2021-10-29

> Today Learned:
>
> [Introduction of Algorithms lecture 2](#)
> (Corresponding to the 3rd chapters of the book)
>
> Today Exercise:
>
> [Inorder-Traversal of Binary Tree (From LeetCode)](#)

Source Code:

```c
void traversal(struct TreeNode* cur, int* rslt, int* resSize)
{
    if (!cur) return;
    traversal(cur->left, rslt, resSize);
    rslt[(*resSize)++] = cur->val;
    traversal(cur->right, rslt, resSize);
}
int* inorderTraversal(struct TreeNode* root, int* returnSize){
    int* rslt = malloc(sizeof(int)*501);
    *returnSize = 0;
    traversal(root, rslt, returnSize);
    return rslt;
}
```

Coding Notes:

Binary tree inorder traversal (recursive)

```
void traversal (struct Treenode* root, int* res, int*resSize){
```

We use pointer instead of global variables, which has the advantage of high security and can be used in all levels of recursion.

```
    if (!root) return; // if the leaf node is null, return the upper level)
    traversal (root, res, resSize);
    res[(*resSize)++] = root->val; // return value of this node
```

Here is a trick involved: the pointer can point to the counter and make it self-increase. // How about we limit the resSize is smaller than the value we set?

```
    traversal( root, res, resSize);
}
```

In the main function, we should do two important things:

1. initialize the "returnSize" and use it as "resSize".
2. initialize a result array whose size is "enough" with malloc and sizeof.

```
int* inordertraversal(strut Treenode* root, int* returnSize){
    *returnSize=0 // 1.
    int* res = malloc( sizeof(int)*501); //2.
    traversal(root, res, returnSize);
    return res;
}
```