树

1.二叉树

二叉树是一种非线性数据结构，代表

```
typedef struct TreeNode{
    int val;
    int height;
    struct TreeNode *left;
    struct TreeNode *right;
}TreeNode;



//构造函数
TreeNode* newTreeNode(int val){
    TreeNode* node =(TreeNode *)malloc(sizeof(TreeNode));
    node->val = val;
    node->left = NULL;
    node->right = NULL;
    return node;
}


//插入与删除节点
TreeNode* p = newTreeNode(0);
    n1->left = p;
    p->left = n2;
    p->right = NULL;


```

```
//二叉树的层序遍历 也即广度优先搜索 需要用到队列先进先出的特点
int *levelOrder(TreeNode* root, int* size){
    int front = 0,rear = 0;
    int index = 0;
    int* arr = (int *)malloc(sizeof(int)*MAX_SIZE);
    TreeNode* node;
    TreeNode** queue = (TreeNode **)malloc(sizeof(TreeNode*) * MAX_SIZE);

    //根节点入队
    queue[rear++] = root;




    //循环条件 当队列不为空
    while(rear > front){
```

```
        //节点出队
        node = queue[front++];

        //保存至数组
        arr[index++] = node->val;

        if(node->left != NULL){
            queue[rear++] = node->left;
        }if(node->right != NULL){
            queue[rear++] = node->right;
        }
    }
    *size = index;
    arr = realloc(arr,sizeof(int) * (*size));
    free(queue);
    return arr;
}
```

```

//二叉树前序遍历

void preorder(struct TreeNode* root,int* res, int* resSize){
    if(root == NULL){
        return;
    }
    res[(*resSize)++] = root->val;  // 1. 先访问根节点
    preorder(root->left, res, resSize); // 2. 再遍历左子树
    preorder(root->right, res, resSize);// 3. 再遍历右子树

 }

int* preorderTraversal(struct TreeNode* root, int* returnSize) {
    int* res = malloc(sizeof(int)*101);
    *returnSize = 0;
    preorder(root, res, returnSize);
    return res;

}


//二叉树的中序遍历
void inorder(struct TreeNode* root, int* res, int* resSize) {
    //终止条件
    if(root == NULL) {
        return;
    }

    inorder(root->left, res, resSize);       // 1. 先遍历左子树
    res[(*resSize)++] = root->val;           // 2. 再访问根节点
    inorder(root->right, res, resSize);      // 3. 最后遍历右子树
}
int* inorderTraversal(struct TreeNode* root, int* returnSize) {
    int* res = malloc(sizeof(int) * 101);
```

```
    *returnSize = 0;
    inorder(root, res, returnSize);
    return res;
}

//二叉树的后序遍历
 void postorder(struct TreeNode* root, int* res, int* resSize){
    if(root == NULL){
        return;
    }
    postorder(root->left, res, resSize);
    postorder(root->right, res, resSize);
    res[(*resSize)++] = root->val;

 }
int* postorderTraversal(struct TreeNode* root, int* returnSize) {
    int* res = malloc(sizeof(int)*2000);
    *returnSize = 0;
    postorder(root, res, returnSize);
    return res;
}
```