▼ 第六章作业
- 选择题
- 判断题
- 简答题
- 算法题

第六章作业

2022211363 谢牧航

选择题

1. C
2. C
3. B
4. B
5. C
6. B
7. A
8. D
9. A
10. D
11. C
12. C

判断题

1. 对
2. 对
3. 错
4. 错
5. 错

简答题

1. 二叉树可以是度为0，1，2的树。
2. (1) 第 h 层为 k^{h-1}。

   (2) $\left\lfloor \dfrac{n-2-k}{k} \right\rfloor$

   (3) $(n-1)k+1+i$

   (4) 条件：$n-1 \not\equiv 0 \pmod{k}$，右兄弟编号：$n+1$
3. (1) [A, B, C, D, E, F, null, null, G, null, null, null, null, null, H]

   (2)

   Node A: [Value:A, *Lchild:pointer to B, *Rchild:pointer to C]

   Node B: [Value:B, *Lchild:pointer to D, *Rchild:pointer to E]

   Node C: [Value:C, *Lchild:pointer to F, *Rchild:null]

   Node D: [Value:D, *Lchild:null, *Rchild:null]

   Node E: [Value:E, *Lchild:pointer to G, *Rchild:null]

   Node F: [Value:F, *Lchild:null, *Rchild:pointer to H]

   Node G: [Value:G, *Lchild:null, *Rchild:null]

   Node H: [Value:H, *Lchild:null, *Rchild:null]

   (3) 中序遍历：Node D, Node B, Node G, Node E, Node A, Node F, Node H, Node C

   Node A: [Value:A, Left Thread:Node E, Right Thread:Node F]

   Node B: [Value:B, Left Thread:Node D, Right Thread:Node G]

   Node C: [Value:B, Left Thread:Node H, Right Thread:null]

   Node D: [Value:D, Left Thread:null, Right Thread:Node B]

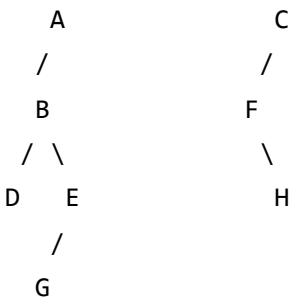
   Node E: [Value:E, Left Thread:G, Right Thread:Node A]

   Node F: [Value:F, Left Thread:A, Right Thread:Node H]

   Node G: [Value:G, Left Thread:B, Right Thread:Node E]

   Node H: [Value:H, Left Thread:F, Right Thread:Node C]

   (4)



4. a.画出哈夫曼树，编码为：a:01 b:001 c:110 d:0001 e:111 f:10 g:0000

   b. WPL:2.61

c. 0.39 / 3 = 0.13

算法题

1.

```
function countLeaves(node):
    if node is null:
        return 0
    # 计算当前节点是否为叶子节点
    isLeaf = (node.firstChild is null) ? 1 : 0
    # 递归计算在孩子链表中的叶子节点数
    childLeaves = countLeaves(node.firstChild)
    # 递归计算在兄弟链表中的叶子节点数
    siblingLeaves = countLeaves(node.nextSibling)
    # 返回该节点是否为叶子节点加上子节点和兄弟节点中的叶子节点数
    return isLeaf + childLeaves + siblingLeaves
```

2.

```
function treeDegree(node):
    if node is null:
        return 0
    # 计算当前节点的度
    currentDegree = 0
    child = node.firstChild
    while child is not null:
        currentDegree = currentDegree + 1
        child = child.nextSibling
    # 递归计算子树中的最大度
    maxDegreeInChildren = treeDegree(node.firstChild)
    # 递归计算兄弟中的最大度
    maxDegreeInSiblings = treeDegree(node.nextSibling)
    # 返回当前节点的度，子树中的最大度和兄弟中的最大度中的最大值
    return max(currentDegree, maxDegreeInChildren, maxDegreeInSiblings)
```