2020 카카오 - 괄호변환

문제

문제 설명

카카오에 신입 개발자로 입사한 "콘"은 선배 개발자로부터 개발역량 강화를 위해 다른 개발자가 작성한 소스 코드를 분석하여 문제점을 발견하고 수정하라는 업무 과제를 받았습니다. 소스를 컴파일하여 로그를 보니 대부분 소스 코드 내 작성된 괄호가 개수는 맞지만 짝이 맞지 않은 형태로 작성되어 오류가 나는 것을 알게 되었습니다.
수정해야 할 소스 파일이 너무 많아서 고민하던 "콘"은 소스 코드에 작성된 모든 괄호를 뽑아서 올바른 순서대로 배치된 괄호 문자열을 알려주는 프로그램을 다음과 같이 개발하려고 합니다.

용어의 정의

'(' 와 ')' 로만 이루어진 문자열이 있을 경우, '(' 의 개수와 ')' 의 개수가 같다면 이를 **균형잡힌 괄호 문자열** 이라고 부릅니다.
그리고 여기에 '('와 ')'의 괄호의 짝도 모두 맞을 경우에는 이를 **올바른 괄호 문자열** 이라고 부릅니다.
예를 들어, `"(()))("` 와 같은 문자열은 "균형잡힌 괄호 문자열" 이지만 "올바른 괄호 문자열"은 아닙니다.
반면에 `"(()))()"` 와 같은 문자열은 "균형잡힌 괄호 문자열" 이면서 동시에 "올바른 괄호 문자열" 입니다.

'(' 와 ')' 로만 이루어진 문자열 w가 "균형잡힌 괄호 문자열" 이라면 다음과 같은 과정을 통해 "올바른 괄호 문자열"로 변환할 수 있습니다.

```
1. 입력이 빈 문자열인 경우, 빈 문자열을 반환합니다.
2. 문자열 w를 두 "균형잡힌 괄호 문자열" u, v로 분리합니다. 단, u는 "균형잡힌 괄호 문자열"로 더 이상 분리할 수 없어야 하며, v는 빈 문자열이 될 수 있습니다.
3. 문자열 u가 "올바른 괄호 문자열" 이라면 문자열 v에 대해 1단계부터 다시 수행합니다.
  3-1. 수행한 결과 문자열을 u에 이어 붙인 후 반환합니다.
4. 문자열 u가 "올바른 괄호 문자열"이 아니라면 아래 과정을 수행합니다.
  4-1. 빈 문자열에 첫 번째 문자로 '('를 붙입니다.
  4-2. 문자열 v에 대해 1단계부터 재귀적으로 수행한 결과 문자열을 이어 붙입니다.
  4-3. ')'를 다시 붙입니다.
  4-4. u의 첫 번째와 마지막 문자를 제거하고, 나머지 문자열의 괄호 방향을 뒤집어서 뒤에 붙입니다.
  4-5. 생성된 문자열을 반환합니다.
```

"균형잡힌 괄호 문자열" p가 매개변수로 주어질 때, 주어진 알고리즘을 수행해 **"올바른 괄호 문자열"**로 변환한 결과를 return 하도록 solution 함수를 완성해 주세요.

매개변수 설명

- p는 '(' 와 ')' 로만 이루어진 문자열이며 길이는 2 이상 1,000 이하인 짝수입니다.
- 문자열 p를 이루는 '(' 와 ')' 의 개수는 항상 같습니다.
- 만약 p가 이미 "올바른 괄호 문자열"이라면 그대로 return 하면 됩니다.

풀이과정

사실 처음에 이 문제를 봤을 땐 DFS를 떠올리지 못하고, 괄호니까 스택만 생각하다가 조금 꼬였던 것 같다. 문제에서 절차도 다 줬는데 조금만 더 생각했더라면 삽질은 덜했을텐데...

재귀라는 키워드를 보고 DFS로 풀어보았다.

소스코드

```
package KAKAO2020parenthesis;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Stack;

public class Main {

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String p = br.readLine();

        String result = solution(p);
        System.out.println(result);
        br.close();
    }

    public static String solution(String p) {
        String answer = dfs(p);
        return answer;
    }

    public static boolean isCorrect(String p){
        Stack<Character> check = new Stack<>();
        for(int i=0 ; i<p.length() ; i++){
            if(p.charAt(i) == '(') check.push('(');
            else if(!check.isEmpty()) check.pop();
            else return false;
        }
        return true;
    }
```

```
    public static String dfs(String p){
        // 1단계
        if(p.length() == 0) return "";

        int open = 0, close = 0;
        for(int i=0 ; i<p.length() ; i++){
            if(p.charAt(i) == '(') open++;
            else close++;

            // 2단계
            if(open == close){

                // 3단계
                if(isCorrect(p.substring(0, i+1))) {

                    // 3-1단계
                    return p.substring(0, i + 1) + dfs(p.substring(i + 1));
                }

                // 4단계
                else {

                    // 4-1 ~ 4-3단계
                    String v = '(' + dfs(p.substring(i+1))+ ')';

                    // 4-4단계
                    String tmp = "";
                    for(int j=1 ; j<i ; j++){
                        if(p.charAt(j) == '(')
                            tmp += ')';
                        else
                            tmp += '(';
                    }
                    // 4-5단계
                    return v + tmp;
                }
            }
        }
        return p;
    }
}
```

함수의 구성

1. boolean isCorrect(String p)
    a. 올바른 괄호 문자열인지 검사하는 함수
2. String dfs(String p)

    a. 문제에서 나온 절차대로 괄호 문자열을 검사하고 조합하는 함수 (DFS 알고리즘 사용)

3. String solution(String p)
    a. dfs 함수를 호출하는 곳

우리가 주요 포인트로 봐야할 것은 dfs 함수이다. 문제에서 나온 절차를 따라가며 함수를 만들었다.

dfs 함수를 단계별로 나누어 살펴보자.

String dfs 함수

1단계

```
// 1단계 - 입력이 빈 문자열인 경우, 빈 문자열 반환
        if(p.length() == 0) return "";
```

2단계

```
// 2단계 - 문자열 w를 두 "균형잡힌 괄호 문자열" u, v로 분리
// 단, u는 "균형잡힌 괄호 문자열"로 더 이상 분리할 수 없어야 하며,
// v는 빈 문자열이 될 수 있다.
int open = 0, close = 0;
        for(int i=0 ; i<p.length() ; i++){
            if(p.charAt(i) == '(') open++;
            else close++;

            // 2단계
            if(open == close){
               ...
```

3단계

```
// 3단계 - 문자열 u가 "올바른 괄호 문자열" 이라면 문자열 v에 대해 1단계부터 다시 수행
if(isCorrect(p.substring(0, i+1))) {

    // 3-1단계 - 수행한 결과 문자열을 u에 이어 붙인 후 반환
    return p.substring(0, i + 1) + dfs(p.substring(i + 1));
}
```

4단계

```
// 4단계 - 문자열 u가 "올바른 괄호 문자열"이 아니라면 아래 과정을 수행
else {

    // 4-1 ~ 4-3단계 - 빈 문자열에 '(' + v에 대해 1단계를 재귀적으로 수행한 문자열 + ')' 붙이기
    String v = '(' + dfs(p.substring(i+1))+ ')';

    // 4-4단계 - u의 첫 번째와 마지막 문자를 제거하고,
    // 나머지 문자열의 괄호 방향을 뒤집어서 뒤에 붙인다.
    String tmp = "";
    for(int j=1 ; j<i ; j++){
        if(p.charAt(j) == '(')
            tmp += ')';
        else
            tmp += '(';
    }
    // 4-5단계 - 생성된 문자열 반환
    return v + tmp;
}
```

문제에서 절차를 안 알려줬다면 아마 삽질을 많이 했을 것 같다. 카카오 코테 1~2번에 이런 유형의 문제가 자주 나온다고 하니 프로그래머스에 있는 카카오 문제들도 많이 풀어봐야겠다.