统计结果按单词字典序输出到屏幕上，每行输出一个单词及其出现次数，单词和其出现次数间由一个空格分隔，出现次数后无空格，直接为回车。</div><div>【样例输入】</div><div>当前目录下文件<b>article.txt</b>内容如下：  "Do not take to heart every thing you hear."  "Do not spend all that you have."  "Do not sleep as long as you want;"</div><div>【样例输出】</div><div>do not take  all 1  as 2  do 3  every 1  have 1  hear 1  heart 1  long 1</div></div><div><div>得分20.00  最后一次提交时间:2022-05-04 14:01:31</div><div>成功编译,但有警告信息.</div><div>word.c: In function 'main':</div><div>word.c:42:17: warning: array subscript has type 'char' [-Wchar-subscripts]</div><div>temp[q++] = s[i];</div><div>^</div><div>word.c:44:17: warning: array subscript has type 'char' [-Wchar-subscripts]</div><div>temp[q] = '\0';</div><div>^</div><div>共有测试数据:5</div><div>平均占用内存:1.399K  平均CPU时间:0.00527S  平均墙钟时间:0.00523S</div></div><div><table><tr><th>测试数据</th><th>评判结果</th></tr><tr><td>测试数据1</td><td>完全正确</td></tr><tr><td>测试数据2</td><td>完全正确</td></tr><tr><td>测试数据3</td><td>完全正确</td></tr><tr><td>测试数据4</td><td>完全正确</td></tr><tr><td>测试数据5</td><td>完全正确</td></tr></table></div><div>详细 </div></div>				测试数据	评判结果	测试数据1	完全正确	测试数据2	完全正确	测试数据3	完全正确	测试数据4	完全正确	测试数据5	完全正确
测试数据	评判结果														
测试数据1	完全正确														
测试数据2	完全正确														
测试数据3	完全正确														
测试数据4	完全正确														
测试数据5	完全正确														

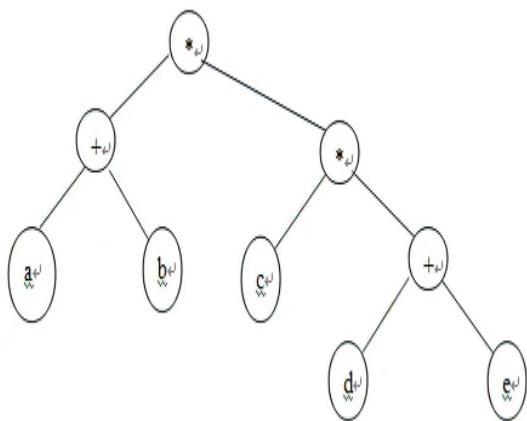
#	题目	分值	批阅信息
	not 3		
	sleep 1		
	spend 1		
	take 1		
	that 1		
	thing 1		
	to 1		
	want 1		
	you 3		
	<p>【样例说明】</p> <p>程序首先在屏幕上输出程序中二叉排序树上根节点、根节点的右子节点及根节点的右子节点的右子节点上的单词，分别为<b>do not take</b>，然后按单词字典序依次输出单词及其出现次数。</p> <p>【评分标准】</p> <p>通过全部测试点得满分</p>		

#	题目	分值	批阅信息
3.	<a href="#">计算器（表达式计算-表达式树实现）</a>	20.00	<a href="#">下载源文件</a>
<div>【问题描述】</div> <p>从标准输入中读入一个整数算术运算表达式，如<math>24 / ( 1 + 2 + 36 / 6 / 2 - 2 ) * ( 12 / 2 / 2 ) =</math>，计算表达式结果，并输出。</p> <p>要求：</p> <ol style="list-style-type: none"><li>表达式运算符只有+、-、*、/，表达式末尾的=字符表示表达式输入结束，表达式中可能会出现空格；</li><li>表达式中会出现圆括号，括号可能嵌套，不会出现错误的表达式；</li><li>出现除号/时，以整数相除进行运算，结果仍为整数，例如：<math>5/3</math>结果应为1。</li></ol> <p>4、要求采用表达式树来实现表达式计算。</p> <p>表达式树（<b>expression tree</b>）：</p> <p>我们已经知道了在计算机中用后缀表达式和栈来计算中缀表达式的值。在计算机中还有一种方式是利用表达式树来计算表达式的值。表达式树是这样一种树，其根节点为操作符，非根节点为操作数，对其进行后序遍历将计算表达式的值。由后缀表达式生成表达式树的方法如下：</p> <ul style="list-style-type: none"><li>● 读入一个符号；</li><li>● 如果是操作数，则建立一个单节点树并将指向他的指针推入栈中；</li><li>● 如果是运算符，就从栈中弹出指向两棵树T1和T2的指针（T1先弹出）并形成一棵新树，树根为该运算符，它的左、右子树分别指向T2和T1，然后将新树的指针压入栈中。</li></ul> <p>例如输入的后缀表达为：</p> <p><b>ab+cde+**</b></p> <p>则生成的表达式树为：</p>			
<div>得分20.00 最后一次提交时间:2022-04-28 20:08:08</div> <div>成功编译,但有警告信息.</div> <div>expressiontree.c: In function 'opt_cmp':</div> <div>expressiontree.c:37:5: warning: array subscript has type 'char' [-Wchar-subscripts]</div> <div>else return h[a]-h[b];</div> <div>^</div> <div>expressiontree.c:37:5: warning: array subscript has type 'char' [-Wchar-subscripts]</div> <div>expressiontree.c: In function 'main':</div> <div>expressiontree.c:121:5: warning: implicit declaration of function 'gets' [-Wimplicit-function-declaration]</div> <div>gets(epr);</div> <div>^</div> <div>expressiontree.c: In function 'Calc':</div> <div>expressiontree.c:119:1: warning: control reaches end of non-void function [-Wreturn-type]</div> <div>}</div> <div>^</div> <div>/tmp/ccEMX0bG.o: In function `main':</div> <div>expressiontree.c:(.text.startup+0xc): warning: the `gets' function is dangerous and should not be used.</div> <div>共有测试数据:5</div> <div>平均占用内存:1.401K 平均CPU时间:0.00635S 平均墙钟时间:0.00633S</div>			

测试数据	评判结果
------	------

测试数据1	完全正确
测试数据2	完全正确
测试数据3	完全正确
测试数据4	完全正确
测试数据5	完全正确

详细





【输入形式】

从键盘输入一个以=结尾的整数算术运算表达式。操作符和操作数之间可以有空格分隔。

【输出形式】

首先在屏幕上输出表达式树根、左子节点及右子节点上的运算符或操作数，中间由一个空格分隔，最后有一个回车（如果无某节点，则该项不输出）。然后输出表达式计算结果。

【样例输入】

24 / ( 1 + 2 + 36 / 6 / 2 - 2 ) \* ( 12 / 2 / 2 ) =

【样例输出】

\* / /

18

【样例说明】

按照运算符及括号优先级依次计算表达式的值。在生成的表达树中，\*是根节点的运算符，/ 是根节点的左子节点上运算符，/是根节点的右子节点上运算符，按题目要求要输出。

【评分标准】

通过所有测试点得满分。



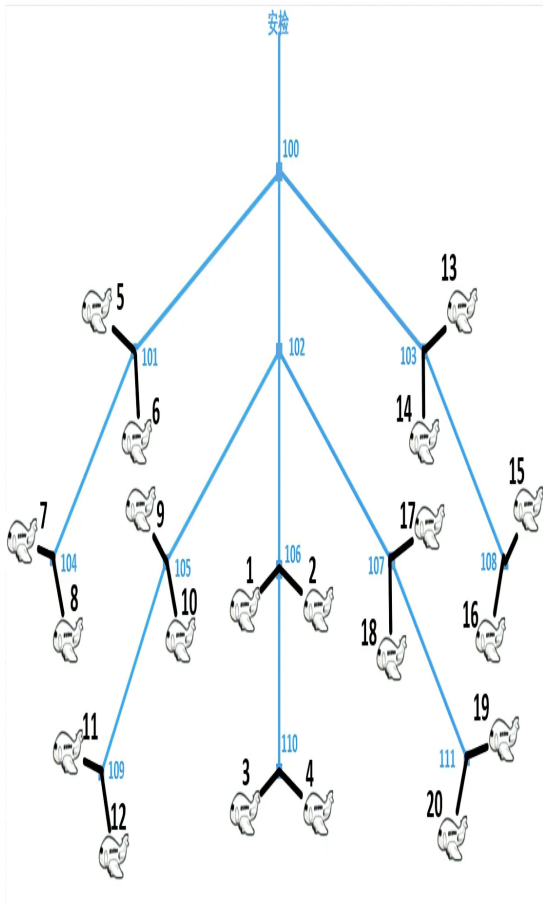
4. 服务优化

20.00

下载源文件

【问题描述】

假设某机场所有登机口（Gate）呈树形排列（树的度为3），安检处为树的根，如下图所示。图中的分叉结点（编号大于等于100）表示分叉路口，登机口用小于100的编号表示（其一定是一个叶结点）。通过对机场所有出发航班的日志分析，得知每个登机口每天的平均发送旅客流量。作为提升机场服务水平的一个措施，在不改变所有航班相对关系的情况下（即：出发时间不变，原在同一登机口的航班不变），仅改变登机口（例如：将3号登机口改到5号登机口的位置），使得整体旅客到登机口的时间有所减少（即：从安检口到登机口所经过的分叉路口最少）。



编写程序模拟上述登机口的调整，登机口调整规则如下：

- 1）首先按照由大到小的顺序对输入的登机口流量进行排序，流量相同的按照登机口编号由小到大排序；
- 2）从上述登机口树的树根开始，将登机口按照从上到下（安检口在最上方）、从左到右的顺序，依次对应上面排序后将要调整的登机口。

例如上图的树中，若只考虑登机口，则从上到下有三层，第一层从左到右的顺序为：5、6、14、13，第二层从左到右的顺序为：7、8、9、10、1、2、18、17、16、15，第三层从左到右的顺序为：11、12、3、4、20、19。若按规则1排序后流量由大至小的前五个登机口为3、12、16、20、15，则将

得分20.00 最后一次提交时间:2022-05-01 15:20:30

共有测试数据:5  
平均占用内存:1.399K 平均CPU时间:0.00547S 平均墙钟时间:0.00544S

测试数据	评判结果
测试数据1	完全正确
测试数据2	完全正确
测试数据3	完全正确
测试数据4	完全正确
测试数据5	完全正确

详细

#	题目	分值	批阅信息
	<p>流量最大的3号登机口调整到最上层且最左边的位置（即：5号登机口的位置），12号调整到6号，16号调整到14号，20号调整到13号，15号调整到第二层最左边的位置（即7号登机口的位置）。</p> <p>【输入形式】</p> <p>1) 首先按层次从根开始依次输入树结点之间的关系。其中分叉结点编号从数字100开始（树根结点编号为100，其它分叉结点编号没有规律但不会重复），登机口为编号小于100的数字（编号没有规律但不会重复，其一定是一个叶结点）。树中结点间关系用下面方式描述：</p> <p>R S1 S2 S3 -1</p> <p>其中R为分叉结点，从左至右S1，S2，S3分别为树叉R的子结点，其可为树叉或登机口，由于树的度为3，S1，S2，S3中至多可以2个为空，最后该行以-1和换行符结束。各项间以一个空格分隔。如：</p> <p>100 101 102 103 -1</p> <p>表明编号100的树根有三个子叉，编号分别为101、102和103，又如：</p> <p>104 7 8 -1</p> <p>表明树叉104上有2个编号分别为7和8的登机口。</p> <p>假设分叉结点数不超过100个。分叉结点输入的顺序不确定，但可以确定：输入某个分叉结点信息时，其父结点的信息已经输入。</p> <p>输入完所有树结点关系后，在新的一行上输入-1表示树结点关系输入完毕。</p> <p>2) 接下来输入登机口的流量信息，每个登机口流量信息分占一行，分别包括登机口编号（1~99之间的整数）和流量（大于0的整数），两整数间以一个空格分隔。登机口数目与前面构造树时的登机机口数目一致。</p> <p>【输出形式】</p> <p>按照上述调整规则中排序后的顺序（即按旅客流量由大到小，流量相同的按照登机口编号由小到大）依次分行输出每个登机口的调整结果：先输出调整前的登机口编号，然后输出字符串“-&gt;”（由英文减号字符与英文大于字符组成），再输出要调整到的登机口编号。</p> <p>【样例输入】</p> <p>100 101 102 103 -1</p> <p>103 14 108 13 -1</p>		

#	题目	分值	批阅信息
	101 5 104 6 -1		
	104 7 8 -1		
	102 105 106 107 -1		
	106 1 110 2 -1		
	108 16 15 -1		
	107 18 111 17 -1		
	110 3 4 -1		
	105 9 109 10 -1		
	111 20 19 -1		
	109 11 12 -1		
	-1		
	17 865		
	5 668		
	20 3000		
	13 1020		
	11 980		
	8 2202		
	15 1897		
	6 1001		
	14 922		
	7 2178		
	19 2189		
	1 1267		
	12 3281		
	2 980		
	18 1020		
	10 980		
	3 1876		
	9 1197		
	16 980		
	4 576		
	【样例输出】		
	12->5		
	20->6		
	8->14		
	19->13		
	7->7		

#	题目	分值	批阅信息
	15->8		
	3->9		
	1->10		
	9->1		
	13->2		
	18->18		
	6->17		
	2->16		
	10->15		
	11->11		
	16->12		
	14->3		
	17->4		
	5->20		
	4->19		
	<b>【样例说明】</b>		
	样例输入了12条树结点关系，形成了如上图的树。然后输入了20个登机口的流量，将这20个登机口按照上述调整规则1排序后形成的顺序为：12、20、8、19、7、15、3、1、9、13、18、6、2、10、11、16、14、17、5、4。最后按该顺序将所有登机口按照上述调整规则2进行调整，输出调整结果。		
	<b>【评分标准】</b>		
	该题要求计算并输出登机口的调整方法，提交程序名为adjust.c。		
5.	<a href="#">基于Huffman码的文件压缩和解压工具 (选做, 本题只测试压缩功能)</a>	0.00	还未提交代码
			详细
6.	<a href="#">基于Huffman码的文件压缩和解压工具 (选做, 本题只测试解压功能)</a>	0.00	还未提交代码
			详细
	程序片段编程题		

#	题目	分值	批阅信息
---	----	----	------

1. [实验：树的构造与遍历](#)

20.00 [下载源文件](#)

## 1. 实验目的与要求

在学习和理解二叉树的原理、构造及遍历方法的基础上，应用所学知识来解决实际问题。

本实验将通过一个实际应用问题的解决过程掌握Huffman树的构造、Huffman编码的生成及基于所获得的Huffman编码压缩文本文件。

涉及的知识点包括树的构造、遍历及C语言位运算和二进制文件。

## 2. 实验内容

### Huffman编码文件压缩

#### 【问题描述】

编写一程序采用Huffman编码对一个正文文件进行压缩。具体压缩方法如下：

- 对正文文件中字符(换行字符'\n' 除外，不统计)按出现次数（即频率）进行统计。
- 依据字符频率生成相应的Huffman树（未出现的字符不生成）。
- 依据Huffman树生成相应字符的Huffman编码。
- 依据字符Huffman编码压缩文件（即将源文件字符按照其Huffman编码输出）。

说明：

- 只对文件中出现的字符生成Huffman树，注意：一定不要处理\n，即不要为其生成Huffman编码。
- 采用ASCII码值为0的字符作为压缩文件的结束符（即可将其出现次数设为1来参与编码）。
- 在生成Huffman树前，初始在对字符频率权重进行（由小至大）排序时，频率相同的字符ASCII编码值小的在前；新生成的权重节点插入到有序权重序列中时，若出现相同权重，则将新生成的权重节点插入到原有相同权重节点之后（采用稳定排序）。
- 在生成Huffman树时，权重节点在前的作为左孩子节点，权重节点在后的作为右孩子节点。
- 遍历Huffman树生成字符Huffman码时，左边为0右边为1。
- 源文件是文本文件，字符采用ASCII编码，每个字符占8个二进制位；而采用Huffman编码后，高频字符编码长度较短（小于8位），

得分20.00 最后一次提交时间:2022-05-06 19:34:08

成功编译,但有警告信息.

```
main.c: In function 'statCount':
main.c:62:3: warning: array subscript has type 'char' [-Wchar-subscripts]
++Ccount[text[i]];
^
main.c: In function 'dfs':
main.c:132:3: warning: array subscript has type 'char' [-Wchar-subscripts]
if(p->left!=NULL){
^
main.c: In function 'atoHZIP':
main.c:160:3: warning: array subscript has type 'char' [-Wchar-subscripts]
if(u%8!=0){
^
```

共有测试数据:4

平均占用内存:1.402K 平均运行时间:0.00534S

#### 测试数据

#### 评判结果

测试数据1

完全正确

测试数据2

完全正确

测试数据3

完全正确

测试数据4

完全正确

详细 

因此最后输出时需要使用C语言中的位运算将字符的Huffman码依次输出到每个字节中。

【输入形式】

对当前目录下文件input.txt进行压缩。

【输出形式】

将压缩后结果输出到文件output.txt中，同时将压缩结果用十六进制形式（printf("%x",...)）输出到屏幕上，以便检查和查看结果。

3. 实验准备

1. 文件下载

从教学平台（judge.buaa.edu.cn）课程下载区下载文件lab\_tree2.rar，该文件中包括了本实验中用到的文件huffman2student.c和input.txt：

- huffman2student.c：该文件给出本实验程序的框架，框架中部分内容未完成（见下面相关实验步骤），通过本实验补充完成缺失的代码，使得程序运行后得到相应要求的运行结果；
- input.txt：为本实验的测试数据。

2. huffman2student.c文件中相关数据结构说明

结构类型说明：

```
struct tnode { //Huffman树
    结构节点类型

    char c;

    int weight;

    struct tnode *left;

    struct tnode *right;

};
```

结构类型struct tnode用来定义Huffman树的节点，其中；

- 1) 对于树的叶节点，成员c和weight用来存放字符及其出现次数；对于非叶节点来说，c值可不用考虑，weight的值满足Huffman树非叶节点生成条件，若p为当前Huffman树节点指针，则有：

```
p->weight = p->left->weight
+ p->right->weight;
```

2) 成员left和right分别为Huffman树节点左右子树节点指针。

全局变量说明：

```
int Ccount[128]={0};

struct tnode *Root=NULL;

char HCode[128][MAXSIZE]={0};

int Step=0;

FILE *Src, *Obj;
```

整型数组Ccount存放每个字符的出现次数，如Ccount['a']表示字符a的出现次数。

变量Root为所生成的Huffman树的根节点指针。

数组HCode用于存储字符的Huffman编码，如HCode['a']为字符a的Huffman编码，本实验中为字符串“1000”。

变量Step为实验步骤状态变量，其取值为1、2、3、4，分别对应实验步骤1、2、3、4。

变量Src、Obj为输入输出的文件指针，分别用于打开输入文件“input.txt”和输出文件“output.txt”。

## 4. 实验步骤

### 【步骤1】

1) 实验要求

在程序文件huffman2student.c中“//【实验步骤1】开始”和“//【实验步骤1】结束”间编写相应代码，以实现函数statCount，统计文本文件input.txt中字符出现频率。

```
//【实验步骤1】开始

void statCount()

{

}

//【实验步骤1】结束
```

2) 实验说明



函数statCount用来统计输入文件（文件指针为全局变量Src）中字符的出现次数（频率），并将字符出现次数存入全局变量数组Ccount中，如Ccount['a']存放字符a的出现次数。

注意：在该函数中Ccount[0]一定要置为1，即Ccount[0]=1。编码值为0（'\0'）的字符用来作为压缩文件的结束符。

3) 实验结果

函数print1()用来打印输出步骤1的结果，即输出数组Ccount中字符出现次数多于0的字符及次数，编码值为0的字符用NUL表示。完成【步骤1】编码后，本地编译并运行该程序，并在标准输入中输入1，程序运行正确时在屏幕上将输出如下结果：

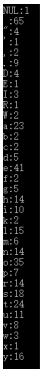


图1步骤1运行结果

在本地运行正确的情况下，将你所编写的程序文件中//【实验步骤1】开始”和“//【实验步骤1】结束”间的代码拷贝粘贴到实验报告后所附代码【实验步骤1】下的框中，然后点击提交按钮，若得到如下运行结果（测试数据1评判结果为完全正确）：

测试数据	评判结果
测试数据1	完全正确
测试数据2	输出错误
测试数据3	输出错误
测试数据4	输出错误

表明实验步骤1：通过，否则：不通过。

【步骤2】

1) 实验要求

在程序文件huffman2student.c中的“//【实验步骤2】开始”和“//【实验步骤2】结束”间编写相应代码，实现函数createHTree，该函数生成一个根结点指针为Root的Huffman树。

```
//【实验步骤2】开始
```

```
void createHTree()
```

```
{
```

```
}
```

```
//【实验步骤2】结束
```

## 2) 实验说明

在程序文件huffman2student.c中函数createHTree将根据每个字符的出现次数（字符出现次数存放在全局数组Ccount中，Ccount[i]表示ASCII码值为i的字符出现次数），按照Huffman树生成规则，生成一棵Huffman树。

算法提示：

1. 依据数组Ccount中出现次数不为0的（即Ccount[i]>0）项，构造出树林F={T0, T1, ..., Tm}，初始时Ti(0≤i≤m)为只有一个根结构的树，且根结点(叶结点)的权值为相应字符的出现次数的二叉树（每棵树结点的类型为struct tnode，其成员c为字符，weight为树节点权值）：

```
for (i=0; i<128; i++)
```

```
if (Ccount[i]>0) {
```

```
    p = (struct tnode *)malloc(sizeof(struct tnode));
```

```
    p->c = i; p->weight = Ccount[i];
```

```
    p->left = p->right = NULL;
```

```
    add p into F;
```

```
}
```

2. 对树林F中每棵树按其根结点的权值由小至大进行排序（排序时，当权值weight

相同时，字符c小的排在前面），得到一个有序树林F

3. while 树个数>1  
n F

a) 将F中T0和T1  
作为左、右子树  
合并成为一棵新的  
二叉树T’，并  
令T’->weight=  
T0->weight+ T1-  
>wei

b) 删除T0和T1 f  
rom F，同时将  
T’ 加入F。要求  
加入T’ 后F仍然  
有序。若F中有树  
根结点权值与  
T’ 相同，则  
T’ 应加入到其后

4. Root = T0 （Root  
为Huffman树的根结点  
指针。循环结束时，F  
中只有一个T0）

注：在实现函数createHTree时，在框中  
还可根据需要定义其它函数，例如：

```
void myfun()

{

...

}

void createHTree()

{

...

myfun();

...

}
```

3) 实验结果

函数print2()用来打印输出步骤2的结果，即按前序遍历方式遍历步骤2所生成（由全局变量Root所指向的）Huffman树结点字符信息。输出时编码值为0的字符用NUL表示、空格符用SP表示、制表符用TAB表示、回车符用CR表示。完成【步骤2】编码后，本地编译并运

行该程序，并在标准输入中输入2，程序运行正确时在屏幕上将输出如下结果：



图2 步骤2运行结果

在本地运行正确的情况下，将你在本地所编写的程序文件中//【实验步骤2】开始”和“//【实验步骤2】结束”间的代码拷贝粘贴到实验报告后所附代码【实验步骤2】下的框中，然后点击提交按钮，若得到如下运行结果（测试数据2评判结果为完全正确）：

测试数据	评判结果
测试数据1	完全正确
测试数据2	完全正确
测试数据3	输出错误
测试数据4	输出错误

表明实验步骤2：通过，否则：不通过。

【步骤3】

1) 实验要求

在程序文件huffman2student.c中的“//【实验步骤3】开始”和“//【实验步骤3】结束”间编写相应代码，实现函数makeHCode，该函数依据【实验步骤3】中所产生的Huffman树为文本中出现的每个字符生成对应的Huffman编码。遍历Huffman树生成字符Huffman码时，左边为0右边为1。

```
//【实验步骤3】开始

void makeHCode()

{

}

//【实验步骤3】结束
```

2) 实验说明

【步骤3】依据【步骤2】所生成的根结点为Root的Huffman树生为文本中出现的每个字符生成相应的Huffman编码。全局变量HCode定义如下：

```
char HCode[128][MAXSIZE];

HCode变量用来存放每个字符的Huffman编码串，如HCode['e']存放的是字母e的Huffman编码串，在
```

本实验中实际值将为字符串“011”。

算法提示：

可编写一个按前序遍历方法对根节点为Root的树进行遍历的递归函数，并在遍历过程中用一个字符串来记录遍历节点时从根节点到当前节点的路径（经过的边），经过左边时记录为’ 0’，经过右边时记录为’ 1’；当遍历节点为叶节点时，将对应路径串存放 to 相应的HCode数组中，即执行strcpy (HCode[p->c], 路径串)。

注：在实现函数makeHCode时，在框中还可根据需要定义其它函数，如调用一个有类于前序遍历的递归函数来遍历Huffman树生成字符的Huffman编码：

```
void visitHTree()

{
    ...
}

void makeHCode()

{
    ...

    visitHTree();

    ...
}
```

3) 实验结果

函数print3()用来打印输出步骤3的结果，即输出步骤3所生成的存储在全局变量HCode中非空字符的Huffman编码串。完成【步骤3】编码后，本地编译并运行该程序，并在标准输入中输入3，在屏幕上将输出ASCII字符与其Huffman编码对应表，冒号左边为字符，右边为其对应的Huffman编码，其中NUL表示ASCII编码为0的字符，SP表示空格字符编码值为0的字符用，程序运行正确时在屏幕上将输出如下结果：

```
NUL:01001010
SP:111
":000000
,:01001011
,:10011111
.:01000
D:000001
E:01011100
I:0101111
R:01011101
W:0000100
a:1000
b:0000101
c:0000110
d:010011
e:011
f:0000111
g:010110
h:10110
i:01010
k:0100100
l:11001
m:100110
n:10111
o:001
p:110100
r:11000
s:0001
t:1010
u:10010
v:110101
w:1001110
x:10011110
y:11011
```

图3 步骤3运行结果

在本地运行正确的情况下，将你在本地所编写的程序文件中//【实验步骤3】开始”和“ //【实验步骤3】结束”间的代码拷贝粘贴到实验报告后所附代码【实验步骤3】下的框中，然后点击提交按钮，若得到如下运行结果（测试数据3评判结果为完全正确）：

测试数据	评判结果
测试数据1	完全正确
测试数据2	完全正确
测试数据3	完全正确
测试数据4	输出错误

表明实验步骤3：通过，否则：不通过。

【步骤4】

1) 实验要求

在程序文件huffman2student.c函数中的“//【实验步骤4】开始”和“ //【实验步骤4】结束”间编写相应代码，实现函数atoHZIP，该函数依据【实验步骤3】中所生成的字符ASCII码与Huffman编码对应表（存储在全局变量HCode中，如HCode['e']存放的是字符e对应的Huffman编码串，在本实验中值为字符串“011”），将原文本文件（文件指针为Src）内容（ASCII字符）转换为Huffman编码文件输出到文件output.txt（文件指针为Obj）中，以实现文件压缩。同

时将输出结果用十六进制形式（printf("%x",...)）输出到屏幕上，以便检查和查看结果。

```
//【实验步骤4】开始

void atoHZIP()

{

}

//【实验步骤4】结束
```

2) 实验说明

Huffman压缩原理：在当前Windows、Linux操作系统下，文本文件通常以ASCII编码方式存储和显示。ASCII编码是定长编码，每个字符固定占一个字节（即8位），如字符'e'的ASCII编码为十进制101（十六进制65，二进制为01100101）。而Huffman编码属于可变长编码，本实验中其依据文本中字符的频率进行编码，频率高的字符的编码长度短（小于8位），而频率低的字符的编码长度长（可能多于8位），如在本实验中，字符' '（空格）的出现频率最高（出现65次），其Huffman编码为111（占3位，远小于一个字节的8位），其它出现频率较高的字符，如字符'e'的Huffman编码为011、字符'o'的Huffman编码为111；字符'x'出现频率低（出现1次），其Huffman编码为10011110（占8位，刚好一个字节）（注意，在其它问题中，字符最长Huffman编码可能会超过8位）。正是由于高频字符编码短，将使得Huffman编码文件（按位）总长度要小于ASCII文本文件，以实现压缩文件的目的。

然而，将普通ASCII文本文件转换为变长编码的文件不便之处在于C语言中输入/输出函数数据处理的最小单位是一个字节（如putchar()），无法直接将Huffman（不定长）编码字符输出，在输出时需要将不定长编码序列转换为定长序列，按字节输出。而对于不定长编码，频率高的字符其编码要比一个字节短（如本实验中字符'e'的Huffman编码为011，不够一个字节，还需要和其它字符一起组成一个字节输出），频率低的编码可能超过

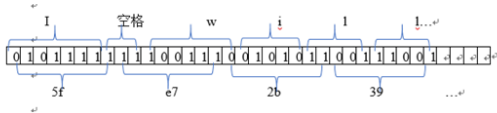


一个字节。如何将不定长编码字符序列转换成定长字符序列输出，一个简单方法是：

- 1) 根据输入字符序列将其 Huffman 编码串连接成一个（由0、1字符组成的）串；
- 2) 然后依次读取该串中字符，依次放入到一个字节的相应位上；
- 3) 若放满一个字节（即8位），可输出该字节；剩余的字符开始放入到下一个字节中；
- 4) 重复步骤2和3，直到串中所有字符处理完。

下面通过实例来说明：

原始文件input.txt中内容以“l wil l...”开始，依据所生成的Huffman码表，字母l对应的Huffman编码串为“0101111”，空格对应“111”，w对应“1001110”，i对应“01010”，l对应“11001”。因此，将其转换后得到一个Huffman编码串“0101111111001110010101100111001...”，由于在C中，最小输出单位是字节（共8位），因此，要通过C语言的位操作符将每8个01字符串放进一个字节中，如第一个8字符串“01011111”中的每个0和1放入到一个字符中十六进制（即printf（”%x”，c）输出时，屏幕上将显示5f）（如下图所示）。下面程序段将Huffman编码串每8个字符串放入一个字节（字符变量hc）中：



```
char hc;

...

for(i=0; s[i] != '\0'; i++) {

    hc = (hc << 1) | (s[i]-'0');

    if((i+1)%8 == 0) {

        fputc(hc,obj); //输出到
        目标（压缩）文件中

        printf("%x",hc); //按十六
        进制输出到屏幕上

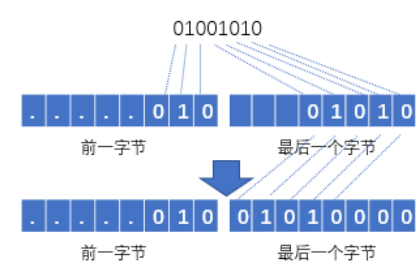
    }

}
```

...

说明：

1. 当遇到源文本文件输入结束时，应将输入结束符的Huffman码放到Huffman编码串最后，即将编码串HCode[0]放到Huffman编码串最后。
2. 在处理完成所有Huffman编码串时（如上述算法结束时），处理源文本最后一个字符（文件结束符）Huffman编码串（其编码串为“01001010”）时，可能出现如下情况：其子串”010”位于前一个字节中输出，而子串“01010”位于另（最后）一个字节的右5位中，需要将这5位左移至左端的头，最后3位补0，然后再输出最后一个字节。



注：在实现函数atoHZIP时，在框中还可根据需要定义其它函数或全局变量，如：

```
void myfun()

{

...

}

void atoHZIP()

{

...

myfun();

...

}
```

1) 实验结果

函数print4()用来打印输出步骤4的结果，即根据输出步骤3所生成的存储在全局变量HCode中Huffman编码串，依次对源文本文件（input.txt）的ASCII字符转换为Huffman编码字符输出到文件output.txt中，同时按十六进行输出到屏幕上。完成【步骤4】编码后，本地编译并运行该程序，并在标准输入中输入4，在屏幕上将输出：

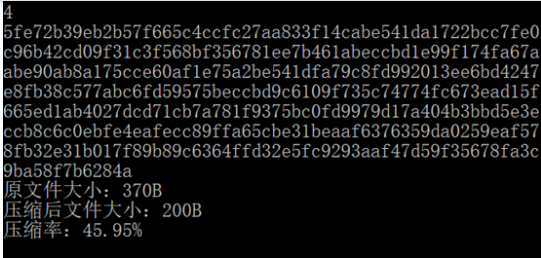


图4 步骤4运行结

说明：

从屏幕输出结果可以看出，由于采用了不定长的Huffman编码，且出现频率高的字符的编码长度短，压缩后，文件大小由原来的370字节变为200字节，文件压缩了45.95%。

在本地运行正确的情况下，将你在本地所编写的程序文件中//【实验步骤4】开始”和“//【实验步骤4】结束”间的代码拷贝粘贴到教学平台实验报告后所附代码【实验步骤4】下的框中，然后点击提交按钮，若得到如下运行结果（测试数据4评判结果为完全正确）：

测试数据	评判结果
测试数据1	完全正确
测试数据2	完全正确
测试数据3	完全正确
测试数据4	完全正确

表明实验步骤4：通过，否则：不通过。

北京航空航天大学

若重置密码，请与当前的任课教师联系