

树

1.二叉树

二叉树是一种非线性数据结构，代表

```
1  typedef struct TreeNode{
2      int val;
3      int height;
4      struct TreeNode *left;
5      struct TreeNode *right;
6  }TreeNode;
7
8
9
10 //构造函数
11 TreeNode* newTreeNode(int val){
12     TreeNode* node =(TreeNode *)malloc(sizeof(TreeNode));
13     node->val = val;
14     node->left = NULL;
15     node->right = NULL;
16     return node;
17 }
18
19
20 //插入与删除节点
21 TreeNode* p = newTreeNode(0);
22     n1->left = p;
23     p->left = n2;
24     p->right = NULL;
25
26
```

```
1  //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
2  int *levelOrder(TreeNode* root, int* size){
3      int front = 0, rear = 0;
4      int index = 0;
5      int* arr = (int *)malloc(sizeof(int)*MAX_SIZE);
6      TreeNode* node;
7      TreeNode** queue = (TreeNode **)malloc(sizeof(TreeNode*) * MAX_SIZE);
8
9      //根节点入队
10     queue[rear++] = root;
11
12
13
14
15     //循环条件 当队列不为空
16     while(rear > front){
```

```

17     //节点出队
18     node = queue[front++];
19
20     //保存至数组
21     arr[index++] = node->val;
22
23     if(node->left != NULL){
24         queue[rear++] = node->left;
25     }if(node->right != NULL){
26         queue[rear++] = node->right;
27     }
28 }
29 *size = index;
30 arr = realloc(arr, sizeof(int) * (*size));
31 free(queue);
32 return arr;
33 }

```

```

1
2 //二叉树前序遍历
3
4 void preorder(struct TreeNode* root, int* res, int* resSize){
5     if(root == NULL){
6         return;
7     }
8     res[(*resSize)++] = root->val; // 1. 先访问根节点
9     preorder(root->left, res, resSize); // 2. 再遍历左子树
10    preorder(root->right, res, resSize); // 3. 再遍历右子树
11
12 }
13
14 int* preorderTraversal(struct TreeNode* root, int* returnSize) {
15     int* res = malloc(sizeof(int)*101);
16     *returnSize = 0;
17     preorder(root, res, returnSize);
18     return res;
19 }
20
21
22
23 //二叉树的中序遍历
24 void inorder(struct TreeNode* root, int* res, int* resSize) {
25     //终止条件
26     if(root == NULL) {
27         return;
28     }
29
30     inorder(root->left, res, resSize); // 1. 先遍历左子树
31     res[(*resSize)++] = root->val; // 2. 再访问根节点
32     inorder(root->right, res, resSize); // 3. 最后遍历右子树
33 }
34 int* inorderTraversal(struct TreeNode* root, int* returnSize) {
35     int* res = malloc(sizeof(int) * 101);

```

```
36     *returnSize = 0;
37     inorder(root, res, returnSize);
38     return res;
39 }
40
41 //二叉树的后序遍历
42 void postorder(struct TreeNode* root, int* res, int* resSize){
43     if(root == NULL){
44         return;
45     }
46     postorder(root->left, res, resSize);
47     postorder(root->right, res, resSize);
48     res[(*resSize)++] = root->val;
49
50 }
51 int* postorderTraversal(struct TreeNode* root, int* returnSize) {
52     int* res = malloc(sizeof(int)*2000);
53     *returnSize = 0;
54     postorder(root, res, returnSize);
55     return res;
56 }
```