DFS - Template II

Report Issue (https://github.com/LeetCode-Feedback/LeetCode-Feedback/issues)

The advantage of the recursion solution is that it is easier to implement. However, there is a huge disadvantage: if the depth of recursion is too high, you will suffer from `stack overflow`. In that case, you might want to use BFS instead or implement DFS using an explicit stack.

Here we provide a template using an explicit stack:

Java

Copy

```java
/*
 * Return true if there is a path from cur to target.
 */
boolean DFS(int root, int target) {
    Set<Node> visited;
    Stack<Node> stack;
    add root to stack;
    while (stack is not empty) {
        Node cur = the top element in stack;
        remove the cur from the stack;
        return true if cur is target;
        for (Node next : the neighbors of cur) {
            if (next is not in visited) {
                add next to visited;
                add next to stack;
            }
        }
    }
    return false;
}
```

The logic is exactly the same with the recursion solution. But we use `while` loop and `stack` to simulate the `system call stack` during recursion. Running through several examples manually will definitely help you understand it better.