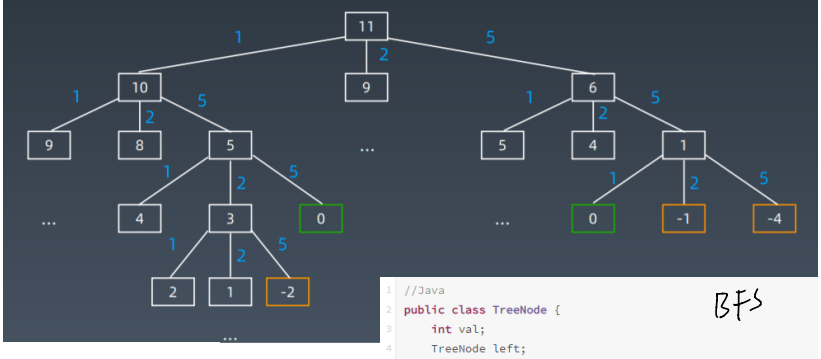
- 剪枝
- 双向 BFS
- 启发式搜索 (A*)

初级搜索 (暴力搜索)

1. 朴素搜索
2. 优化方式：不重复（fibonacci）、剪枝（生成括号问题）
3. 搜索方向：
   DFS: depth first search 深度优先搜索
   BFS: breadth first search 广度优先搜索

双向搜索、启发式搜索 高级搜索

Coin change（零钱置换）的状态树



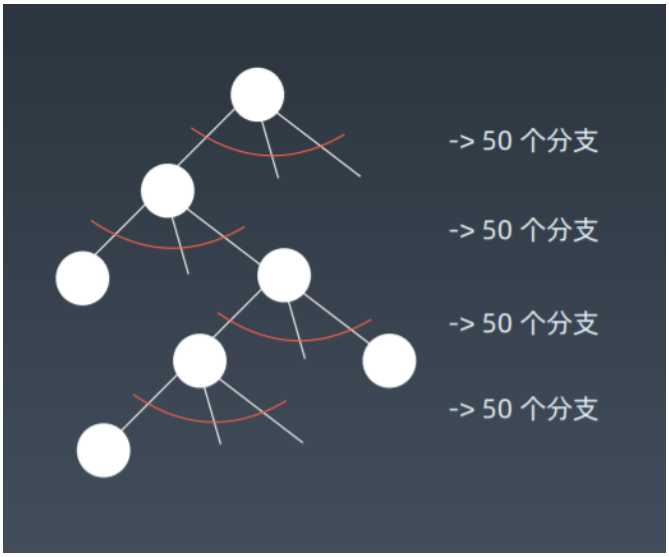
DFS

```java
//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>> results){
        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);
        }
    }
```

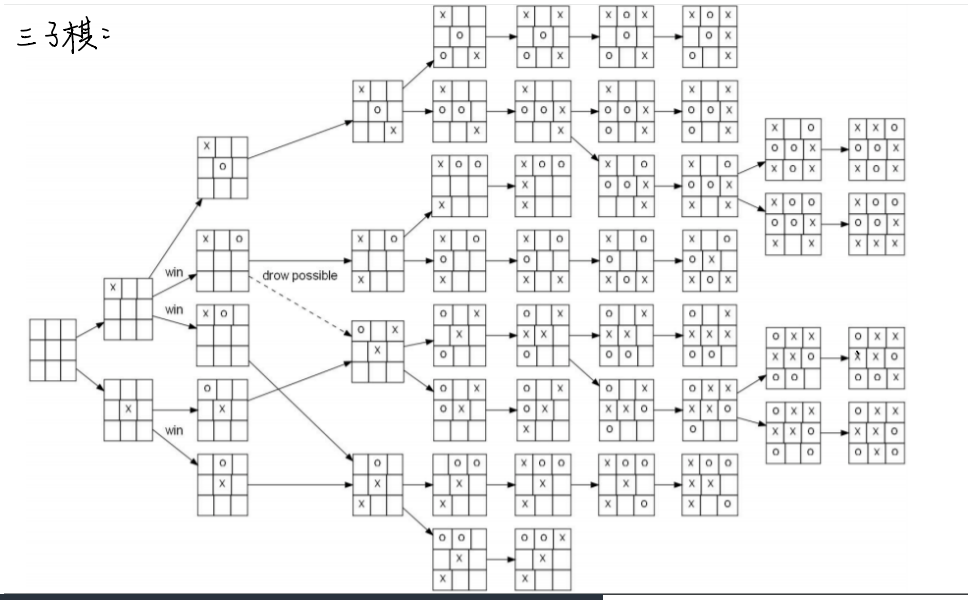
BFS

```java
//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;
}
```

剪枝



三子棋：



其他应用：

AlphaGo

/AlphaZero

回溯法

（方法+试错）

回溯法采用试错的思想，它尝试分步的去解决一个问题。在分步解决问题的过程中，当它通过尝试发现现有的分步答案不能得到有效的正确的解答的时候，它将取消上一步甚至是上几步的计算，再通过其它的可能的分步解答再次尝试寻找问题的答案。

回溯法通常用最简单的递归方法来实现，在反复重复上述的步骤后可能出现两种情况：

- 找到一个可能存在的正确的答案
- 在尝试了所有可能的分步方法后宣告该问题没有答案

在最坏的情况下，回溯法会导致一次复杂度为指数时间的计算。