|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ch5 作业** | | | | | | | |
| 班级 | 01 | 学号 | 2022040906023 | 姓名 | 梁书恺 | 成绩 |  |
| 一、选择题（每个小题5分，共40分）  1. 用顺序存储的方法将完全二叉树中的所有结点逐层存放在数组A[1] ~ A[n]中，结点A[i]若有左子树，则左子树的根结点是（ D ）。  A[2i-1] B A[2i+1] C A[i/2] D A[2i]  2. 如果某二叉树的前序序列、中序序列和后序序列，结点a都在结点b的前面，则（ A ）。  A. a和b是兄弟 B. a是b的双亲 C. a是b的左孩子 D. a是b的右孩子  3. 已知某完全二叉树采用顺序存储，结点数据信息的存放顺序是ABCDEFGH，则该完全二叉树的后序遍历序列为（ A ）。  A. HDEBFGCA B. HEDBFGCA C. HDEBAFGC D. HDEFGBCA  4. 设X是树T中的一个非根结点，B是T所对应的二叉树。在B中，X是其双亲的右孩子，则下列结论正确的是（ D ）。  A 在树T中，X是其双亲的第一个孩子 B 在树T中，X一定无右兄弟  C 在树T中，X一定是叶子结点 D 在树T中，X一定有左兄弟  5. 一棵完全二叉树上有1001个结点，其中叶子结点的个数是（ D ）。  A．250 B． 500 C．254 D．501  6．把一棵树转换为二叉树后，这棵二叉树的形态是（ A ）。  A．唯一的 Ｂ．有多种  C．有多种，但根结点都没有左孩子 Ｄ．有多种，但根结点都没有右孩子  7. 由3个结点可以构造出多少种不同的二叉树？（ D ）  A．2 B．3 C．4 D．5  8. 设哈夫曼树中有199个结点，则该哈夫曼树中有（ B ）个叶子结点。  A．99 B．100 C．101 D．102  二、简答题（共40分）  图5-1 一棵二叉树  *GD*  *A*  *C*  *B*  *E*  *F*  *H*  1. 对于图5-1所示二叉树，解答下列问题：  （1）画出顺序存储示意图；  （2）画出二叉链表存储示意图；  （3）转换为森林。  **顺序存储：**    **二叉链表存储：**  **森林：**    2. 请将图5-2所示树画出孩子兄弟表示法存储示意图，并转换为二叉树。  孩子兄弟表示法存储示意图如下：  图5-2 一棵树  *I*  *A*  *B*  *C*  *D*  *E*  *F*  *G*  *H*  *J*    二叉树如下：    3. 已知一棵二叉树的前序遍历序列和中序遍历序列分别为ABCDEFGH 和CDBAFEHG，请构造该二叉树，写出后序遍历结果。  二叉树如下：    后序遍历结果为：DCBFHGEA  4. 假设用于通信的电文由字符集{a, b, c, d, e, f, g}中的字符构成，它们在电文中出现的次数分别为{31, 16, 10, 8, 11, 20, 4}。请为这7个字符设计哈夫曼编码，使用哈夫曼编码比使用等长编码使电文总长压缩多少？  哈夫曼树如下：    a:00  b:010  c:110  d:0111  e:111  f:10  g:0110  长度压缩：31 + 20 – 4 – 8 = 39 bit  三、算法设计（供20分）  1. 设计算法，统计二叉树的叶结点个数、分支结点个数。  typedef struct TreeNode  {  int data;  struct TreeNode \*left;  struct TreeNode \*right;  } TreeNode;  void count(TreeNode \*root, int \*leafCount, int \*branchCount)  {  if (root == NULL)  return;  if (root->left == NULL && root->right == NULL)  (\*leafCount)++;  else  (\*branchCount)++;  count(root->left, leafCount, branchCount);  count(root->right, leafCount, branchCount);  }  2. 交换二叉树每个结点的左孩子和右孩子。  typedef struct TreeNode  {  int data;  struct TreeNode \*left;  struct TreeNode \*right;  } TreeNode;  void swap(TreeNode \*root)  {  if (root == NULL)  return;  TreeNode \*temp = root->left;  root->left = root->right;  root->right = temp;  swap(root->left);  swap(root->right);  } | | | | | | | |
|  | | | | | | | |