Week4

DFS 代码模板

1.递归写法

python

```
visited = set()
def dfs(node, visited):
    if node in visited: # terminator
        # already visited
        return
    visited.add(node)
    # process current node here.
    ...
    for next_node in node.children():
        if next_node not in visited:
            dfs(next_node, visited)
```

Java

```
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> allResults = new ArrayList<>();
        if(root==null){
            return allResults;
        }
        travel(root,0,allResults);
        return allResults;
    }
    private void travel(TreeNode root,int level,List<List<Integer>> result
s){
        if(results.size()==level){
            results.add(new ArrayList<>());
        }
        results.get(level).add(root.val);
        if(root.left!=null){
            travel(root.left,level+1,results);
        }
        if(root.right!=null){
            travel(root.right,level+1,results);
        }
    }
```

2.非递归写法

python

```
def DFS(self, tree):
    if tree.root is None:
        return []
    visited, stack = [], [tree.root]
    while stack:
        node = stack.pop()
        visited.add(node)
        process (node)
        nodes = generate_related_nodes(node)
        stack.push(nodes)
    # other processing work
    ...
```

BFS代码模板

python

```
def BFS(graph, start, end):
    visited = set()
    queue = []
    queue.append([start])
    while queue:
        node = queue.pop()
        visited.add(node)
        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)
    # other processing work
    ...
```

Java

```
public class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode(int x) {
        val = x;
    }
}
public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> allResults = new ArrayList<>();
    if (root == null) {
        return allResults;
    }
    Queue<TreeNode> nodes = new LinkedList<>();
    nodes.add(root);
```

```
    while (!nodes.isEmpty()) {
        int size = nodes.size();
        List<Integer> results = new ArrayList<>();
        for (int i = 0; i < size; i++) {
            TreeNode node = nodes.poll();
            results.add(node.val);
            if (node.left != null) {
                nodes.add(node.left);
            }
            if (node.right != null) {
                nodes.add(node.right);
            }
        }
        allResults.add(results);
    }
    return allResults;
}
```

二分查找模板

python

```
left, right = 0, len(array) - 1
while left <= right:
      mid = (left + right) / 2
      if array[mid] == target:
            # find the target!!
            break or return result
      elif array[mid] < target:
            left = mid + 1
      else:
            right = mid - 1
```

Java

```
public int binarySearch(int[] array, int target) {
    int left = 0, right = array.length - 1, mid;
    while (left <= right) {
        mid = (right - left) / 2 + left;
        if (array[mid] == target) {
            return mid;
        } else if (array[mid] > target) {
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}
```