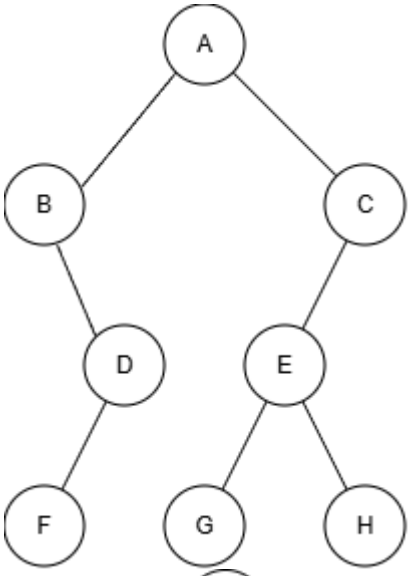
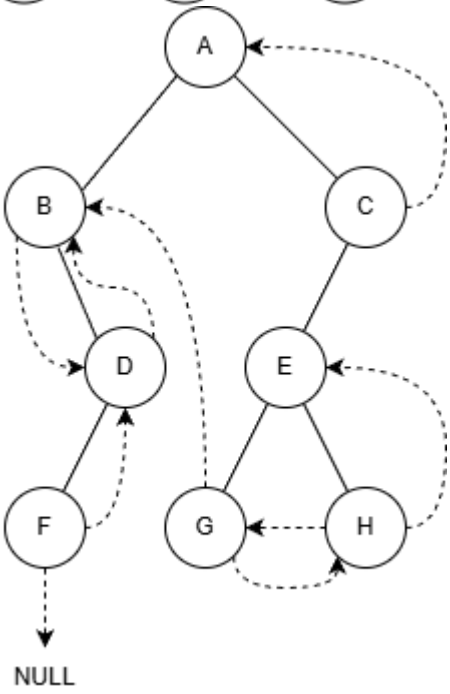
HW5

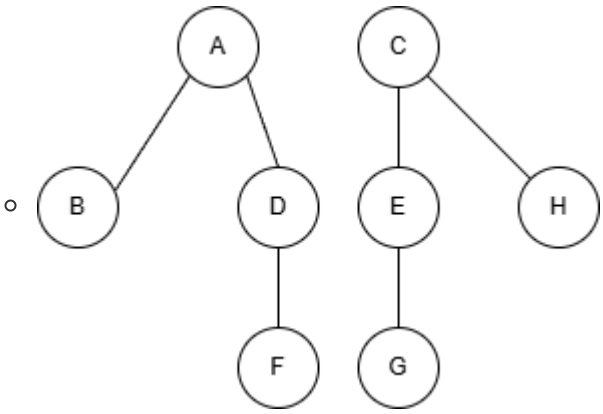
一. 选择题

1. D;
2. C;
3. B;
4. C;

二. 应用题

1.
   - 

   - 



2. 不妨认为 8 个字母依次为 A(0.07), B(0.19), C(0.02), D(0.06), E(0.32), F(0.03), G(0.21), H(0.10)。
    - A: 0101;
        - B: 11;
        - C: 01111;
        - D: 0110;
        - E: 00;
        - F: 01110;
        - G: 10;
        - H: 0100.
    - A: 000;
        - B: 001;
        - C: 010;
        - D: 011;
        - E: 100;
        - F: 101;
        - G: 110;
        - H: 111.
    - 方案一：

```
a. 优点：

    i. 平均码长2.61，压缩效率高，方便大量数据的传输与存储；

b. 缺点：

    i. 引入了哈夫曼树结构，需要额外存储解码工具，增加了数据量；

    ii. 解码较为复杂；

    iii. 对错误较为敏感，一位错误会导致后续的一系列错误。
```

        - 方案二：

          a. 优点：

            i. 模式简单，容易实现；

            ii. 定长解码，允许多线程进行；

            iii. 一位错误仅影响一个字符，抗干扰能力强；

          b. 缺点：

            i. 压缩效率低，浪费了存储空间和带宽。

三. 算法设计题

1.

```
//
// 头文件内容
//

#ifndef BINARY_TREE_H
#define BINARY_TREE_H

#include <stdbool.h>

typedef int ElemType;

typedef struct TreeNode
{
    ElemType data;
    struct TreeNode *lchild, *rchild;
} TreeNode;

typedef struct QueueNode
{
    TreeNode* treenode;
    struct QueueNode* next;
} QueueNode;

typedef struct
{
    QueueNode* front;
    QueueNode* rear;
} Queue;

TreeNode* createNode(ElemType data); // 创造新节点
bool isEqual(const TreeNode* t1, const TreeNode* t2); // 是否相等
void clear(TreeNode* root); // 销毁树

Queue* createQueue();
bool emptyQueue(const Queue* q);
void push(Queue* q, TreeNode* treenode);
TreeNode* pop(Queue* q);
void clearQueue(Queue* q);
void destoyQueue(Queue* q);

int getMaxWidth(TreeNode* root);

#endif //BINARY_TREE_H

```

```cpp
// 判别两棵树是否相等。
bool isEqual(const TreeNode* t1, const TreeNode* t2)
{
    if (t1 == NULL && t2 == NULL) return true;
    if (t1 == NULL || t2 == NULL || t1->data != t2->data) return false;

    return isEqual(t1->lchild, t2->lchild) && isEqual(t1->rchild, t2->rchild);
}
```

```cpp
// 计算二叉树最大的宽度
int getMaxWidth(TreeNode *root)
{
    if (root == NULL) return 0;

    Queue* q = createQueue();
    push(q, root);
    int maxWidth = 0;
    while(!emptyQueue(q))
    {
        QueueNode* temp = q->front->next; // q->front 为哑元
        int currentWidth = 0;
        while(temp != NULL)
        {
            currentWidth++;
            temp = temp->next;
        }

        if (currentWidth > maxWidth) maxWidth = currentWidth;

        for (int i = 0; i < currentWidth; i++)
        {
            TreeNode* node = pop(q);
            if (node->lchild != NULL) push(q, node->lchild);
            if (node->rchild != NULL) push(q, node->rchild);
        }
    }

    destoyQueue(q);
    return maxWidth;
}
```