LeetCode 301

https://leetcode.com/problems/remove-invalid-parentheses/description/

Yifeng Zeng

Description

301. Remove Invalid Parentheses

Idea Report

The basic idea is to check if current string s is valid. If it is, add to the output. If it is not, remove any one character and see if the substring is valid or not. And treat substring as input s and redo the above process. We can do both BFS or DFS. To save time, we can use a hash table to store any string that has already been searched. So for any substring that is in the hash table, we do not need to search again.

Code

```
class Solution {
    // BFS AC
    public List<String> removeInvalidParentheses(String s) {
        List<String> res = new ArrayList<>();
        if (s == null) {
            return res;
        }

        // BFS
        Deque<String> q = new LinkedList<>();
        Set<String> visited = new HashSet<>();
        q.offer(s);
        visited.add(s);

        while (!q.isEmpty()) {
            int size = q.size();
            for (int j = 0; j < size; j++) {
                String cur = q.poll();
                if (isValid(cur)) {
                    res.add(cur);
                }
```

```
                for (int i = 0; i < cur.length() && res.size() == 0; i++) {
                    if (cur.charAt(i) != '(' && cur.charAt(i) != ')') {
                        continue;
                    }
                    String substring = cur.substring(0, i)
                                       + cur.substring(i + 1);
                    if (!visited.contains(substring)) {
                        q.offer(substring);
                        visited.add(substring);
                    }
                }
            }
        }

        return res;
    }

    private boolean isValid(String s) {
        int count = 0;
        for (char ch : s.toCharArray()) {
            if (ch == '(') {
                count++;
            } else if (ch == ')') {
                count--;
            }
            if (count < 0) {
                return false;
            }
        }
        return count == 0;
    }
}
```

Code

```
class Solution {
    // DFS AC
    public List<String> removeInvalidParentheses(String s) {
        List<String> res = new ArrayList<>();
        Set<String> visited = new HashSet<>();
        helper(res, s, visited);
        return res;
    }

    private void helper(List<String> res, String s, Set<String> visited) {
        if (res.size() != 0 && s.length() < res.get(0).length()) {
            return;
```

```
        }

        if ((res.size() == 0 || s.length() >= res.get(0).length())
            && isValid(s)) {
            if (res.size() != 0 && s.length() > res.get(0).length()) {
                res.clear();
            }
            res.add(s);
            return;
        }

        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (ch != '(' && ch != ')') {
                continue;
            }
            String substring = s.substring(0, i) + s.substring(i + 1);
            if (!visited.contains(substring)) {
                visited.add(substring);
                helper(res, substring, visited);
            }
        }
    }

    private boolean isValid(String s) {
        int count = 0;
        for (char ch : s.toCharArray()) {
            if (ch == '(') {
                count++;
            } else if (ch == ')') {
                count--;
            }
            if (count < 0) {
                return false;
            }
        }
        return count == 0;
    }
}
```

Summary

- Try modulize the code using isValid().
- Use a visited hash table to prune the search.